Définitions par réécriture dans le lambda-calcul : confluence, réductibilité et typage

Colin Riba

Nancy-Université INPL

Département de formation doctorale en informatique

École doctorale IAEM Lorraine

Définitions par réécriture dans le lambda-calcul : confluence, réductibilité et typage

THÈSE

présentée et soutenue publiquement le 14 décembre 2007

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(spécialité informatique)

par

Colin Riba

Composition du jury

Président :	Gilles Barthe	Directeur de recherche, INRIA, Sophia-Antipolis, France
Rapporteurs :	Thierry Coquand	Professeur, Chalmers University, Göteborg, Suède
	Delia Kesner	Professeur, Université Paris 7, Paris, France
Examinateurs :	Mariangiola Dezani	Professeur, Università di Torino, Italie
	Jean-Yves Marion	Professeur, École des Mines de Nancy, INPL, Nancy, France
	Frédéric Blanqui (dir.)	Chargé de recherche, INRIA, Nancy, France
	Claude Kirchner (dir.)	Directeur de recherche, INRIA, Nancy, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe scrbook.

It is more important that a proposition be interesting than that it be true.

Alfred North Whitehead
Adventures of Ideas

Permettez-moi de vous répéter ce que je vous disais ici : traiter la nature par le cylindre, la sphère, le cône, le tout mis en perspective... et les lignes parallèles à l'horizon donnent l'étendue, soit une réaction à la nature. Les lignes perpendiculaires à cet horizon donnent de la profondeur. Car la nature, pour nous les hommes, est plus en profondeur qu'en surface.

Lettre de Georges Braque à Émile Bernard

Remerciements

Je tiens tout d'abord à remercier Frédéric Blanqui et Claude Kirchner pour m'avoir accueilli à Nancy et m'avoir patiemment guidé pendant ces trois années. En particulier Frédéric pour son soutient sans failles, son engagement et la qualité de son encadrement ; Claude pour l'environnement de travail excellent qu'il a su créer au sein de l'équipe Protheo et l'attention qu'il a toujours porté à mon travail malgré ses multiples occupations.

Je remercie Thierry Coquand et Delia Kesner, qui on eu la gentillesse de bien vouloir rapporter cette thèse. Delia Kesner pour les discussions enrichissantes que l'on a pu avoir ainsi que pour l'intérêt qu'elle a manifesté pour mon travail.

Je tiens aussi a remercier les membres de mon jury, tout d'abord pour avoir accepté d'en faire partie. Mariangiola Dezani pour la gentillesse dont elle fait preuve a mon égard ainsi que pour l'intérêt des discussions que nous avons pu avoir ; Jean-Yves Marion pour quelques discussions intéressantes et ludiques ; et Gilles Barthe, ainsi que le projet Everest à Sophia-Antipolis pour m'avoir accueilli dès avant ma soutenance.

Cette thèse doit énormément aux univers scientifiques dans lesquels elle s'insère, et dont je ne peux énumérer les membres. Mais je me dois de remercier Alexandre Miquel pour m'avoir introduit aux biorthogonaux pendant l'école d'été Types 2005. Le chapitre 10 de cette thèse doit énormément à cette rencontre. De manière plus brève mais tout aussi déterminante, ce chapitre doit beaucoup à l'exemple que m'a communiqué Philippe de Groote. Je tiens aussi à remercier Andreas Abel pour les nombreux échanges que nous avons eu et les remarques pertinentes qu'il a pu faire sur mon travail.

Je remercie les membres de l'équipe PPS de m'avoir offert un espace appréciable pour présenter mes travaux, en particulier Delia Kesner, Alexandre Miquel et Paul-André Melliès. J'ai aussi eu le plaisir d'avoir été longuement cuisiné par les membres de l'équipe Plume du LIP, en particulier par Daniel Hirschkoff, ainsi que par leurs comparses de l'équipe de logique du LAMA, René David et Christophe Raffalli. Merci aussi au membres du projet LogiCal, en particulier Gilles Dowek pour quelques discussions aux cours desquelles j'ai essayé de m'imprégner de sa clarté de pensée, et Jean-Pierre Jouannaud pour l'interêt qu'il a porté à mes travaux. Je voudrais enfin remercier Rachid Echahed pour avoir guidé mes premiers pas en recherche, ainsi que pour le temps qu'il m'a offert pour présenter mes travaux à l'équipe CAPP du LIG.

Merci aux membres et thésards de l'équipe Protheo, en particulier Germain pour sa patience lors de discussions presqu'interminables et pour sa curiosité scientifique communicative ; ainsi qu'Antoine, dont le pragmatisme éclairé a souvent dénoué quelques problèmes métaphysiques. Merci aussi à mes anciens co-bureaux Guillaume, Oana et Anderson pour leur compagnie et pour m'avoir supporté. Merci à Yves pour son éclairage nouveau, dont j'espère pouvoir tirer parti un jour, ainsi que pour sa disponibilité lors de la préparation de ma soutenance. Merci enfin à Richard pour son accueil chaleureux

à Nancy, sans lequel la soutenance n'aurait pu avoir lieu.

Je suis très heureux d'avoir pu effectuer ma thèse dans un environnement aussi riche et varié que le Loria. Certaines personnes y ont très agréablement enrichi mes trois années passées à Nancy, tout particulièrement Sébastien, Heinrich, mais aussi Romain et Benjamin, ainsi que beaucoup d'autres. Quelques connaissances nancéenes, en particulier Said, et mes anciens colocataires, dont Fabien.

Enfin ma famille, dont j'ai la chance d'avoir toujours eu le soutient inconditionnel et à qui cette thèse doit énormément. Je ne peux conclure sans remercier Olivier dont la patience, la curiosité et l'exigence lors de mes tentatives pour lui expliquer mon travail ont sûrement beaucoup contribué, bien qu'indirectement, à cette thèse. Une mention spéciale va à Mary Carmen.

Table des matières

Introduction

Cette thèse concerne l'étude de la combinaison de deux formalismes de calculs, la réécriture et le lambda-calcul, tous deux nés à la frontière entre l'informatique et la logique.

Généralités

Notre point de départ pourraient être les travaux de Frege puis de Russell sur les systèmes formels. Poursuivant les idées de Leibniz, ceux-ci ont pour objectif de clarifier le statut des mathématiques, en partant de l'hypothèse que les concepts mathématiques ont une origine logique. L'idée est de pouvoir, en utilisant des transformations symboliques, ramener tout énoncé mathématique à un énoncé logique dont la validité puisse être elle aussi vérifiée de manière symbolique.

Ce projet aboutit à la description de systèmes de raisonnement formels, c'est-à-dire de description symbolique (et discrète) des énoncés mathématiques. Le programme de Hilbert projetait d'utiliser de tels systèmes pour formaliser l'arithmétique, et de donner une preuve de la correction de cette formalisation par des moyens « finitaires ».

Une caractéristique importante des systèmes formels est qu'ils sont eux-mêmes des objets mathématiques. Ainsi, un système formel peut servir de cadre pour la description d'autres systèmes formels, en particulier de lui-même. Le second théorème d'incomplétude de Gödel donne une limite à cette réflexivité : un système formel cohérent et suffisamment puissant pour formaliser l'arithmétique (et donc se formaliser lui même) ne peut prouver sa propre cohérence.

Il s'en suit que tout système formel suffisamment puissant permet de formuler des propositions dont il ne peut établir ni la vérité ni la fausseté, c.-à-d. qui ne sont ni vraies ni fausses de son point de vue. Ainsi, la non prouvabilité d'une proposition dans un système formel n'implique pas sa fausseté.

Cela amène à s'intéresser non plus seulement à la prouvabilité dans un système formel, mais à la manière dont se fait cette prouvabilité, c'est-à-dire à s'intéresser à la notion de preuve. De ce point de vue, l'enjeu se déplace donc vers les systèmes de déduction. La déduction naturelle et le calcul des séquents sont des systèmes de déduction introduits par Gentzen qui mettent l'accent sur un aspect dynamique des preuves : la réduction des coupures. Ceci donne une interprétation calculatoire des déductions, via l'algorithme d'élimination des coupures (ou de normalisation des preuves). Cette interprétation calculatoire est particulièrement pertinente pour la logique intuitionniste. En effet, les règles de la déduction naturelle intuitionniste correspondent à l'explication intuitionniste des connecteurs logiques [ML84].

La notion de calcul apparaît aussi avec des descriptions formelles de machines de calcul, telles que la machine de Turing. Ces machines permettent une formulation d'une variante du « paradoxe » de Gödel : d'une part, il existe une machine de Turing universelle qui peut simuler le comportement de n'importe quelle machine de Turing sur n'importe quelle entrée, et d'autre part il n'existe pas de machine de Turing donnant une réponse observable à la question « Est-ce que telle machine de Turing s'arrête lorsqu'elle est exécutée avec telle entrée ? ». Ainsi, il y a des fonctions qui sont calculables par des machines de Turing et d'autres qui ne le sont pas. La thèse de Turing postule que n'importe quelle fonction « effectivement » calculable est calculable par une machine de Turing.

Notons que la réalisabilité, due à Kleene, offre un cadre pour comparer les notions de calcul de Gentzen et de Turing. Voir [Oos02] pour un rapide survol historique de ce domaine.

Les systèmes formels sont d'une grande importance en informatique. En effet, d'une part ils permettent de formaliser les langages de programmation, et d'autre part la description formelle des systèmes formels permet de les informatiser eux-mêmes. En somme, les systèmes formels donnent des outils pour informatiser l'étude des systèmes informatiques, et donc pour automatiser une partie du raisonnement sur ces systèmes. Cet aspect est un enjeu fondamental pour le développement de systèmes informatiques complexes, surtout dans un contexte où la sûreté de fonctionnement est primordiale.

Objets d'étude

Notre travail concerne l'étude de la réécriture et du lambda-calcul, dont la combinaison est intéressante vis-à-vis du calcul dans le sens de Turing et dans le sens de Gentzen.

Ce sont des formalismes similaires (le lambda-calcul est une forme de réécriture), mais qui correspondent cependant à des approches différentes. Nous renvoyons le lecteur à [Bar84, Kri90] pour plus détails sur le lambda-calcul, et à [DJ90, Ter03] pour la réécriture.

Lambda-calcul

Le lambda-calcul, inventé par Alonzo Church, est un formalisme intéressant par de nombreux aspects.

Tout d'abord, il permet de calculer au sens de Turing : toutes les fonctions sur les entiers calculables par les machines de Turing (c.-à-d. les fonctions récursives de Kleene) y sont codables. C'est un formalisme *d'expressions*, ce qui le rend plus souple et facile à utiliser que les machines de Turing. Par ailleurs, le lambda-calcul dispose d'une stratégie d'évaluation séquentielle, qui trouve la forme normale d'un lambda-terme lorsqu'elle existe.

D'autre part, le lambda-calcul peut être typé. Le typage est une manière d'exprimer une abstraction du comportement d'un lambda-terme en lui associant un type. Les types sont utilisés dans des règles de typage permettant de contrôler la formation des lambda-termes, c.-à-d. la façon dont on peut les composer. L'isomorphisme de Curry-Howard établit une

correspondance entre les termes typables dans certains systèmes de types et les preuves de certains systèmes de déduction naturelle. Par exemple, les termes simplement typés du lambda-calcul pur sont les preuves de la déduction naturelle pour la logique minimale intuitionniste. Cette correspondance peut aussi se retrouver au niveau algorithmique, les règles d'évaluation du calcul correspondant à la réduction des coupures. Par exemple, dans le cas du lambda-calcul simplement typé, la β-réduction correspond exactement à la réduction des coupures du connecteur « implication ». Ainsi, le lambda-calcul est aussi un système de calcul dans le sens de Gentzen.

Ceci est particulièrement intéressant pour la logique intuitionniste, car son caractère constructif permet d'extraire de certaines preuves des programmes corrects par construction. Par exemple, une preuve intuitionniste qu'il existe un algorithme de tri pour les listes contient cet algorithme.

Par ailleurs, il existe des systèmes de types décrivant des propriétés calculatoires des lambda-termes, comme par exemple les types intersection [CD78]. De tels système ne sont en général pas sujets à l'isomorphisme de Curry-Howard, et leur signification logique n'est pas évidente.

Le lambda-calcul est donc un formalisme intéressant pour servir de base à la programmation fonctionnelle ainsi qu'aux assistants à la preuve, qui l'utilisent pour représenter les preuves de systèmes de déduction. Il a cependant certaines limitations. En effet, c'est un formalisme qui décrit essentiellement le *sujet* du calcul. Il donne une description relativement fine d'une procédure de calcul, grâce à laquelle on peut coder beaucoup d'algorithmes. Mais ces codages sont souvent ardus et pas toujours efficaces. Une utilisation pratique du lambda-calcul requiert donc de lui ajouter d'autres moyens de définitions de fonctions. De fait, les langages fonctionnels (typés) OCaml [LDG+07], ML [MTHM97] et Haskell [PJ03], ainsi que les assistants à la preuve comme Coq [Tea06] et Agda [Coq] utilisent des extensions du lambda-calcul.

Réécriture

Une possibilité d'extension du lambda-calcul est d'y ajouter des symboles algébriques et des fonction définies par réécriture. La réécriture est un formalisme qui permet de définir des fonctions de manière équationnelle, simplement en décrivant (récursivement) le résultat de l'application d'une fonction sur tel ou tel schéma d'entrée.

De même que le lambda-calcul, la réécriture est un formalisme Turing complet. Mais à la différence du lambda-calcul, il se préoccupe plus de la manière dont les fonctions sont spécifiées que de la manière dont elles sont évaluées. Ce mode de description est beaucoup plus souple et a priori plus simple à appréhender que le lambda-calcul, mais par contre, il est moins opérationnel. En effet, il n'existe pas pour la réécriture de procédure séquentielle d'évaluation qui trouve toujours la forme normale d'un terme lorsqu'elle existe [HL91].

La réécriture peut donc être vue comme un outil de spécification de fonctions, qui permet de décrire *l'objet* d'un calcul. De ce point de vue, il est intéressant d'étendre la réécriture en réécriture conditionnelle. Si on voit la réécriture comme résultant de l'orientation de spécification équationnelles atomiques, alors la réécriture conditionnelle correspond à l'orientation de spécifications équationnelles utilisant des clauses de Horn. C'est

d'ailleurs cette forme de réécriture qui est utilisée dans les langages Maude [CDE+07] et Elan [BCD+06].

Combinaison du lambda-calcul et de la réécriture

L'intérêt de l'étude de la combinaison du lambda-calcul et de la réécriture est donc d'obtenir un formalisme qui permet de profiter à la fois des caractéristiques logiques et d'ordre supérieur du lambda-calcul, et de la souplesse d'utilisation de la réécriture. Un tel formalisme est intéressant pour le calcul dans le sens de Turing et dans le sens de Gentzen, et peut ainsi servir de base à la fois à des langages de programmation et à des systèmes de preuve.

Comme nous l'avons dit, nous nous intéressons à la combinaison du lambda-calcul et de la réécriture en considérant les deux formalismes au même niveau : le lambda-calcul étend la réécriture et la réécriture étend le lambda-calcul.

Lorsqu'on se place dans le cadre de l'isomorphisme de Curry-Howard, cela permet d'ajouter au lambda-calcul des constructions reflétant au niveau des preuves certains connecteurs logiques. C'est le cas par exemple du lambda-calcul avec produits, qui correspond à la logique intuitionniste minimale avec conjonctions.

D'autre part, dans un système de déduction, la réécriture peut aussi être intégrée au niveau des types. C'est typiquement le cas de la déduction modulo [DHK03], dans laquelle les propositions sont identifiées par une congruence qui peut être issue d'un système de réécriture.

Ces deux niveaux d'intégration de la réécriture ne sont pas nécessairement indépendants. Par exemple, dans les systèmes avec types dépendants comme le calcul des constructions [CH88], les types peuvent dépendre des termes de preuves. Ainsi, un système de réécriture au niveau des preuves induit une congruence au niveau des types.

Ces deux niveaux peuvent aussi cohabiter dans un même système. Par exemple, l'assistant à la preuve Coq est basé sur une extension du calcul des constructions, le calcul des constructions inductives [CP88], qui utilise un mécanisme de définitions inductives dont les règles de réduction réécrivent aussi bien des types (élimination forte) que des termes (élimination faible).

Enfin, la combinaison du lambda-calcul et de la réécriture peut être un cadre syntaxique pour la réalisabilité, comme par exemple dans [BBC98]. Nous n'avons pas abordé cette question, bien que les preuves de normalisation forte du lambda-calcul typé que nous étudions aux chapitres 9 et 10 utilisent la « réductibilité[1] », qui est une technique issue de la réalisabilité.

Cohérence équationnelle et cohérence déductive

Les notions de calcul de Turing et de Gentzen donnent lieu, dans le cadre de la combinaison de la réécriture et du lambda-calcul, à deux notions cohérences.

— La « cohérence équationnelle » est la propriété que la théorie équationnelle décrite par une relation de réécriture n'est pas triviale.

[1] « reducibility » en Anglais.

— La « cohérence déductive » est la propriété que, via l'isomorphisme de Curry-Howard, le système de déduction correspondant à une extension du lambda-calcul est cohérent.

Au regard de ces deux formes de cohérences, nous étudions deux propriétés de la combinaison de la réécriture et du lambda-calcul :

— la confluence, qui permet d'assurer la cohérence équationnelle et le déterminisme des calculs ;
— la normalisation forte, qui permet d'assurer la cohérence déductive et la terminaison des calculs.

Le plus souvent, la normalisation faible suffit pour obtenir la cohérence déductive d'un système de types vu comme un système logique. Cependant, comme la réécriture n'a en général pas de moyen d'évaluation effectif, dans notre cas il est important du point de vue de la terminaison d'étudier la normalisation forte plutôt que la normalisation faible. D'autre part, la normalisation forte est intéressante en pratique car elle permet le choix d'une stratégie de normalisation, qui peut donc être fait sur des critères d'efficacité. De plus, elle permet, par lemme de Newman, d'assurer la confluence à partir de la confluence locale.

Enfin, rappelons que la cohérence équationnelle et la cohérence déductive sont liées.

— Dans des systèmes avec types dépendants, l'incohérence équationnelle peut amener à une relation d'égalité incohérente et à l'incohérence déductive.
— Comme on le verra au chapitre 6, l'incohérence déductive du lambda-calcul non typé (c'est-à-dire « uni-typé », voir par exemple [AC98]), peut amener à l'incohérence calculatoire en présence de réécriture.

Survol du contenu

Partie I Réécriture conditionnelle et lambda-calcul

Dans cette première partie nous présentons les objets de base sur lesquels nous allons travailler.

Au chapitre 1, nous définissons précisément les termes que nous utilisons. Les termes du lambda-calcul contiennent un lieur de variables qui impose d'identifier les termes qui ne diffèrent que par renommage de leur variables liées. Nous utilisons une relation d'alpha-conversion inspirée de celle de Krivine [Kri90] pour expliciter les liens entre les algèbres de termes du premier ordre (utilisés pour la réécriture du premier ordre) et les termes quotientés par alpha-conversion. En particulier, nous montrons précisément comment une algèbre de termes du premier ordre peut être vue comme un sous-ensemble des lambda-termes quotientés par alpha-conversion.

Le chapitre 2 rappelle des notations concernant les relations et systèmes de réécriture.

Nous présentons au chapitre 3 le lambda-calcul et la réécriture. Nous insistons sur la manière dont s'articulent ces deux formalismes, et essayons de motiver l'étude de leur combinaison. En partant de la réécriture du premier ordre, nous passons au lambda-calcul non typé, puis au lambda-calcul typé. Nous rappelons les limitations des codages possibles dans ces systèmes, ce qui motive le fait de travailler sur des extensions du

lambda-calcul, puis nous rappelons le théorème de Böhm, qui suggère que ces extensions soient aussi des extensions de la syntaxe du lambda-calcul. Une possibilité est d'étendre la syntaxe du lambda-calcul par des symboles algébriques et d'étendre la β-réduction par des fonctions définies par réécriture sur ces symboles.

Le chapitre 4 est consacré à la réécriture conditionnelle. Nous montrons en particulier que pour un système conditionnel du premier ordre, la relation de réécriture conditionnelle sur les lambda-termes est le quotient par alpha-conversion de la relation de réécriture conditionnelle sur les termes du premier ordre.

Partie II Confluence

Dans cette seconde partie, nous étudions la confluence de la combinaison du lambda-calcul et de la réécriture conditionnelle.

Nous commençons au chapitre 5 par quelques rappels sur l'utilisation des relations de réécriture parallèles pour la confluence des systèmes de réécriture orthogonaux et du lambda-calcul. Ensuite, nous montrons comment ces notions sont étendues à la réécriture conditionnelle, ce qui nous permet d'aborder brièvement la différence entre confluence et confluence par niveaux.

Enfin, nous rassemblons quelques points importants liés à la modularité. Cela nous permet de caractériser la confluence par la confluence close et d'évoquer la possibilité d'un lien entre d'une part le fait que la confluence n'est pas préservée lorsqu'on ajoute aux termes du premier ordre des éléments d'une algèbre non libre (comme par exemple des termes infinis), et d'autre part le fait que la confluence n'est en général pas préservée par combinaison de systèmes non disjoints. De plus, en utilisant nos développements sur l'alpha-conversion, nous montrons que la confluence de la réécriture conditionnelle du premier ordre est préservée lorsqu'on passe des termes du premier ordre aux lambda-termes quotientés par alpha-conversion. Ceci est intéressant pour compléter la preuve de préservation de la confluence par curryfication de [Kah95], qui ne prend pas en compte cette question.

Au chapitre 6, nous présentons quelques études systématiques sur la confluence de la combinaison du lambda-calcul et de la réécriture. Nous abordons de plus la question de la combinaison de la réécriture du premier ordre avec l'η-réduction du lambda-calcul, et donnons une preuve qui nous semble nouvelle du fait, que si $\mathcal{R}$ est un système de réécriture du premier ordre, alors la cohérence de $=_{\mathcal{R}}$ implique la cohérence de $=_{\beta\eta\mathcal{R}}$ sur les termes typés dans le système F [BT88, BTG89, BTG94]. Cela implique que, sur les termes typés, la $\beta\eta\mathcal{R}$-conversion est cohérente lorsque la $\mathcal{R}$-conversion l'est aussi, bien que la $\beta\eta\mathcal{R}$-réduction ne soit par toujours confluente lorsque que la $\beta\mathcal{R}$-réduction est confluente.

Le chapitre 7 est consacré à l'étude de la préservation de la confluence pour la réécriture conditionnelle combinée au lambda-calcul. Notre approche a été essentiellement d'étendre à la réécriture conditionnelle les résultats présentés au chapitre 6 sur la combinaison du

lambda-calcul à la réécriture non-conditionnelle. Nous avons de plus amélioré un résultat de Dougherty [Dou92] sur la confluence de la réécriture combinée au lambda-calcul. Le résultat original montre, pour un système de réécriture du premier ordre confluent $\mathcal{R}$, que la $\beta\mathcal{R}$-réduction est confluente sur les termes fortement β-normalisants satisfaisant certaines conditions. Nous avons étendu le résultat aux termes faiblement β-normalisants satisfaisant des conditions similaires. Le contenu de ce chapitre a été publié dans [BKR06].

Enfin, au chapitre 8 nous présentons des résultats préliminaires sur l'extension à la réécriture conditionnelle du résultat de Kahrs [Kah95] sur la préservation de la confluence par curryfication de la réécriture non conditionnelle. Nous montrons de plus que la confluence sur les termes clos n'est en général pas préservée par curryfication de la réécriture non conditionnelle.

Partie III Normalisation forte et réductibilité

Cette troisième partie concerne les preuves par réductibilité de la normalisation forte de la réécriture combinée au lambda-calcul typé. Dans ces preuves, les types sont interprétés par des ensembles de termes fortement normalisants, et on montre que l'interprétation est adéquate, c.-à-d. que les termes typables appartiennent à l'interprétation de leur type.

Le chapitre 9 est un exposé général des preuves de normalisation forte par réductibilité. Nous commençons par le lambda-calcul simplement typé avec produits, puis nous présentons quelques point importants sur l'insertion de la réécriture dans ces preuves.

Nous abordons enfin le système F [Gir72], dont la spécificité logique amène à envisager la réductibilité sous forme de familles d'ensembles de termes fortement normalisants vérifiant certaines conditions de clôture. Nous appelons ces familles d'ensembles des « familles de réductibilité ». Nous explicitons un lien entre l'utilisation d'une intersection dans l'interprétation par réductibilité du système F et la non terminaison du système F combiné à certains systèmes de réécriture polymorphes non paramétriques. De plus, nous donnons une généralisation des candidats de réductibilité de Girard qui utilise une notion de terme neutre basée sur l'interaction entre une relation de réécriture et des « contextes d'élimination ».

Dans le chapitre 10, nous discutons la stabilité par union de familles de réductibilité. Nous présentons un système de types intersection et union qui est correct et complet vis-à-vis de la normalisation forte de systèmes de réécriture « simples » combinés au lambda-calcul. Cela nous permet de voir que pour certains systèmes de réécriture simples confluents, on peut typer des termes non fortement normalisants si on ajoute au système une règle d'élimination des types union. Comme cette règle est validée par toute famille de réductibilité stable par union, nous en déduisons qu'il existe des systèmes de réécriture simples confluents qui n'admettent pas de famille de réductibilité stable par union.

Nous étudions ensuite des conditions pour la stabilité par union de familles de réductibilité. Nous donnons une condition nécessaire et suffisante sur une relation de réécriture et un ensemble de contextes d'élimination pour que les candidats de réductibilité de Gi-

rard soient stables par union. Nous donnons de plus une condition nécessaire et suffisante pour que la clôture par union d'une famille de biorthogonaux forme une famille de réductibilité, et montrons à l'aide de la réécriture simple que cette condition est strictement plus générale que la condition de stabilité par union des candidats de réductibilité de Girard. Ceci nous permet de raffiner l'incorporation de la réécriture dans les ensembles saturés de Tait, et ainsi de comparer plus précisément les ensembles saturés de Tait aux candidats de Girard en présence de réécriture.

Notons que notre condition pour la stabilité par union des candidats de Girard nous renseigne de manière intéressante sur leur structure « algébrique » : ils sont stables par union exactement lorsque ce sont les ensembles non vides de termes fortement normalisants clos par le bas pour un ordre d'observation faible.

Enfin, nous donnons une interprétation des types par des ensembles clos par biorthogonalité, qui, bien que non stable par union, est « complète » vis-à-vis de la normalisation forte dans notre système avec élimination des types unions : pour tout système de réécriture simple, la typabilité avec élimination de l'union implique la normalisation forte si et seulement si cette interprétation est adéquate pour le typage dans ce système. Cela donne une forme de contenu calculatoire aux biorthogonaux car le typage avec la règle d'élimination de l'union spécifie une propriété de « bonne interaction » entre les réduits d'un terme neutre lorsqu'il est substitué dans un terme fortement normalisant.

Une grande partie du contenu de ce chapitre a été publiée dans [Rib07a, Rib07b].

Le chapitre 11 est consacré à la présentation d'un critère de terminaison pour la combinaison de la réécriture conditionnelle au lambda-calcul simplement typé avec produits et types inductifs du premier ordre. C'est à notre connaissance le premier critère de terminaison pour la réécriture conditionnelle (d'ordre supérieur) qui permette de prendre en compte, dans l'argument de terminaison, l'information donnée par la satisfaction des conditions des règles de réécriture.

Il repose sur un système de types contraints par des formules de l'arithmétique de Presburger avec booléens, qui contient notamment des types contraints existentiels. Nous présentons un schéma d'algorithmes pour la vérification du typage dans ce système. Ce schéma d'algorithmes utilise un processus « bidirectionnel » de vérification et de synthèse de types, qui fonctionne par génération de contraintes dans l'arithmétique de Presburger. La sémantique par réductibilité de ce système repose sur l'interprétation des types existentiels par des unions et sur une interprétation « singleton » des types inductifs du premier ordre.

D'autre part, nous montrons comment utiliser notre notion de candidats de réductibilité avec des types inductifs en utilisant des « destructeurs » de types inductifs.

Un grande partie du contenu de ce chapitre a été publiée dans [BR06].

Notations et notions de base

Nous rappelons ici quelques notions de base.

Relations

Si A est un ensemble et $n \in \mathbb{N}$ alors A^n est défini par induction sur n comme suit

$$\begin{array}{lll} A^0 & =_{\text{def}} & \emptyset \\ A^{n+1} & =_{\text{def}} & A \times A^n \ . \end{array}$$

Nous appelons *relations* les couples (A, R) où A est un ensemble et $R \in \mathcal{P}(A)$ ($\mathcal{P}(A)$ est l'ensemble des parties de A). Si $n \in \mathbb{N}$, une *relation n-aire* sur $A_1, \ldots, A_n$ est une relation $(A_1 \times \cdots \times A_n, R)$. Si $A_1 = \cdots = A_n = A$, on dit que R est une relation n-aire sur A et on désigne (A^n, R) par (A, R).

Lorsque R est une relation binaire, il est souvent agréable d'écrire $a\,R\,b$ pour $(a, b) \in R$. Si $(A \times B, R)$ et $(A \times B, R')$ sont des relations binaires, alors $(A \times B, R \cup R')$, $(A \times B, R \cap R')$ et $(A \times B, R \setminus R')$ en sont aussi. *La composition* de $(A \times B, R)$ et $(B \times C, R')$ est la relation binaire $(A \times C, R \cdot R')$ définie par

$$R \cdot R' \quad =_{\text{def}} \quad \{(a, c) \mid \exists b \in B. \quad a\,R\,b \quad \wedge \quad b\,R'\,c\} \ .$$

La relation $(A \times C, R \cdot R')$ est aussi notée $(A \times C, RR')$.

Relations d'équivalence

Une relation binaire (A, R) est une *relation d'équivalence* si elle est

réflexive : $a\,R\,a$ pour tout $a \in A$,

transitive : ($a\,R\,b$ et $b\,R\,c$) implique $a\,R\,c$ pour tout $a, b, c \in A$,

symétrique : $a\,R\,b$ implique $b\,R\,a$ pour tout $a, b \in A$.

Si (A, R) est une relation d'équivalence, alors le *quotient* de A par R, noté A/R est l'ensemble des classes d'équivalence pour R :

$$B \in A/R \qquad \text{si et seulement si} \qquad \exists a \in A. \quad B \quad = \quad \{b \in A \mid a\,R\,b\} \ .$$

Fonctions

Si A et B sont deux ensembles, une *fonction de A dans B* est une relation $(A \times B, F)$ telle que

$$\forall x \in A.\, \forall y, y' \in B. \quad \big(x\,F\,y \quad \wedge \quad x\,F\,y'\big) \quad \Longrightarrow \quad y = y' \ .$$

On écrit $F : A \to B$ pour dire que F est une fonction de A dans B. *Le domaine* de F est la partie $\mathcal{D}om(F)$ de A telle que

$$\forall x.\quad x \in \mathcal{D}om(F) \qquad \text{si et seulement si} \qquad \exists y.\quad x\,F\,y\ .$$

La fonction F est *totale* si $\mathcal{D}om(F) = A$ et *partielle* sinon. Pour tout $x \in \mathcal{D}om(F)$, on écrit $F(x)$ pour désigner l'unique $y \in B$ tel que $x\,F\,y$.

Il est pratique de convenir que les *valuations* sont des fonctions de *domaine fini*, et que les *assignements* sont des fonctions *totales*.

Si A et I sont deux ensembles, *une famille d'éléments de* A *indexée par* I est une fonction totale $F : I \to A$. On désigne F par $(a_i)_{i\in I}$, où $a_i = F(i)$ pour tout $i \in I$.

Relations bien fondées

La *partie stricte* d'une relation binaire $(A, \leq)$ est la relation binaire $(A, <)$ telle que

$$a < b \qquad \text{si} \qquad (a \leq b \quad \text{et} \quad b \not\leq a)\ .$$

L'*équivalence* engendrée par la relation binaire $(A, \leq)$ est la relation binaire $(A, \simeq)$ telle que

$$a \simeq b \qquad \text{si} \qquad (a \leq b \quad \text{et} \quad b \leq a)\ .$$

Une relation binaire $(A, \leq)$ est dite

— *quasi bonne* si pour tout $X \subseteq A$ non vide, il existe $a \in X$ tel que

$$\forall b \in X.\quad b \not< a\ .$$

Un tel a est dit *minimal*.

— *bien fondée* s'il n'existe pas de suite infinie strictement décroissante, c'est-à-dire de famille $(a_i)_{i\in\mathbb{N}}$ telle que $a_i > a_{i+1}$ pour tout $i \in \mathbb{N}$.

Il est aisé de voir qu'une relation quasi bonne est bien fondée. Supposons l'existence d'une suite infinie strictement décroissante $(a_i)_{i\in\mathbb{N}}$. Alors l'ensemble $X =_{\text{def}} \{a_i \mid i \in \mathbb{N}\}$ a un élément minimal a_{i_0}, donc en particulier $a_{i_0+1} \not< a_{i_0}$, ce qui contredit le fait que $(a_i)_{i\in\mathbb{N}}$ est une suite infinie strictement décroissante.

L'inverse est vrai si on dispose d'une *fonction de choix* h qui à chaque partie non vide X de A associe $h(X) \in X$. Supposons donc que $(A, \leq)$ soit une relation bien fondée qui n'est pas quasi bonne. Il existe donc $X \subseteq A$, non vide qui n'a pas d'élément minimal. On construit alors une suite infinie strictement décroissante comme suit :

$$\begin{array}{lll} a_0 & =_{\text{def}} & h(X) \\ a_{i+1} & =_{\text{def}} & h(\{b \in X \mid b < a_i\})\ . \end{array}$$

Proposition 1 (Principe d'induction) *Soit* $(A, \leq)$ *une relation quasi bonne et* P *une partie de* A *vérifiant*

$$\forall a.\ (\forall b.\ b < a \implies b \in P) \implies a \in P\ .$$

Alors $A = P$.

On appelle *extension produit* des relations binaires $(A_1, R_1), \ldots, (A_n, R_n)$ la relation $(R_1, \ldots, R_n) \subseteq A^n \times A^n$ définie par

$$(a_1, \ldots, a_n) \quad (R_1, \ldots, R_n) \quad (b_1, , \ldots, b_n)$$
$$\text{si et seulement si} \qquad (\exists i.\ \ a_i \, R_i \, b_i \quad \wedge \quad \forall j \neq i.\ \ a_j = b_j) \ .$$

Lorsque $(A_1, R_1) = \cdots = (A_n, R_n) = (A, R)$, on écrit

$$(a_1, \ldots, a_n) \quad R \quad (b_1, \ldots, b_n)$$

au lieu de

$$(a_1, \ldots, a_n) \quad (R, \ldots, R) \quad (b_1, \ldots, b_n) \ .$$

L'extension lexicographique des relations binaires $(A_1, R_1), \ldots, (A_n, R_n)$ est la relation $(R_1, \ldots, R_n)_{\text{lex}} \subseteq A^n \times A^n$ définie par

$$(a_1, \ldots, a_n) \quad (R_1, \ldots, R_n)_{\text{lex}} \quad (b_1, \ldots, b_n)$$
$$\text{si et seulement si} \qquad (\forall i.\ \ a_i \, R_i \, b_i \quad \wedge \quad \forall j < i.\ \ a_j = b_j) \ .$$

Lorsque $(A_1, R_1) = \cdots = (A_n, R_n) = (A, R)$, on écrit

$$(a_1, \ldots, a_n) \quad R_{\text{lex}} \quad (b_1, \ldots, b_n)$$

au lieu de

$$(a_1, \ldots, a_n) \quad (R, \ldots, R)_{\text{lex}} \quad (b_1, \ldots, b_n) \ .$$

Proposition 2 *Les relations* $(R_1, \ldots, R_n)$ *et* $(R_1, \ldots, R_n)_{lex}$ *sont bien fondées dès lors que les relations* $R_1, \ldots, R_n$ *le sont.*

Multiensembles

Les *mutliensembles* permettent de combiner des relations bien fondées de manière très pratique. Nous les utilisons notamment à la section 7.2.3.

Étant donné un ensemble A, un *multiensemble* sur A est une fonction $\mathcal{M} : A \to \mathbb{N}$ telle que l'ensemble $\{a \in A \mid \mathcal{M}(a) \neq 0\}$ est fini. Si $\mathcal{M}_1$ et $\mathcal{M}_2$ sont deux multiensembles sur A, alors $\mathcal{M}_1 + \mathcal{M}_2$ est le multiensemble sur A tel que

$$\forall a \in A. \qquad (\mathcal{M}_1 + \mathcal{M}_2)(a) \quad = \quad \mathcal{M}_1(a) + \mathcal{M}_2(a) \ .$$

Pour tout $a \in A$, on écrit $a \in \mathcal{M}$ lorsque $\mathcal{M}(a) > 0$. Il est pratique d'écrire les multiensembles entre accolades, comme les ensembles.

L'extension multiensemble d'une relation binaire R sur A est la plus petite relation R_{mul} telle que pour tous multiensembles $\mathcal{M}$ et $\mathcal{M}'$ sur A

$$\forall a \in A. \qquad \big(\forall b \in \mathcal{M}'. \quad b \, R \, a\big) \qquad \Longrightarrow \qquad \mathcal{M} + \mathcal{M}' \quad R_{\text{mul}} \quad \mathcal{M} + \{a\} \ .$$

Proposition 3 *La relation* R_{mul} *est bien fondée lorsque* R *est bien fondée.*

Mots

Nous utilisons à la section 1.1 des *mots* pour construire une algèbre libre particulière. Nous rappelons ici les notions dont nous avons besoin sur ce sujet.

Un mot sur un ensemble A est une fonction partielle $\omega : \mathbb{N} \to A$ de domaine fini telle que pour tout $i \in \mathbb{N}$, si $i+1 \in \mathcal{D}om(f)$ alors $i \in \mathcal{D}om(f)$. On désigne par A^* l'ensemble des mots sur A.

Un mot ω de domaine $\{0, \dots, n\}$ s'écrit linéairement sous la forme

$$\omega(0).\omega(1).\cdots.\omega(n)$$

Le mot vide sur A est la fonction $\varepsilon : \mathbb{N} \to A$ de domaine vide. On le note ε. Il est pratique, au niveau des notations, de confondre l'élément $a \in A$ avec la fonction ω de domaine $\{0\}$ telle que $\omega(0) = a$.

Si ω_1 et ω_2 sont deux mots de domaines respectifs $\{0, \dots, n\}$ et $\{0, \dots, m\}$, alors le mot $\omega_1.\omega_2$ est la fonction de domaine $\{0, \dots, n+m\}$ telle que

$$\forall i \in \mathcal{D}om(\omega_1.\omega_2). \qquad (\omega_1.\omega_2)(i) \quad = \quad \begin{cases} \omega_1(i) & \text{si } i \leq n \\ \omega_2(i-n) & \text{sinon} \end{cases}$$

Si B est un ensemble de mots sur A et si $a \in A$, alors $a.B$ désigne l'ensemble de mots $\{a.\omega \mid \omega \in B\}$.

Ordinaux dénombrables

Nous utilisons à la partie III des *ordinaux dénombrables*. Nous rappelons ici leur définition, en nous basant sur la présentation de [Gal91].

Nous avons besoin des notions suivantes.

— Un ensemble O est *dénombrable* si c'est l'ensemble vide ou s'il existe une fonction surjective $f : \mathbb{N} \to O$.

— Un ensemble ordonné $(O, \leq)$ est *bien ordonné* si pour tout sous-ensemble non vide X de O,

$$\exists x \in O. \quad \forall y \in X. \quad x \leq y \ .$$

— Un sous-ensemble $X \subseteq O$ est *strictement borné* si

$$\exists x \in O. \quad \forall y \in X. \quad y < x \ .$$

Un ensemble $(\mathfrak{O}, \leq)$ *satisfait les axiomes des ordinaux dénombrables* si

(i) $\mathfrak{O}$ est bien ordonné par $\leq$,

(ii) tout sous-ensemble strictement borné de $\mathfrak{O}$ est dénombrable,

(iii) tout sous-ensemble dénombrable de $\mathfrak{O}$ est strictement borné.

Les éléments de $\mathfrak{O}$ sont appelés *ordinaux dénombrables*. Nous distinguons l'ordinal 0, les ordinaux successeurs et les ordinaux limites.

— Il suit de (iii) que $\mathfrak{O}$ a un minimum, noté 0, qui est la borne supérieure de $\emptyset$.

— D'autre part, il suit de (iii) et de (i) que tout sous-ensemble dénombrable X de $\mathfrak{O}$ a une plus petite borne supérieure stricte dans $\mathfrak{O}$. En particulier, pour tout $\mathfrak{a} \in \mathfrak{O}$, la plus petite borne supérieure de $\{\mathfrak{a}\}$ dans $\mathfrak{O}$ est *le successeur* de $\mathfrak{a}$, qui est noté $\mathfrak{a} + 1$.
— Un *ordinal limite* est un ordinal $\mathfrak{a}$ de $\mathfrak{O}$ différent de 0 tel qu'il n'existe pas de $\mathfrak{b} \in \mathfrak{O}$ tel que $\mathfrak{b} = \mathfrak{a}+1$. Si $\mathsf{X} \subseteq \mathfrak{O}$ est dénombrable mais n'a pas d'élément maximal, alors la plus petite borne supérieure de X dans $\mathfrak{O}$ est l'ordinal limite $\bigvee \mathsf{X}$. De plus, pour tout ordinal limite $\mathfrak{a}$, on a $\mathfrak{a} = \bigvee\{\mathfrak{b} \mid \mathfrak{b} < \mathfrak{a}\}$.

Première partie

Réécriture conditionnelle et lambda-calcul

Chapitre 1

Structures de termes

Ce chapitre est consacré aux constructions de base sur les termes. Pour pouvoir manipuler convenablement les termes avec lieurs, en particulier les λ-termes, il faut pouvoir identifier les termes qui ne diffèrent que par renommage de leurs variables liées. Pour ce faire, les termes avec lieur sont définis comme les quotients de termes du premier ordre modulo α-équivalence. Nous utilisons une relation d'α-équivalence inspirée de celle de Krivine [Kri90], que nous notons $=_\alpha$. Cette relation a l'avantage d'être facilement axiomatisable.

Certaines classes d'α-équivalence ne contiennent que des termes sans variables liées. Ce sont alors des singletons pour $=_\alpha$, qui représentent des termes du premier ordre, mais n'en sont pas à proprement parler si on définit les termes du premiers ordre « concrètement » comme étant des arbres ou des mots. Cependant, ces quotients sont bien des algèbres libres et à ce titre peuvent être considérés comme des termes. Afin de pouvoir traiter ceci rigoureusement, nous définissons les termes du premier ordre d'une façon un peu plus abstraite que celle décrite dans les textes standards [MT92, BN98].

Une question importante, une fois définie une relation d'α-équivalence, est de pouvoir trouver un représentant d'une classe d'équivalence donnée. Pour ce faire, nous utilisons les termes de de Bruijn [Bar84]. Ces termes sont construits sur une signature particulière dans laquelle les variables sont encodées par des entiers. Ceci permet, tout en restant au premier ordre, de ne pas avoir de problème de renommage. Afin de pouvoir utiliser les termes de de Bruijn comme représentants des classes d'α-équivalence de Krivine, nous étudions les rapports entre $=_\alpha$ et la relation d'équivalence sur les termes du premier ordre qui identifie les termes ayant même représentation de de Bruijn. Nous montrons que ces deux relations coïncident pour les termes du premier ordre vérifiant la condition que dans un terme $\lambda x.t$, la variable x n'est pas liée dans t. Nous appelons cette condition la « convention de Barendregt locale », en référence à la convention 2.1.13 de [Bar84].

1.1 Termes du premier ordre

Nous rappelons ici certaines notions fondamentales concernant les termes du premier ordre. Nos termes sur Σ et $\mathcal{X}$ sont les éléments de Σ-algèbres libres sur $\mathcal{X}$. Leur structure provient uniquement de la liberté de l'algèbre, elle est indépendante de son implantation, c'est à dire de la nature de ses éléments : ce peuvent être indifféremment des arbres, des mots, ou bien quelque chose d'autre. Nous utilisons des notions classiques, présentées dans les livres [MT92, BN98] et les notes de cours [CJ96] (voir aussi [GTWW77]). Cependant,

notre définition des termes comme algèbre libre quelconque ne se trouve pas dans ces références.

Définition 1.1.1 (Algèbres) *Soit une famille d'ensembles dénombrables* $\Sigma = (\Sigma_n)_{n\in\mathbb{N}}$ *que nous appelons* signature.

- *Une* Σ-algèbre *est un couple* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *où* A *est un ensemble et* $\Sigma_{\mathcal{A}}$ *est un ensemble qui contient une fonction* $f_{\mathcal{A}} : A^n \to A$ *pour chaque* $f \in \Sigma_n$.
- *Étant données deux* Σ*-algèbres* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *et* $\mathcal{B} = (B, \Sigma_{\mathcal{B}})$*, un* morphisme de Σ-algèbres $h : \mathcal{A} \to \mathcal{B}$ *est une fonction* $h : A \to B$ *telle que pour tout* $f \in \Sigma_n$*, pour tout* $a_1, \ldots, a_n \in A$*, on ait* $h(f_{\mathcal{A}}(a_1, \ldots, a_n)) = f_{\mathcal{B}}(h(a_1), \ldots, h(a_n))$.
- *Une* Σ*-algèbre* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *est une* sous-algèbre *d'une* Σ*-algèbre* $\mathcal{B} = (B, \Sigma_{\mathcal{B}})$ *si la fonction identité* $\mathcal{I}d : \mathcal{A} \to \mathcal{A}$ *induit un morphisme d'algèbres de* $\mathcal{A}$ *dans* $\mathcal{B}$.
- *Étant donné un ensemble* $\mathcal{X}$ *disjoint de chacun des* Σ_n*, une* Σ*-algèbre* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *et une injection* $i : \mathcal{X} \to A$*, on dit que* $\mathcal{A}$ *est* libre sur $\mathcal{X}$ pour i *si pour toute* Σ*-algèbre* $\mathcal{B} = (B, \Sigma_{\mathcal{B}})$*, pour toute fonction* $j : \mathcal{X} \to B$*, il existe un unique morphisme* $\bar{j} : \mathcal{A} \to \mathcal{B}$ *tel que* $\bar{j} \circ i = j$ *(voir figure 1.1). Le morphisme* $\bar{j} : \mathcal{A} \to \mathcal{B}$ *est dit* canonique *pour* $(\mathcal{A}, \mathcal{B}, j)$.

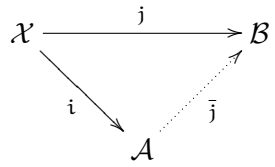


FIG. 1.1: $\bar{j} \circ i = j$

Notons que $\mathcal{A}$ est une sous algèbre de $\mathcal{B}$ si et seulement si $A \subseteq B$ et pour tout $f \in \Sigma_n$, pour tout $a_1, \ldots, a_n \in A$, on a $f_{\mathcal{A}}(a_1, \ldots, a_n) = f_{\mathcal{B}}(a_1, \ldots, a_n)$.

Lorsque le contexte le permet, nous désignons $\biguplus_{n\in\mathbb{N}} \Sigma_n$ par Σ. Nous disons que Σ est d'arité n si $\Sigma_m = \emptyset$ pour tout $m \neq n$. Les éléments de Σ sont notés avec la police sans sérif, par exemple $f, g, h, \ldots$ mais aussi pivot, filter, etc.

Remarque 1.1.2 *Il est aisé de voir que deux* Σ*-algèbres libres sur* $\mathcal{X}$ *sont isomorphes, et ce pour un unique isomorphisme. En effet, soient* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *et* $\mathcal{B} = (B, \Sigma_{\mathcal{B}})$ *deux* Σ*-algèbres libres sur* $\mathcal{X}$ *respectivement pour* $(\_)_{\mathcal{A}}$ *et* $(\_)_{\mathcal{B}}$*. Alors,* $\overline{(\_)}_{\mathcal{A}} : \mathcal{B} \to \mathcal{A}$ *et* $\overline{(\_)}_{\mathcal{B}} : \mathcal{A} \to \mathcal{B}$ *sont les uniques morphismes tels que*

$$\overline{(\_)}_{\mathcal{A}} \circ (\_)_{\mathcal{B}} = (\_)_{\mathcal{A}} \qquad \overline{(\_)}_{\mathcal{B}} \circ (\_)_{\mathcal{A}} = (\_)_{\mathcal{B}} .$$

Il s'en suit que

$$(\overline{(\_)}_{\mathcal{A}} \circ \overline{(\_)}_{\mathcal{B}}) \circ (\_)_{\mathcal{A}} = (\_)_{\mathcal{A}} \qquad (\overline{(\_)}_{\mathcal{B}} \circ \overline{(\_)}_{\mathcal{A}}) \circ (\_)_{\mathcal{B}} = (\_)_{\mathcal{B}} .$$

Or la liberté de $\mathcal{A}$ *et* $\mathcal{B}$ *implique que les identités* $\mathcal{I}\mathrm{d}_{\mathcal{A}} : \mathcal{A} \to \mathcal{A}$ *et* $\mathcal{I}\mathrm{d}_{\mathcal{B}} : \mathcal{B} \to \mathcal{B}$ *sont les uniques morphismes tels que*

$$\mathcal{I}\mathrm{d}_{\mathcal{A}} \circ (\_)_{\mathcal{A}} \quad = \quad (\_)_{\mathcal{A}} \qquad\qquad \mathcal{I}\mathrm{d}_{\mathcal{B}} \circ (\_)_{\mathcal{B}} \quad = \quad (\_)_{\mathcal{B}} \; ;$$

d'où

$$\overline{(\_)}_{\mathcal{A}} \circ \overline{(\_)}_{\mathcal{B}} \quad = \quad \mathcal{I}\mathrm{d}_{\mathcal{A}} \qquad\qquad \overline{(\_)}_{\mathcal{B}} \circ \overline{(\_)}_{\mathcal{A}} \quad = \quad \mathcal{I}\mathrm{d}_{\mathcal{B}} \; .$$

Les isomorphismes $\overline{(\_)}_{\mathcal{A}} : \mathcal{B} \to \mathcal{A}$ *et* $\overline{(\_)}_{\mathcal{B}} : \mathcal{A} \to \mathcal{B}$ *sont appelés respectivement* isomorphisme canonique de $\mathcal{B} \to \mathcal{A}$ *et* isomorphisme canonique de $\mathcal{A} \to \mathcal{B}$.

Toutes les algèbres (même lorsqu'elles ne sont pas libres), contiennent une notion canonique d'objets générés à partir des variables et par itération des fonctions de la signature. Lorsque l'algèbre n'est pas libre, ces objets ne peuvent pas être vus comme des termes. Nous les appelons donc « pré-termes ».

Définition 1.1.3 (Pré-termes d'une algèbre) *Soit un ensemble dénombrable* $\mathcal{X}$*, une* Σ*-algèbre* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *et une injection* $(\_)_{\mathcal{A}} : \mathcal{X} \to A$*. L'ensemble* $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *des* pré-termes sur $(\mathcal{A}, (\_)_{\mathcal{A}})$ *est le plus petit sous-ensemble de* A *tel que*

— *si* $x \in \mathcal{X}$ *alors* $x_{\mathcal{A}} \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$,

— *si* $f \in \Sigma_n$ *et* $t_1, \ldots, t_n \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *alors* $f_{\mathcal{A}}(t_1, \ldots, t_n) \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$.

Notons que $(\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}), \Sigma_{\mathcal{A}})$ est une Σ-algèbre.

Remarque 1.1.4 *La définition de* $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *fournit un principe d'induction sur les éléments de cet ensemble. En effet, on peut voir* $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *comme étant le plus petit ensemble satisfaisant aux règles suivantes :*

$$(\textsc{Var}) \; \frac{}{x_{\mathcal{A}} \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})} \; (x \in \mathcal{X})$$

$$(\textsc{Symb}) \; \frac{t_1 \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}}) \quad \ldots \quad t_n \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})}{f_{\mathcal{A}}(t_1, \ldots, t_n) \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})} \; (f \in \Sigma_n)$$

Alors, raisonner par induction sur $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *revient à raisonner par induction sur la hauteur des arbres de preuves construits par les règles* (Var) *et* (Symb)*. Notons que si* $\mathcal{A}$ *n'est pas libre, alors il peut y avoir plusieurs arbres de preuve pour* $t \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$*. Nous dirons « par induction sur* $t \in \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *» ou même « par induction sur* t *» au lieu de « par induction sur* $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *».*

Le lemme suivant est bien connu. Notre cas est cependant différent du cas habituel (voir par exemple le lemme 3.4.8 de [MT92]) car nos algèbres libres sont libres pour des injections $\mathcal{X} \to A$ qui ne sont pas nécessairement des inclusions.

Lemme 1.1.5 *Soit deux* Σ*-algèbres* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *et* $\mathcal{B} = (B, \Sigma_{\mathcal{B}})$*. Si* $(\_)_{\mathcal{A}} : \mathcal{X} \to A$ *est une injection et* $j_1, j_2 : \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}}) \to \mathcal{B}$ *sont deux morphismes tels que* $j_1(x_{\mathcal{A}}) = j_2(x_{\mathcal{A}})$ *pour tout* $x \in \mathcal{X}$*, alors* $j_1 = j_2$.

PREUVE. On montre que $j_1(t) = j_2(t)$ en raisonnant par induction sur les arbres de preuves de $t \in \mathcal{T}er(\Sigma_\mathcal{A}, \mathcal{X}_\mathcal{A})$. On raisonne par cas sur la dernière règle utilisée.

(VAR) Dans ce cas $t =_\mathcal{A} x_\mathcal{A}$ avec $x \in \mathcal{X}$, et par hypothèse on a $j_1(x_\mathcal{A}) = j_2(x_\mathcal{A})$.

(SYMB) Dans ce cas, $t =_\mathcal{A} f_\mathcal{A}(t_1, \ldots, t_n)$ avec $f \in \Sigma_n$. Par hypothèse d'induction on a $j_1(t_i) = j_2(t_i)$ pour tout $i \in \{1, \ldots, n\}$. On en conclut que $j_1(t) = j_2(t)$ car

$$\begin{array}{lll} & j_1(f_\mathcal{A}(t_1, \ldots, t_n)) & = \quad f_\mathcal{B}(j_1(t_1), \ldots, j_1(t_n)) \\ \text{et} & j_2(f_\mathcal{A}(t_1, \ldots, t_n)) & = \quad f_\mathcal{B}(j_2(t_1), \ldots, j_2(t_n)) \ . \end{array}$$

□

Le théorème d'existence d'algèbres libres est un résultat classique d'algèbre universelle. Exhiber une algèbre libre revient à décrire une représentation des termes, c'est-à-dire leur implantation. Il en existe plusieurs : ce sont des arbres dans [CJ96], mais ils peuvent être aussi des mots sur l'alphabet $\Sigma \cup \mathcal{X} \cup \{\text{`(`}, \text{`)`}, \text{`,`}\}$, comme dans [GTWW77]. Nous considérons le cas des arbres.

Théorème 1.1.6 *Soit une signature Σ et un ensemble $\mathcal{X}$ disjoint de Σ. Il existe une Σ-algèbre $\mathcal{A} = (A, \Sigma_\mathcal{A})$ et une injection $i : \mathcal{X} \to A$ telles que $\mathcal{A}$ est libre sur $\mathcal{X}$ pour i.*

Rappelons que $\mathbb{N}^*$ désigne l'ensemble des mots finis sur $\mathbb{N}$.

PREUVE. Soit T le plus petit ensemble de fonctions partielles de $\mathbb{N}^* \to \Sigma \cup \mathcal{X}$ tel que

— pour tout $x \in \mathcal{X}$ la fonction x_T de domaine $\{\varepsilon\}$ telle que $x_T(\varepsilon) = x$ est dans T ;

— pour tout $f \in \Sigma_n$, pour tout $t_1, \ldots, t_n \in T$, la fonction $f_T(t_1, \ldots, t_n)$ de domaine $\{\varepsilon\} \cup 1.\mathcal{D}om(t_1) \cup \cdots \cup n.\mathcal{D}om(t_n)$ telle que

$$f_T(t_1, \ldots, t_n)(\varepsilon) = f \qquad \text{et} \qquad f_T(t_1, \ldots, t_n)(i.p) = t_i(p)$$

est dans T.

On munit T d'une structure de Σ-algèbre en assignant à chaque $f \in \Sigma_n$ la fonction $f_T : t_1, \ldots, t_n \in T \mapsto f_T(t_1, \ldots, t_n)$.

Montrons que c'est une Σ-algèbre libre sur $\mathcal{X}$ pour $(\_)_T : x \in \mathcal{X} \mapsto x_T$. Soit $\mathcal{A} = (A, \Sigma_\mathcal{A})$ une Σ-algèbre et une fonction $j : \mathcal{X} \to A$. Alors, la fonction $\bar{j} : T \to A$ définie par induction sur T par $\bar{j}(x_T) =_{\text{def}} j(x)$ pour tout $x \in \mathcal{X}$ et $\bar{j}(f_T(t_1, \ldots, t_n)) =_{\text{def}} f_\mathcal{A}(\bar{j}(t_1), \ldots, \bar{j}(t_n))$ pour tout $f \in \Sigma_n$ est un morphisme de (T, Σ_T) dans $\mathcal{A}$ tel que $\bar{j} \circ (\_)_T = j$. D'autre part, il est aisé de voir que $T = \mathcal{T}er(\Sigma_T, \mathcal{X}_T)$. D'après le lemme 1.1.5, $\bar{j}$ est donc le seul morphisme de $(T, \Sigma_T) \to \mathcal{A}$ tel que $\bar{j} \circ (\_)_T = j$. □

La propriété suivante est fondamentale pour pouvoir considérer comme ensemble de termes n'importe quelle algèbre libre.

Proposition 1.1.7 (Séparation) *Si $\mathcal{A}$ est libre sur $\mathcal{X}$, alors*

— *pour tout $x \in \mathcal{X}$, pour tout $f \in \Sigma_n$ et $a_1, \ldots, a_n \in A$, on a $x_\mathcal{A} \neq_\mathcal{A} f_\mathcal{A}(a_1, \ldots, a_n)$,*

— *pour tout $f \in \Sigma_n$, tout $g \in \Sigma_m$, tout $a_1, \ldots, a_n \in A$ et tout $b_1, \ldots, b_m \in A$, si $f_\mathcal{A}(a_1, \ldots, a_n) =_\mathcal{A} g_\mathcal{A}(b_1, \ldots, b_m)$ alors on a $n = m$, $f = g$ et $a_i = b_i$ pour tout $i \in \{1, \ldots, n\}$.*

PREUVE. Tout d'abord il est clair que cette propriété est satisfaite pour l'algèbre libre d'arbres (T, Σ_T) définie dans la preuve du théorème 1.1.6.

Soit maintenant une Σ-algèbre $\mathcal{A} = (A, \Sigma_\mathcal{A})$ libre sur $\mathcal{X}$ pour $(\_)_\mathcal{A}$. Il existe donc un unique morphisme $j : \mathcal{A} \to (T, \Sigma_T)$ tel que $j \circ (\_)_\mathcal{A} = (\_)_T$. La liberté de (T, Σ_T) et $\mathcal{A}$ implique que c'est un isomorphisme, en particulier qu'il est injectif. Il s'en suit :

— Pour tout $x \in \mathcal{X}$, pour tout $f \in \Sigma_n$, pour tout $a_1, \ldots, a_n \in A$, on a $x_T \neq f_T(j(a_1), \ldots, j(a_n))$. Or $j(x_\mathcal{A}) = x_T$ et $j(f_\mathcal{A}(a_1, \ldots, a_n)) = f_T(j(a_1), \ldots, j(a_n))$, donc $x_\mathcal{A} \neq f_\mathcal{A}(a_1, \ldots, a_n)$.

— Pour tout $f \in \Sigma_n$, $g \in \Sigma_m$, pour tout $a_1, \ldots, a_n \in A$, pour tout $b_1, \ldots, b_m \in A$, si $f_\mathcal{A}(a_1, \ldots, a_n) = g_\mathcal{A}(b_1, \ldots, b_m)$, alors $f_T(j(a_1), \ldots, j(a_n)) = g_T(j(b_1), \ldots, j(b_m))$. On a donc $f = g$, $m = n$ et $j(a_i) = j(b_i)$ pour tout $i \in \{1, \ldots, n\}$. Comme j est injectif, on en déduit que $a_i = b_i$ pour tout $i \in \{1, \ldots, n\}$. □

Lemme 1.1.8 *Soient une signature Σ et un ensemble $\mathcal{X}$ disjoint de Σ. Si $\mathcal{A} = (A, \Sigma_\mathcal{A})$ est une Σ-algèbre libre sur $\mathcal{X}$ pour l'injection $(\_)_\mathcal{A} : \mathcal{X} \to A$ alors,*

(i) l'algèbre $(\mathcal{T}er(\Sigma_\mathcal{A}, \mathcal{X}_\mathcal{A}), \Sigma_\mathcal{A})$ est libre sur $\mathcal{X}$ pour $(\_)_\mathcal{A}$,

(ii) $A = \mathcal{T}er(\Sigma_\mathcal{A}, \mathcal{X}_\mathcal{A})$.

PREUVE. Soit $T =_{\text{def}} \mathcal{T}er(\Sigma_\mathcal{A}, \mathcal{X}_\mathcal{A})$ et $\mathcal{T} =_{\text{def}} (T, \Sigma_\mathcal{A})$.

(i) On vérifie que $\mathcal{T}$ est bien une Σ-algèbre libre sur $\mathcal{X}$ pour $(\_)_\mathcal{A}$. Pour toute Σ-algèbre $\mathcal{B} = (B, \Sigma_\mathcal{B})$ avec $j : \mathcal{X} \to B$, on étend $(\_)_\mathcal{A}$ en un morphisme $\bar{j} : \mathcal{T} \to \mathcal{B}$, en posant par induction $\bar{j}(x_\mathcal{A}) =_{\text{def}} j(x)$ et $\bar{j}(f_\mathcal{A}(a_1, \ldots, a_n)) =_{\text{def}} f_\mathcal{B}(\bar{j}(a_1), \ldots, \bar{j}(a_n))$. Cette définition est correcte d'après la remarque 1.1.4 et la proposition 1.1.7 (rappelons que $T \subseteq A$). De plus, cette extension est unique d'après le lemme 1.1.5.

(ii) On sait déjà que $T \subseteq A$, montrons que $A \subseteq T$.
Comme $\mathcal{A}$ est libre sur $\mathcal{X}$, il existe $i : \mathcal{A} \to \mathcal{T}$ tel que $i \circ (\_)_\mathcal{A} = (\_)_\mathcal{A}$. De plus, $\mathcal{T}$ étant une sous-algèbre de $\mathcal{A}$, l'identité $\mathcal{I}d_\mathcal{T}$ est un morphisme de $\mathcal{T} \to \mathcal{A}$. On a donc $(\mathcal{I}d_\mathcal{T} \circ i) \circ (\_)_\mathcal{A} = (\_)_\mathcal{A}$. Or, la liberté de $\mathcal{A}$ implique que $\mathcal{I}d_\mathcal{A}$ est le seul morphisme tel que $\mathcal{I}d_\mathcal{A} \circ (\_)_\mathcal{A} = (\_)_\mathcal{A}$. Il s'en suit que $\mathcal{I}d_\mathcal{T} \circ i = \mathcal{I}d_\mathcal{A}$.
Donc, pour tout $a \in A$, on a $i(a) = (\mathcal{I}d_\mathcal{T} \circ i)(a) = \mathcal{I}d_\mathcal{A}(a) = a$, soit $a \in T$ car $i(a) \in T$. □

Il s'en suit que nous pouvons utiliser comme termes du premier ordre les pré-termes d'une algèbre libre quelconque.

Définition 1.1.9 (Termes du premier ordre) *Étant donnés une signature Σ et un ensemble de variables $\mathcal{X}$ disjoint de Σ, un ensemble de termes sur $(\Sigma, \mathcal{X})$ est une Σ-algèbre libre sur $\mathcal{X}$.*

L'ensemble $V(t)$ *des* variables de $t \in \mathcal{T}er(\Sigma_\mathcal{A}, \mathcal{X}_\mathcal{A})$ *est défini inductivement comme suit :*

— $V(x_\mathcal{A}) =_{def} \{x\}$,

— $V(f_\mathcal{A}(t_1, \ldots, t_n)) =_{def} \bigcup_{1 \leq i \leq n} V(t_i)$.

Un terme t est dit clos *lorsque $V(t) = \emptyset$. Soit $\mathcal{T}er(\Sigma_\mathcal{A})$ l'ensemble des termes clos de $\mathcal{T}er(\Sigma_\mathcal{A}, \mathcal{X}_\mathcal{A})$.*

Remarque 1.1.10 *Les arbres de preuves de 1.1.4 peuvent être étiquetés de la manière suivante : chaque application de la règle* (VAR) *à la variable* x *est étiquetée par* $x_{\mathcal{A}}$ *et chaque application de la règle* (SYMB) *au symbole* f *est étiquetée par* $f_{\mathcal{A}}$*. Il est aisé de voir que l'ensemble d'arbres ainsi construit est isomorphe à* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *et peut être vu comme une* Σ*-algèbre libre sur* $\mathcal{X}$ *(une fois muni des bonnes opérations). Ainsi, nous avons pu donner une structure d'arbre aux éléments d'une algèbre libre quelconque. Ceci justifie l'idée qu'il est approprié de voir les termes comme des arbres, même s'il ne sont pas implantés comme tels.*

Lorsque le contexte le permet, on écrit $\mathcal{T}er(\Sigma, \mathcal{X})$ pour désigner un ensemble quelconque de termes sur $(\Sigma, \mathcal{X})$.

Remarque 1.1.11 (Grammaires de termes) *Nous écrivons* $\mathcal{X}$ *pour désigner un ensemble infini dénombrable dont les éléments peuvent être énumérés par* $(x_i)_{i\in\mathbb{N}}$*. Par convention, quand nous manipulons une signature* Σ*, nous supposons qu'elle est disjointe de* $\mathcal{X}$*, sauf mention du contraire.*

Nous écrivons

$$t \in T \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n)$$

où $x \in \mathcal{X}$ *et* $f \in \Sigma_n$*, pour désigner un ensemble* T *arbitrairement choisi parmi les ensembles des termes sur* $(\Sigma, \mathcal{X})$*.*

Nous rappelons ici quelques opérations classiques et fondamentales sur les termes du premier ordre. Nos notations sont celles de [DJ90]. La proposition 1.1.7 assure la correction de ces définitions pour les algèbres libres. Rappelons que les éléments des algèbres libres sont des termes *finis*.

Définition 1.1.12 (Positions, occurences et remplacements) *Soit* $t \in \mathcal{T}er(\Sigma, \mathcal{X})$*.*

(i) L'ensemble des positions *dans* t *est l'ensemble* $\mathcal{P}os(t)$ *de mots sur* $\mathbb{N}$ *défini inductivement comme suit :*

$$\begin{array}{lll} \mathcal{P}os(x) & =_{def} & \{\varepsilon\} \\ \mathcal{P}os(f(t_1, \ldots, t_n)) & =_{def} & \{\varepsilon\} \cup 1.\mathcal{P}os(t_1) \cup \cdots \cup n.\mathcal{P}os(t_n) \quad \textit{si } f \in \Sigma_n \ . \end{array}$$

(ii) Étant donné $p \in \mathcal{P}os(t)$*,* le sous-terme *de* t *à la position* p*, noté* $t|_p$*, est défini inductivement comme suit :*

$$\begin{array}{lll} t|_\varepsilon & =_{def} & t \\ f(t_1, \ldots, t_n)|_{i.p} & =_{def} & t_i|_p \quad \textit{si } f \in \Sigma_n \ . \end{array}$$

Si il existe $p \in \mathcal{P}os(t)$ *tel que* $t|_p = u$*, nous dirons que* u a une occurrence dans t.

(iii) Le remplacement *dans* t *de* $t|_p$ *par un terme* u *est le terme* $t[u]_p$ *défini inductivement comme suit :*

$$\begin{array}{lll} t[u]_\varepsilon & =_{def} & u \\ f(t_1, \ldots, t_n)[u]_{i.p} & =_{def} & f(t_1, \ldots, t_i[u]_p, \ldots, t_n) \quad \textit{si } f \in \Sigma_n \ . \end{array}$$

(iv) Enfin, l'ensemble des occurrences *d'un terme* u *dans* t *est le plus petit ensemble* $\mathcal{O}cc(t,u)$ *tel que* $\mathcal{O}cc(t,u) =_{def} \{\varepsilon\}$ *si* $t = u$*, et sinon,*

$$\mathcal{O}cc(f(t_1,\ldots,t_n),u) \;=_{def}\; \bigcup_{1\leq i\leq n} i.\mathcal{O}cc(t_i,u) \quad \textit{si } f \in \Sigma_n\ .$$

Dans certaines situations, il est plus lisible de noter $t[u]_p$ par $t[p \leftarrow u]$.

Remarque 1.1.13 *Toutes ces constructions sont indépendantes du choix d'une* Σ*-algèbre particulière. Ainsi, si* $\mathcal{T}er(\Sigma_\mathcal{A},\mathcal{X}_\mathcal{A})$ *et* $\mathcal{T}er(\Sigma_\mathcal{B},\mathcal{X}_\mathcal{B})$ *sont deux ensembles de termes sur* $(\Sigma,\mathcal{X})$*, et si* $i : \mathcal{A} \to \mathcal{B}$ *est un isomorphisme, alors pour tout* $t,u \in \mathcal{T}er(\Sigma_\mathcal{A},\mathcal{X}_\mathcal{A})$ *et tout* $p \in \mathcal{P}os(t)$ *on a* $V(t) = V(i(t))$*,* $\mathcal{P}os(t) = \mathcal{P}os(i(t))$*,* $i(t|_p) = i(t)|_p$*,* $i(t[u]_p) = i(t)[i(u)]_p$*, et* $\mathcal{O}cc(t,u) = \mathcal{O}cc(i(t),i(u))$*.*

Rappelons qu'une valuation est une fonction de domaine fini.

Définition 1.1.14 (Substitutions du premier ordre) *Soit une valuation* $\sigma : \mathcal{X} \to \mathcal{T}er(\Sigma_\mathcal{A},\mathcal{X}_\mathcal{A})$*. On l'étend en une* substitution du premier ordre $\overline{\sigma} : \mathcal{T}er(\Sigma_\mathcal{A},\mathcal{X}_\mathcal{A}) \to \mathcal{T}er(\Sigma_\mathcal{A},\mathcal{X}_\mathcal{A})$ *définie inductivement comme suit :*

$$\begin{array}{lll}
\overline{\sigma}(x_\mathcal{A}) & =_{def}\ \sigma(x) & \textit{si } x \in \mathcal{D}om(\sigma)\\
\overline{\sigma}(x_\mathcal{A}) & =_{def}\ x_\mathcal{A} & \textit{sinon}\\
\overline{\sigma}(f_\mathcal{A}(t_1,\ldots,t_n)) & =_{def}\ f_\mathcal{A}(\overline{\sigma}(t_1),\ldots,\overline{\sigma}(t_n)) & \textit{si } f \in \Sigma_n\ .
\end{array}$$

Remarque 1.1.15 *La liberté de* $\mathcal{T}er(\Sigma_\mathcal{A},\mathcal{X}_\mathcal{A})$ *sur* $\mathcal{X}$ *implique que* $\overline{\sigma}$ *est en fait l'unique morphisme de* $\mathcal{A} \to \mathcal{A}$ *tel que* $\overline{\sigma} \circ (\_)_\mathcal{A} = \sigma$*.*

Lorsque le contexte le permet, nous écrivons σ pour désigner la substitution issue de la valuation σ. Si σ est la valuation qui associe t_i à x_i pour $i \in \{1,\ldots,n\}$, alors $[\vec{x} \mapsto \vec{t}]$ désigne la substitution issue de σ et $t[\vec{x} \mapsto \vec{t}]$ désigne son application au terme t. L'ensemble $\{V(\sigma(x)) \mid x \in \mathcal{D}om(\sigma)\}$ des variables d'une substitution σ est noté $V(\sigma)$.

Proposition 1.1.16 *Soient un terme* $t \in \mathcal{T}er(\Sigma,\mathcal{X})$ *et deux substitutions* σ*,* σ' *telles que* $V(t) \cap \mathcal{D}om(\sigma) = V(t) \cap \mathcal{D}om(\sigma')$*. Si* $\sigma(x) = \sigma'(x)$ *pour tout* $x \in V(t) \cap \mathcal{D}om(\sigma)$*, alors* $\sigma(t) = \sigma'(t)$*.*

Preuve. Par induction sur t.

- $t = x \in \mathcal{X}$. Comme $x \in V(t)$, si $x \in \mathcal{D}om(\sigma)$, par hypothèse on a $x \in \mathcal{D}om(\sigma')$, donc $\sigma(t) = \sigma(x) = \sigma'(x) = \sigma'(t)$. Sinon, par hypothèse $x \notin \mathcal{D}om(\sigma')$ donc $\sigma(t) = t = \sigma'(t)$.
- $t = f(t_1,\ldots,t_n)$. Comme $V(t_i) \subseteq V(t)$, on a $V(t_i) \cap \mathcal{D}om(\sigma) = V(t_i) \cap \mathcal{D}om(\sigma')$ pour tout $i \in \{1,\ldots,n\}$, donc par hypothèse d'induction $\sigma(t_i) = \sigma'(t_i)$, d'où $\sigma(t) = \sigma'(t)$. □

En particulier, si $V(t) \cap \mathcal{D}om(\sigma) = \emptyset$, alors on a $\sigma(t) = t$. En effet, sur $V(t)$, σ coïncide avec la substitution de domaine vide.

Si σ et θ sont deux substitutions, alors $\theta \circ \sigma$ est la substitution définie par la valuation $\theta \circ \sigma(x) = \theta(\sigma(x))$, et dont le domaine est $\mathcal{D}om(\sigma)$. Il est bien connu que la composition des valuations ne se retrouve pas directement au niveau des substitution. En effet, avec $\sigma =_{\text{def}} [x \mapsto y]$ et $\theta =_{\text{def}} [y \mapsto z]$, on a $(\theta \circ \sigma)(f(x,y)) = f(z,y)$, alors que $\theta(\sigma(f(x,y))) = f(z,z)$. On a donc besoin d'hypothèses sur le domaine et les variables des substitutions.

Lemme 1.1.17 (Composition des substitutions du premier ordre) *Soit un terme* $t \in \mathcal{T}er(\Sigma, \mathcal{X})$ *et deux substitutions* σ, θ *telles que* $\mathcal{D}om(\sigma) \cap (\mathcal{D}om(\theta) \cup \mathcal{V}(\theta)) = \emptyset$*. Alors,*

$$\theta(\sigma(t)) \quad = \quad \theta \circ \sigma(\theta(t)) \ .$$

Preuve. Par induction sur t.

$t = x \in \mathcal{X}$. Si $x \in \mathcal{D}om(\theta)$, alors $x \notin \mathcal{D}om(\sigma)$ par hypothèse, donc $\theta(\sigma(x)) = \theta(x)$. Comme le domaine de σ est disjoint des variables de θ, on a $\theta \circ \sigma(\theta(x)) = \theta(x)$.
Sinon, supposons que $x \in \mathcal{D}om(\sigma)$. Alors, $\theta(\sigma(x)) = \theta \circ \sigma(x)$, et on a le résultat recherché car $\theta(x) = x$.
Enfin, si $x \notin \mathcal{D}om(\sigma) \cup \mathcal{D}om(\theta)$, alors $\theta(\sigma(x)) = x = \theta \circ \sigma(\theta(x))$.

$t = f(t_1, \ldots, t_n)$. Par hypothèse d'induction, pour tout $i \in \{1, \ldots, n\}$ on a $\theta(\sigma(t_i)) = \theta \circ \sigma(\theta(t_i))$, d'où le résultat recherché. □

Proposition 1.1.18 *Pour tout terme* $t \in \mathcal{T}er(\Sigma, \mathcal{X})$*, et toutes variables* $x, y, z \in \mathcal{X}$*, si* $y \notin \mathcal{V}(t)$ *alors*

$$t[x \mapsto z] \quad = \quad t[x \mapsto y][y \mapsto z] \ .$$

Preuve. Par induction sur t.

$t \in \mathcal{X}$. Si $t = x$, alors $t[x \mapsto z] = z$ et $t[x \mapsto y][y \mapsto z] = y[y \mapsto z] = z$.
Sinon, $t[x \mapsto z] = t$. Or $t[x \mapsto y][y \mapsto z] = t[y \mapsto z]$, donc $t[x \mapsto y][y \mapsto z] = t$ car $y \notin \mathcal{V}(t)$.

$t = f(t_1, \ldots, t_n)$. Pour tout $i \in \{1, \ldots, n\}$, $y \notin \mathcal{V}(t_i)$ donc par hypothèse d'induction, $t_i[x \mapsto z] = t_i[x \mapsto y][y \mapsto z]$, d'où le résultat. □

Pour les termes du premier ordre, les contextes sont des termes « à trou », contenant exactement une occurrence d'une variable notée [ ].

Définition 1.1.19 (Contextes) Un contexte sur $L \subseteq \mathcal{T}er(\Sigma_{App})$ *est un terme* $C \in L$ *contenant exactement une occurrence de la variable* [ ]*. On pose* $C[t] =_{def} C[t/[\,]]$ *pour tout* $t \in L$*.*

Si R *est une relation binaire sur* $L \subseteq \mathcal{T}er(\Sigma_{App}, \mathcal{X})$*, et* $C[\,], D[\,]$ *sont deux* L*-contextes alors on note* $C[\,]\,R\,D[\,]$ *si* $C\,R\,D$*. Dans le cas où* $C[\,]\,F\,D[\,]$ *pour une fonction* $F : L \to L$*, on pose* $F(C[\,]) =_{def} D[\,]$*.*

Toutes les constructions présentées ici s'étendent aux algèbres multisortées.

Définition 1.1.20 (Algèbres multisortées) *Une* signature multisortée *est la donnée*
— *d'un ensemble* $\mathcal{S}$ *de sortes,*
— *d'une famille d'ensembles* $(\Sigma_{s_1,\ldots,s_n,s})_{s_1,\ldots,s_n,s \in \mathcal{S}^{n+1}}$.
Une $(\mathcal{S}, \Sigma)$-algèbre $\mathcal{A}$ *est composée*
— *d'une famille d'ensembles disjoints* $(A_s)_{s \in \mathcal{S}}$*,*
— *pour tout* $s_1, \ldots, s_n, s \in \mathcal{S}^{n+1}$*, d'une fonction* $f_{\mathcal{A}} : A_{s_1} \times \cdots \times A_{s_n} \to A_s$*.*

Lorsque le contexte le permet, on désigne $f \in \Sigma_{s_1,\ldots,s_n,s}$ par $f : s_1 \times \cdots \times s_n \to s$ et on écrit $f : s$ si $n = 0$.

1.2 Termes du lambda-calcul et alpha-conversion

Nous introduisons ici les termes du λ-calcul, aussi appelés λ-termes. La syntaxe du λ-calcul contient un lieur de variable. Afin d'avoir une notion de substitution qui permette de définir la β-réduction, nous devons identifier les λ-termes qui ne diffèrent que par le nom de leur variables liées. Cette identification est axiomatisée par la relation d'α-conversion. L'ensemble des λ-termes est donc un ensemble de termes du premier ordre quotienté par l'α-conversion.

Définition 1.2.1 (Lambda-termes bruts) *Étant donnés une signature Σ et un ensemble $\mathcal{X}$ infini dénombrable et disjoint de Σ, les λ-termes bruts sont les éléments de l'ensemble $\mathcal{L}(\Sigma)$ décrit par la grammaire suivante :*

$$t, u \in \mathcal{L}(\Sigma) \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n) \quad | \quad t \cdot u \quad | \quad \lambda x.t$$

où $f \in \Sigma_n$ et $x \in \mathcal{X}$.

L'ensemble $\mathcal{L}(\Sigma)$ des λ-termes bruts est donc un ensemble de termes sur la signature $\Sigma \cup \{\_ \cdot \_\} \cup \{\lambda x.\_ \mid x \in \mathcal{X}\}$. De ce fait, le symbole d'application $\_ \cdot \_$ est un symbole binaire du premier ordre, et on pourrait le considérer comme un symbole de la signature. Ce n'est pas ce que nous avons choisi de faire dans ce qui suit : il nous semble plus intuitif de traiter le cas de $\_ \cdot \_$ séparément des symboles de Σ.

Précisons quelques conventions d'écriture et de vocabulaire. Nous désignons $\mathcal{L}(\emptyset)$ par $\mathcal{L}$ et les termes de $\mathcal{L}$ sont dits *purs*. Le symbole $\_ \cdot \_$ d'application est souvent implicite : au lieu de $t \cdot u$, nous écrivons $t\ u$ ou même tu lorsque le contexte le permet. Nous adoptons la convention usuelle que le parenthésage de l'application associe à gauche : $(tu)v$ est noté tuv. Par $\vec{t}$ nous désignons une liste de termes $t_1, \ldots, t_n$ de longueur $|\vec{t}| =_{\text{def}} n$, avec $n \in \mathbb{N}$.

Remarque 1.2.2 *Les λ-termes bruts étant des termes du premier ordre sur la signature $\overline{\Sigma}$, ils sont munis de la substitution du premier ordre. Mais cette opération ne voit pas l'abstraction comme un lieur, et de ce fait ne se préoccupe pas d'éventuelles captures de variables. Par exemple, on a $(\lambda x.y)[y \mapsto x] = \lambda x.x$.*

Pour que l'abstraction soit considérée comme un lieur, il faut pouvoir identifier les termes qui ne diffèrent que par renommage de leurs variables liées, comme par exemple $\lambda x.x$ et $\lambda y.y$.

Commençons par définir ce que sont les variables libres et liées d'un terme.

Définition 1.2.3 (Variables libres, variables liées) *Les ensembles des variables libres et liées d'un terme t, respectivement notés $\mathsf{FV}(t)$ et $\mathsf{BV}(t)$, sont définis inductivement comme suit :*

$$\begin{array}{lll}
\textit{Si } t = x \in \mathcal{X} & \textit{alors } \mathsf{FV}(t) = \{x\} & \textit{et } \mathsf{BV}(t) = \emptyset\ , \\
\textit{Si } t = f(t_1, \ldots, t_n) & \textit{alors } \mathsf{FV}(t) = \bigcup_{1 \le i \le n} \mathsf{FV}(t_i) & \textit{et } \mathsf{BV}(t) = \bigcup_{1 \le i \le n} \mathsf{BV}(t_i)\ , \\
\textit{Si } t = t_1\ t_2 & \textit{alors } \mathsf{FV}(t) = \mathsf{FV}(t_1) \cup \mathsf{FV}(t_2) & \textit{et } \mathsf{BV}(t) = \mathsf{BV}(t_1) \cup \mathsf{BV}(t_2)\ , \\
\textit{Si } t = \lambda x.t_1 & \textit{alors } \mathsf{FV}(t) = \mathsf{FV}(t_1) \setminus \{x\} & \textit{et } \mathsf{BV}(t) = \mathsf{BV}(t_1) \cup \{x\}\ .
\end{array}$$

Ces notions sont étendues aux valuations $\sigma : \mathcal{X} \to \mathcal{L}(\Sigma)$ de la façon suivante. On pose

$$\begin{array}{lll} \mathsf{FV}(\sigma) & =_{\text{def}} & \bigcup \{\mathsf{FV}(\sigma(x)) \mid x \in \mathcal{D}\mathrm{om}(\sigma)\} \ , \\ \mathsf{BV}(\sigma) & =_{\text{def}} & \bigcup \{\mathsf{BV}(\sigma(x)) \mid x \in \mathcal{D}\mathrm{om}(\sigma)\} \ . \end{array}$$

Notons que $\mathsf{FV}(\sigma)$ et $\mathsf{BV}(\sigma)$ ne sont pas nécessairement des ensembles disjoints.

Remarque 1.2.4 *Il est important de noter que pour tout* $t \in \mathcal{L}(\Sigma)$*, les variables de* t *sont toutes soit des variables libres, soit des variables liées :* $\mathsf{V}(t) \subseteq \mathsf{FV}(t) \cup \mathsf{BV}(t)$*. De plus,* $\mathsf{FV}(t) \subseteq \mathsf{V}(t)$*, mais en général* $\mathsf{BV}(t) \not\subseteq \mathsf{V}(t)$ *: la variable* x *est liée dans le terme* $\lambda x.t$ *même si elle n'apparaît pas dans* t*.*

Nous pouvons maintenant définir l'α-conversion. Notre définition est inspirée de celle de Krivine [Kri90]. Elle se fait par induction sur la taille des termes, que l'on définit comme suit.

Définition 1.2.5 (Taille d'un terme) *La taille d'un terme* $t \in \mathcal{L}(\Sigma)$*, notée* $|t|$ *est définie inductivement comme suit :*

$$\begin{array}{lll} |x| & =_{def} & 1 \qquad\qquad\qquad\qquad\quad \textit{si } x \in \mathcal{X} \\ |f(t_1, \ldots, t_n)| & =_{def} & 1 + |t_1| + \cdots + |t_n| \quad \textit{si } f \in \Sigma_n \\ |t_1\ t_2| & =_{def} & 1 + |t_1| + |t_2| \\ |\lambda x.t_1| & =_{def} & 1 + |t_1| \end{array}$$

Notons que si z est une variable, alors $|t[x \mapsto z]| = |t|$.

Définition 1.2.6 (Alpha-conversion) *La relation binaire d'*α*-conversion, notée* $=_\alpha$*, est définie par induction sur la taille des termes comme suit :*

$$\begin{array}{rcll} x & =_\alpha & x \ , & \\ f(\vec{t}) & =_\alpha & f(\vec{u}) & \textit{si } \vec{t} =_\alpha \vec{u} \ , \\ t_1 t_2 & =_\alpha & u_1 u_2 & \textit{si } t_1 =_\alpha u_1 \textit{ et } t_2 =_\alpha u_2 \ , \\ \lambda x.t & =_\alpha & \lambda y.u & \textit{si } t[x \mapsto z] =_\alpha u[y \mapsto z] \textit{ pour tout } z \in \mathcal{X} \textit{ sauf un nombre fini} \ . \end{array}$$

Il est aisé de voir que $=_\alpha$ est une relation d'équivalence sur $\mathcal{L}(\Sigma)$. De plus, pour tout $t, u \in \mathcal{L}(\Sigma)$, si $t =_\alpha u$, alors on a $\mathcal{P}os(t) = \mathcal{P}os(u)$.

Remarque 1.2.7 *Notre relation d'*α*-conversion est directement inspirée de celle de Krivine [Kri90]. Elle est cependant différente, du fait de sa notion de subsitution du premier ordre. Dans notre cas, la substitution du premier ordre voit les abstractions* λx *comme des symobles algébriques* $\lambda x.\_$ *et on a* $(\lambda x.t)[x \mapsto u] = \lambda x.(t[x \mapsto u])$*, alors que dans [Kri90], on a* $(\lambda x.t)[x \mapsto u] = \lambda x.t$*.*

Exemple 1.2.8 (Termes alpha-convertibles)

(i) On a $\lambda x.x =_\alpha \lambda y.y$ *car pour tout* $z \in \mathcal{X}$*, on a*

$$x[x \mapsto z] \quad = \quad z \quad = \quad y[y \mapsto z] \ .$$

(ii) *Si* $z \neq x, y$, *alors on a* $\lambda x.z =_\alpha \lambda y.z$ *: pour tout* $z' \in \mathcal{X}$,

$$z[x \mapsto z'] \quad = \quad z \quad = \quad z[y \mapsto z'] \ .$$

(iii) *On a* $\lambda x.\lambda x.x =_\alpha \lambda x.\lambda y.x$. *En effet, si* $y \neq x$ *alors pour tout* $z \in \mathcal{X} \setminus \{x, y\}$ *on a*

$$(\lambda x.x)[x \mapsto z] \quad = \quad \lambda x.z \quad =_\alpha \quad \lambda y.z \quad = \quad (\lambda y.x)[x \mapsto z] \ .$$

Ceci est en désaccord avec l'exemple d'α*-conversion donné dans la définition 2.1.11 de [Bar84], ainsi qu'avec celle de [Kri90] (voir aussi la remarque 1.2.7). Nous reviendrons sur ce sujet dans les sections 1.5 et 1.6.*

Voici quelques propriétés importantes de $=_\alpha$. Elles sont prouvées dans [Kri90] dans les termes purs et s'étendent aisément à notre cas.

Proposition 1.2.9 (Proposition 2, p. 6 dans [Kri90]) *Soient* t, t' *et* $\vec{u}$ *des termes, et* $\vec{x}$ *une séquence de variables distinctes. Si* $t =_\alpha t'$ *et aucune variable libre de* $\vec{u}$ *n'est liée dans* t, t', *alors* $t[\vec{x} \mapsto \vec{u}] =_\alpha t'[\vec{x} \mapsto \vec{u}]$.

Lemme 1.2.10 (Lemme 6, p. 7 dans [Kri90]) *Soit* $t \in \mathcal{L}(\Sigma)$ *et* Z *un ensemble fini de variables. Il existe un* $t' \in \mathcal{L}(\Sigma)$ α*-équivalent à* t *tel qu'aucune variable de* Z *n'est liée dans* t'.

La proposition 1.2.9 a pour conséquence que $=_\alpha$ passe au contexte. Le seul cas non trivial est celui de l'abstraction. Or, si $t =_\alpha t'$, alors $t[x \mapsto z] = t'[x \mapsto z]$ pour tout $z \in \mathcal{X} \setminus (\mathsf{FV}(t) \cup \mathsf{FV}(t'))$, donc $\lambda x.t =_\alpha \lambda x.t'$ car $\mathsf{FV}(t) \cup \mathsf{FV}(t')$ est fini.

Définition 1.2.11 (Lambda-termes) *L'ensemble des* λ*-termes sur* Σ *est* $\Lambda(\Sigma) =_{def} \mathcal{L}(\Sigma)/=_\alpha$.

De même que pour les λ-termes bruts, on écrit Λ au lieu de $\Lambda(\emptyset)$, et les λ-termes de Λ sont dits *purs*.

Pour tout $t \in \mathcal{L}(\Sigma)$, soit $[t] =_{\text{def}} \{t' \in \mathcal{L}(\Sigma) \mid t' =_\alpha t\}$. Par définition, chaque terme de $\Lambda(\Sigma)$ peut être écrit $[t]$ pour un $t \in \mathcal{L}(\Sigma)$. Cette notation est étendue aux valuations : $[\sigma]$ est l'ensemble des valuations σ' de même domaine que σ et telles que $\sigma(x) =_\alpha \sigma'(x)$ pour tout $x \in \mathcal{D}\mathrm{om}(\sigma)$. Pour les variables $x \in \mathcal{X}$ nous écrivons $x \in \Lambda(\Sigma)$ au lieu de $[x] \in \Lambda(\Sigma)$.

Lemme 1.2.12 (Structure des lambda-termes) *Définissons les opérations :*

— $\_ \cdot \_ : \Lambda(\Sigma) \times \Lambda(\Sigma) \to \Lambda(\Sigma)$ *telle que*

$$[t] \cdot [u] \quad = \quad \{u_1 \cdot u_2 \mid u_1 =_\alpha t_1 \ \wedge \ u_2 =_\alpha t_2\} \ ,$$

— *pour chaque* $f \in \Sigma_n$, $f(\_, \ldots, \_) : \Lambda(\Sigma) \times \cdots \times \Lambda(\Sigma) \to \Lambda(\Sigma)$ *telle que*

$$f([t_1], \ldots, [t_n]) \quad = \quad \{f(u_1, \ldots, u_n) \mid u_1 =_\alpha t_1 \ \wedge \ \ldots \ \wedge \ u_n =_\alpha t_n\} \ ,$$

— *pour chaque* $x \in \mathcal{X}$, $\lambda x.\_ : \Lambda(\Sigma) \to \Lambda(\Sigma)$ *telle que*

$$\lambda x.[t_1] = \{\lambda y.u_1 \mid u_1[y \mapsto z] =_\alpha t_1[x \mapsto z] \text{ pour tout } z \in \mathcal{X} \text{ sauf un nombre fini}\} \ .$$

On a alors :

(i) $[t_1 \cdot t_2] = [t_1] \cdot [t_2]$,

(ii) $[f(t_1, \ldots, t_n)] = f([t_1], \ldots, [t_n])$,

(iii) $[\lambda x.t_1] = \lambda x.[t_1]$.

PREUVE. Assurons nous que ces fonctions sont bien définies, c'est à dire qu'elles ne dépendent pas du choix des représentants des classes d'α-équivalence. C'est trivial pour les cas (i) et (ii) (transitivité de $=_\alpha$). Considérons le cas de $\lambda x.[t_1]$. Soit $t'_1 =_\alpha t_1$, montrons que $\lambda x.[t_1] = \lambda x.[t'_1]$. Si $\lambda y.u_1 \in \lambda x.[t_1]$, alors il existe un ensemble fini $Z \subseteq \mathcal{X}$ tel que pour toute variable z non dans Z, on ait $u_1[y \mapsto z] =_\alpha t_1[x \mapsto z]$. Par la proposition 1.2.9, pour tout z non dans $BV(t_1) \cup BV(t'_1)$, on a $t_1[x \mapsto z] =_\alpha t'_1[x \mapsto z]$. Il s'en suit que $u_1[y \mapsto z] =_\alpha t'_1[x \mapsto z]$ pour tout $z \notin Z \cup BV(t_1) \cup BV(t'_1)$, et donc que $\lambda y.u_1 \in [\lambda x.t'_1]$.

Le reste de la preuve est une conséquence directe de la définition 1.2.6. □

Remarque 1.2.13 *Il s'en suit que les termes de $\Lambda(\Sigma)$ peuvent être naturellement décrits par la grammaire suivante :*

$$t, u \in \Lambda(\Sigma) \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n) \quad | \quad t \cdot u \quad | \quad \lambda x.t$$

où $f \in \Sigma_n$ et $x \in \mathcal{X}$. Les sous-ensembles de $\Lambda(\Sigma)$ contenant les termes construits sans abstractions sont intéressants. Nous montrons dans le théorème 1.4.4 que ce sont des termes du premier ordre.

Notons que pour tout $u \in [t]$, on a $|u| = |t|$. De ce fait, on peut étendre la notion de taille de la définition 1.2.5 aux λ-termes modulo α-conversion.

Définition 1.2.14 (Taille d'un lambda-terme) *La taille d'un terme $[t] \in \Lambda(\Sigma)$, notée $|[t]|$ est définie par $|[t]| =_{def} |t|$.*

À cause de l'α-conversion, on ne peut, sur $\Lambda(\Sigma)$, définir la notion de sous-terme comme on l'a fait pour les termes du premier ordre en 1.1.12.(ii). Le problème vient du fait cette notion de sous-terme ne commute pas avec l'α-conversion.

Exemple 1.2.15 *Sur $\mathcal{L}(\Sigma)$, on a $\lambda x.x =_\alpha \lambda y.y$, mais $\lambda x.x|_1 = x \neq y = \lambda y.y|_1$.*

De se fait, un λ-terme $t \in \Lambda(\Sigma)$ peut avoir plusieurs sous-termes à la position $p \in \mathcal{P}os(t)$. Rappelons que $\mathcal{P}os(u) = \mathcal{P}os(t)$ pour tout $u \in [t]$.

Définition 1.2.16 (Positions et sous-termes d'un lambda-terme) *Soit $[t] \in \Lambda(\Sigma)$.*

(i) On définit l'ensemble des positions de $[t]$, *noté $\mathcal{P}os([t])$, par*

$$\mathcal{P}os([t]) \quad =_{def} \quad \mathcal{P}os(t) \ .$$

(ii) On définit l'ensemble des sous-termes de $[t]$ à la position $p \in \mathcal{P}os([t])$, *noté $[t]|_p$, par*

$$[t]|_p \quad =_{def} \quad \{[u|_p] \mid u \in [t]\} \ .$$

Remarque 1.2.17 *Si* $[u] \in [t]|_p$, *alors on peut avoir* $u \neq_\alpha t|_p$. *Par exemple,* $[y] \in [\lambda x.x]|_1$ *mais* $y \neq_\alpha x$. *Cependant, on a toujours* $|[u]| = |[t|_p]|$ *et* $\mathcal{P}os([u]) = \mathcal{P}os([t|_p])$.

On ne peut donc pas définir les λ-contextes sur $\Lambda(\Sigma)$ comme des termes de $\Lambda(\Sigma)$ à trous, comme on l'a fait en 1.1.19 pour $\mathcal{T}er(\Sigma, \mathcal{X})$. Par contre, chaque λ-contexte peut être représenté par un terme à trou de $\mathcal{L}(\Sigma)$.

Définition 1.2.18 (Lambda-contextes) *L'ensemble des* λ*-contextes est le plus petit ensemble de fonctions* $\Lambda(\Sigma) \to \Lambda(\Sigma)$ *qui contient l'identité* $[\,] : \Lambda(\Sigma) \to \Lambda(\Sigma)$ *et tel que si* $C[\,]$ *est un* λ*-contexte, alors*

- *pour tout* $x \in \mathcal{X}$, *la fonction* $C[\lambda x.[\,]] : u \mapsto C[\,](\lambda x.u)$ *en est un,*
- *pour tout* $t \in \Lambda(\Sigma)$, *la fonction* $C[[\,] \cdot t] : u \mapsto C[\,](u \cdot t)$ *en est un,*
- *pour tout* $t \in \Lambda(\Sigma)$, *la fonction* $C[t \cdot [\,]] : u \mapsto C[\,](t \cdot u)$ *en est un,*
- *pour tout* $f \in \Sigma_n$, *pour tout* $t_1, \ldots, t_n \in \Lambda(\Sigma)$, *pour tout* $i \in \{1, \ldots, n\}$, *la fonction*

$$C[f(t_1, \ldots, t_{i-1}, [\,], t_{i+1}, \ldots, t_n)] : \; u \;\mapsto\; C[\,](f(t_1, \ldots, t_{i-1}, u, t_{i+1}, \ldots, t_n))$$

en est un.

Si $C[\,]$ est un λ-contexte, on désigne $C[\,](t)$ par $C[t]$. Les λ-contextes sont aussi appelés « contextes ».

Définition 1.2.19 *Soit* $L \subseteq \Lambda(\Sigma)$. *Si* $C[\,] : \Lambda(\Sigma) \to \Lambda(\Sigma)$ *est un contexte tel que* $C[t] \in L$ *pour tout* $t \in L$, *alors c'est* un L-contexte, *noté* $C[\,] : L \to L$.

On définit maintenant la relation « sous-terme » dans $\Lambda(\Sigma)$.

Définition 1.2.20 *Étant donnés* $t, u \in \Lambda(\Sigma)$, *on dit que* u est un sous-terme de t *s'il existe un contexte* $C[\,]$ *différent de l'identité tel que* $t = C[u]$.

Remarque 1.2.21 *La relation sous-terme sur* $\Lambda(\Sigma)$ *ne préserve pas les classes d'*α*-équivalence. En effet,* x *et* y *sont des sous-termes de* $[\lambda x.x]$ *car* $[\lambda x.x] = [\lambda y.y]$.

1.3 Substitution sans capture

Nous allons maintenant définir la substitution sans capture. Notre définition diffère de celle de Krivine donnée dans son livre [Kri90]. En effet, cette dernière pose des difficultés si on a besoin de définir une relation de réécriture sur $\Lambda(\Sigma)$ comme étant le quotient d'une relation de réécriture sur $\mathcal{L}(\Sigma)$. C'est typiquement ce que nous faisons à la section 4.3 pour la réécriture conditionnelle puis au chapitre 8 pour traiter la préservation de la confluence par curryfication.

Cela nous amène à définir la substitution sans capture sur $\Lambda(\Sigma)$ à partir d'une notion de substitution sans capture qui est une *fonction partielle* sur $\mathcal{L}(\Sigma)$. Cette façon de faire semble originale.

Définition 1.3.1 (Substitution partielle sans capture) *Étant donnée une valuation* σ *sur* $\mathcal{L}(\Sigma)$*, la* substitution sans capture $\_\sigma$ *est la fonction partielle* $\mathcal{L}(\Sigma) \to \mathcal{L}(\Sigma)$ *définie comme suit :*

$$\begin{array}{lll} x\sigma & =_{def} \sigma(x) & \text{si } x \in \mathcal{D}om(\sigma) \\ x\sigma & =_{def} x & \text{sinon} \\ f(t_1,\dots,t_n)\sigma & =_{def} f(t_1\sigma,\dots,t_n\sigma) & \text{si } f \in \Sigma_n \\ (t_1\ t_2)\sigma & =_{def} t_1\sigma\ t_2\sigma & \\ (\lambda x.t_1)\sigma & =_{def} \lambda x.(t_1\sigma) & \text{si } x \notin \mathcal{D}om(\sigma) \cup FV(\sigma) \end{array}$$

C'est à cause de la dernière clause que la fonction $\_\sigma : \mathcal{L}(\Sigma) \to \mathcal{L}(\Sigma)$ n'est pas totale. En particulier, $t\sigma$ n'est pas définie si $BV(t) \cap \mathcal{D}om(\sigma) \neq \emptyset$. Par contre, si σ est la valuation de domaine vide, alors $t\sigma$ est défini.

Lorsque le contexte le permet, nous identifions une substitution $\_\sigma$ à sa valuation σ. On désigne par $FV(\sigma)$ l'ensemble $\{FV(\sigma(x)) \mid x \in \mathcal{D}om(\sigma)\}$ des *variables libres* de σ, et par $BV(\sigma)$ l'ensemble $\{BV(\sigma(x)) \mid x \in \mathcal{D}om(\sigma)\}$ l'ensemble de ses *variables liées*.

Lorsqu'elle est définie, la substitution partielle sans capture correspond à la substitution du premier ordre. Rappelons qu'une valuation est une fonction de domaine fini.

Proposition 1.3.2 *Pour tout* $t \in \mathcal{L}(\Sigma)$ *et toute valuation* $\sigma : \mathcal{X} \to \Lambda(\Sigma)$*, si* $t\sigma$ *est défini, alors* $t\sigma = \sigma(t)$*.*

Preuve. Par induction sur t.

- $t = x \in \mathcal{X}$. Si $x \in \mathcal{D}om(x)$, alors, $t\sigma = \sigma(x) = \sigma(t)$. Sinon, $t\sigma = t = \sigma(t)$.
- $t = f(\vec{t})$. Comme $\vec{t}\sigma$ est défini, on a $\vec{t}\sigma = \sigma(\vec{t})$ par hypothèse d'induction. Il s'en suit que $f(\vec{t})\sigma = f(\vec{t}\sigma) = f(\sigma(\vec{t})) = \sigma(f(\vec{t}))$.
- $t = t_1t_2$. Comme $t_1\sigma$ et $t_2\sigma$ sont définis, par hypothèse d'induction $t_1\sigma = \sigma(t_1)$ et $t_2\sigma = \sigma(t_2)$, donc $t_1t_2)\sigma = \sigma(t_1\theta_2)$.
- $t = \lambda x.t_1$. Comme $t\sigma$ est défini, on a $t\sigma = \lambda x.(t_1\sigma)$. Or, comme $t_1\sigma$ est défini, il suit de l'hypothèse d'induction que $t_1\sigma = \sigma(t_1)$, d'où $t\sigma = \sigma(\lambda x.t_1) = \sigma(t)$. □

Remarque 1.3.3

(i) Notons que $t\sigma$ *est défini si* $BV(t)$ *est disjoint de* $FV(\sigma)$ *et de* $\mathcal{D}om(\sigma)$*. C'est en particulier le cas lorsque* t *ne contient pas de variables liées.*

(ii) Grâce au lemme 1.2.10, pour tout $t \in \mathcal{L}(\Sigma)$*, pour toute substitution* σ*, il existe* $t' =_\alpha t$ *tel que* $t'\sigma$ *est défini.*

La proposition 1.3.2 nous permet d'étendre la proposition 1.1.16 aux substitutions sans capture.

Proposition 1.3.4 *Soient* $t \in \mathcal{L}(\Sigma)$ *et* σ*,* σ' *sont deux substitutions telles que* $t\sigma$ *et* $t\sigma'$ *soient définis. Si* $FV(t) \cap \mathcal{D}om(\sigma) = FV(t) \cap \mathcal{D}om(\sigma')$ *et* $\sigma(x) = \sigma'(x)$ *pour tout* $x \in FV(t) \cap \mathcal{D}om(\sigma)$ *alors on a* $t\sigma = t\sigma'$*.*

PREUVE. Tout d'abord, par la proposition 1.3.2, comme $t\sigma$ et $t\sigma'$ sont définis, on a $t\sigma = \sigma(t)$ et $t\sigma' = \sigma'(t)$.

De plus, comme $t\sigma$ et $t\sigma'$ sont définis, on a $BV(t) \cap \mathcal{D}om(\sigma) = BV(t) \cap \mathcal{D}om(\sigma') = \emptyset$. Il s'en suit que $FV(t) \cap \mathcal{D}om(\sigma) = V(t) \cap \mathcal{D}om(\sigma)$ et $FV(t) \cap \mathcal{D}om(\sigma) = V(t) \cap \mathcal{D}om(\sigma)$. Par la proposition 1.3.4 on a donc $\sigma(t) = \sigma'(t)$, d'où $t\sigma = t\sigma'$. □

En particulier, si $FV(t) \cap \mathcal{D}om(\sigma) = \emptyset$, alors on a $t\sigma = t$. En effet, sur $FV(t)$, σ coïncide avec la substitution de domaine vide.

Si σ est la valuation qui associe t_i à x_i pour $i \in \{1, \ldots, n\}$, alors $[t_1/x_1, \ldots, t_n/x_n]$ désigne la substitution issue de σ et $t[t_1/x_1, \ldots, t_n/x_n]$ désigne son application au terme t. De plus, $\sigma[u/x]$ désigne la substitution de domaine $\mathcal{D}om(\sigma) \cup \{x\}$ telle que $\sigma[u/x](x) = u$ et $\sigma[u/x](y) = \sigma(y)$ pour tout $y \in \mathcal{D}om(\sigma) \setminus \{x\}$.

Proposition 1.3.5 *Soit $t, u \in \mathcal{L}(\Sigma)$ et σ une substitution. Si $t(\sigma[u/x])$ est défini et $x \notin FV(\sigma) \cup \mathcal{D}om(\sigma)$, alors $(t\sigma)[u/x]$ est défini et $t(\sigma[u/x]) = (t\sigma)[u/x]$.*

PREUVE. Par induction sur t.

- $t = y \in \mathcal{X}$. Si $y = x$, alors $x(\sigma[u/x]) = u = x[u/x] = \sigma(x)[u/x]$ car $x \notin \mathcal{D}om(\sigma)$. Si $y \in \mathcal{D}om(\sigma)$ alors $y(\sigma[u/x]) = \sigma(y) = \sigma(y)[u/x]$ car $x \notin FV(\sigma)$.
- $t = t_1 t_2$ et $t = f(t_1, \ldots, t_n)$. Par hypothèse d'induction.
- $t = \lambda y.t_1$. Comme $t(\sigma[u/x])$ est défini, on a $y \notin (\mathcal{D}om(\sigma) \cup \{x\} \cup FV(\sigma) \cup FV(u))$ et $t(\sigma[u/x]) = t_1(\sigma[u/x])$. Par hypothèse d'induction $(t_1\sigma)[u/x]$ est défini et on a $t_1(\sigma[u/x]) = (t_1\sigma)[u/x]$. De plus, comme $y \notin \mathcal{D}om(\sigma) \cup FV(\sigma)$, $t\sigma = \lambda x.t_1\sigma$ et comme $y \notin \{x\} \cup FV(u)$ on a $(t\sigma)[u/x] = \lambda x.((t_1\sigma)[u/x])$. □

Étant données $R \subseteq \mathcal{L}(\Sigma) \times \mathcal{L}(\Sigma)$ et deux substitutions σ, σ', on note $\sigma R \sigma'$ si $\mathcal{D}om(\sigma) = \mathcal{D}om(\sigma')$ et $\sigma(x)\ R\ \sigma'(x)$ pour tout $x \in \mathcal{D}om(\sigma)$. Dans la proposition suivante, nous utilisons le fait que les valuations sont à domaine fini.

Proposition 1.3.6 *Soit un terme $t \in \mathcal{L}(\Sigma)$ et une substitution σ.*

(i) Si t' et t'' sont des termes α-équivalents à t tels que $t'\sigma$ et $t''\sigma$ sont définis, alors $t'\sigma$ est α-équivalent à $t''\sigma$.

(ii) Si $t\sigma$ est défini et $\sigma' =_\alpha \sigma$, alors $t\sigma'$ est défini et α-équivalent à $t\sigma$.

PREUVE.

(i) Par induction sur la taille de t. Soient t' et t'' deux termes α-équivalents à t tels que $t'\sigma$ et $t''\sigma$ soient définis.

- $t \in \mathcal{X}$. Dans ce cas, $[t] = \{t\}$, donc $t = t' = t''$, c.-à-d. $t'\sigma = t''\sigma$.
- $t = f(\vec{t})$. Par définition de $=_\alpha$, on a $t' = f(\vec{t'})$ et $t'' = f(\vec{t''})$ avec $\vec{t'}, \vec{t''}$ α-équivalents à $\vec{t}$. Par hypothèse d'induction, on a $\vec{t'}\sigma =_\alpha \vec{t''}\sigma$, d'où $f(\vec{t'})\sigma =_\alpha f(\vec{t''})\sigma$.
- $t = t_1 t_2$. Comme pour le cas précédent : Par définition de $=_\alpha$, on a $t' = t'_1 t'_2$ et $t'' = t''_1 t''_2$ avec t'_i, t''_i α-équivalents à t_i ($i \in \{1, 2\}$). Par hypothèse d'induction, on a $t'_i\sigma =_\alpha t''_i\sigma$, et il s'en suit que $(t'_1 t'_2)\sigma =_\alpha (t''_1 t''_2)\sigma$.

$t = \lambda x.t_1$. Par définition de $=_\alpha$, on a $t' = \lambda x'.t'_1$, $t'' = \lambda x''.t''_1$ et il existe trois ensembles finis $Z, Z', Z'' \subseteq \mathcal{X}$ tels que

$$\begin{array}{rrl} & t'_1[x' \mapsto z] =_\alpha t''_1[x'' \mapsto z] & \text{pour tout } z \in \mathcal{X} \setminus Z\ , \\ & t_1[x \mapsto z] =_\alpha t'_1[x' \mapsto z] & \text{pour tout } z \in \mathcal{X} \setminus Z'\ , \\ \text{et} & t''_1[x'' \mapsto z] =_\alpha t_1[x \mapsto z] & \text{pour tout } z \in \mathcal{X} \setminus Z''\ . \end{array}$$

Donc, avec $Y =_{\text{def}} Z_1 \cup Z' \cup Z''$ on a un sous-ensemble fini de $\mathcal{X}$ tel que $t_1[x \mapsto z], t'_1[x' \mapsto z]$ et $t''[x'' \mapsto z]$ sont α-équivalents pour tout $z \in \mathcal{X} \setminus Y$.
Comme $t'\sigma$ et $t''\sigma$ sont définis, on a $x', x'' \notin \mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$. Par conséquent, étant donné une variable $z \notin Y \cup \mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$, on a

$$t'_1\sigma[x' \mapsto z] = t'_1[x' \mapsto z]\sigma \qquad \text{et} \qquad t''_1\sigma[x'' \mapsto z] = t''_1[x'' \mapsto z]\sigma\ .$$

Il s'en suit que $t'_1[x' \mapsto z]\sigma$ et $t''_1[x'' \mapsto z]\sigma$ sont définis, et comme $t'_1[x' \mapsto z]$ et $t''_1[x'' \mapsto z]$ sont α-équivalents à $t_1[x \mapsto z]$, par hypothèse d'induction on a $t'_1[x' \mapsto z]\sigma =_\alpha t''_1[x'' \mapsto z]\sigma$.
Comme $t'_1\sigma[x' \mapsto z] =_\alpha t''_1\sigma[x'' \mapsto z]$ pour toute variable non dans l'ensemble fini $Y \cup \mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$, nous avons $\lambda x'.(t'_1\sigma) =_\alpha \lambda x''.(t''_1\sigma)$. Nous concluons que $(\lambda x'.t'_1)\sigma =_\alpha (\lambda x''.t''_1)\sigma$ car $x', x'' \notin \mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$.

(ii) Par induction sur t. Rappelons que σ et σ' ont même domaine et même variables libres.

$t = x \in \mathcal{X}$. Si $x \in \mathcal{D}om(\sigma)$, alors $t\sigma = \sigma(x) =_\alpha \sigma'(x)$; sinon, $t\sigma = x = t\sigma'$.

$t = f(\vec{t})$. Par hypothèse d'induction, $\vec{t}\sigma =_\alpha \vec{t}\sigma'$; donc $f(\vec{t})\sigma =_\alpha f(\vec{t})\sigma'$.

$t = t_1 t_2$. Par hypothèse d'induction, $t_i\sigma =_\alpha t_i\sigma'$ pour tout $i \in \{1, 2\}$. Il s'en suit que $(t_1 t_2)\sigma =_\alpha (t_1 t_2)\sigma'$.

$t = \lambda x.t_1$. Comme $t\sigma$ est défini, la variable x n'apparaît pas dans $\mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$; et par définition on a $t\sigma = \lambda x.(t_1\sigma)$ ainsi que $t\sigma' = \lambda x.(t_1\sigma')$. Par hypothèse d'induction on a $t_1\sigma =_\alpha t_1\sigma'$, donc $(\lambda x.t_1)\sigma =_\alpha (\lambda x.t_1)\sigma'$. □

Grâce à la propriété 1.3.6, nous pouvons définir la substitution sans capture sur $\Lambda(\Sigma)$ à partir de la substitution partielle sur $\mathcal{L}(\Sigma)$.

Proposition 1.3.7 *Soit une valuation $\sigma : \mathcal{X} \to \Lambda(\Sigma)$. La relation qui à chaque terme $[t] \in \Lambda(\Sigma)$ associe l'ensemble des $[t'\sigma']$ pour $t' \in [t]$ et $\sigma' \in [\sigma]$ tels que $t'\sigma'$ soit défini est une fonction totale de $\Lambda(\Sigma)$ vers $\Lambda(\Sigma)$.*

Preuve. Soit $[t] \in \Lambda(\Sigma)$ et $\sigma' \in [\sigma]$. Comme le domaine de σ' est fini, le lemme. 1.2.10 implique qu'il existe $t' \in [t]$ tel que $\mathsf{BV}(t')$ soit disjoint de $\mathcal{D}om(\sigma)$ et de $\mathsf{FV}(\sigma)$. Il s'en suit que l'ensemble des $[t'\sigma']$ tels que $t'\sigma'$ est défini n'est pas vide. La proposition 1.3.6 dit que c'est un singleton : si $t'' =_\alpha t'$, $\sigma'' =_\alpha \sigma'$ et $t''\sigma''$ est défini, alors $[t'\sigma'] = [t''\sigma'']$. □

Définition 1.3.8 (Substitution sans capture) *Étant donnée une valuation $[\sigma]$ de $\mathcal{X}$ dans $\Lambda(\Sigma)$, la* substitution sans capture sur $\Lambda(\Sigma)$, *notée $\_\,[\sigma] : \Lambda(\Sigma) \to \Lambda(\Sigma)$, est définie par $[t]\sigma =_{def} [t'\sigma']$ pour $\sigma' \in [\sigma]$ et $t' \in [t]$ tel que $t'\sigma'$ est défini.*

Nous identifions dès que possible une substitution $\_[\sigma]$ à sa valuation $[\sigma]$. Remarquons que si $t\sigma$ est défini, alors $[t][\sigma] = [t\sigma]$.

Nous allons maintenant vérifier qu'en utilisant la notation de la remarque 1.2.13, la substitution sans capture est bien celle que nous attendons. Rappelons que $\theta \circ \sigma$ est la substitution issue de la valuation $\theta \circ \sigma(x) = \theta(\sigma(x))$ de domaine $\mathcal{D}om(\sigma)$.

Lemme 1.3.9 *Soit un terme $t \in \Lambda(\Sigma)$ et une substitution σ.*

(i) Alors,
- $x\sigma = \sigma(x)$ *si* $x \in \mathcal{D}om(\sigma)$, $x\sigma = x$ *sinon,*
- $f(\vec{t})\sigma = f(\vec{t\sigma})$,
- $(t_1 t_2)\sigma = t_1\sigma\ t_2\sigma$,
- $(\lambda x.t_1)\sigma = \lambda x.(t_1\sigma)$ *si* $x \notin \mathcal{D}om(\sigma) \cup FV(\sigma)$, *et*
- $(\lambda x.t_1)\sigma = \lambda z.(t_1[z/x])\sigma$ *si* $z \notin FV(t_1) \cup \mathcal{D}om(\sigma) \cup FV(\sigma)$.

(ii) Si de plus θ est une substitution telle que $\mathcal{D}om(\sigma) \cap (\mathcal{D}om(\theta) \cup FV(\theta)) = \emptyset$, alors

$$(t\sigma)\theta \quad = \quad t\theta(\theta \circ \sigma)\ .$$

Preuve.

(i) On raisonne par cas sur t. Soient $t' \in \mathcal{L}(\Sigma)$ et σ' tels que $t = [t']$ et $\sigma = [\sigma']$, c'est à dire $t\sigma = [t'][\sigma']$.

$t = x \in \mathcal{X}$. Dans ce cas, $t' = x$ et $[t'][\sigma'] = [t'\sigma']$. On a $[t'\sigma'] = [\sigma'(x)] = \sigma(x)$ si $x \in \mathcal{D}om(\sigma)$ et $[t'\sigma'] = [x] = x$ sinon.

$t = f(\vec{t})$. Dans ce cas, $t' = f(\vec{t'})$ et $t\sigma = [f(\vec{t'})\sigma'] = [f(\vec{t'\sigma'})] = f([\vec{t'\sigma'}]) = f(\vec{t\sigma})$.

$t = t_1 t_2$. Dans ce cas, $t' = t'_1 t'_2$ et

$$t\sigma \quad = \quad [(t'_1 t'_2)\sigma'] \quad = \quad [t'_1\sigma'\ t'_2\sigma'] \quad = \quad [t'_1]\sigma'\ [t'_2]\sigma' \quad = \quad t_1\sigma\ t_2\sigma\ .$$

$t = \lambda x.t_1$. Tout d'abord, supposons que $x \notin \mathcal{D}om(\sigma) \cup FV(\sigma)$. Soit $u_1 \in t_1$ tel que $t_1\sigma = [u_1\sigma']$. Alors grâce à au lemme 1.2.12, on a

$$\begin{aligned}\lambda x.t_1\sigma &= \lambda x.[u_1\sigma'] \\ &= \{\lambda y.u'_1 \mid u'_1[y \mapsto z] =_\alpha u_1\sigma'[x \mapsto z] \text{ pour tout } z \text{ sauf un nombre fini}\} \\ &= [\lambda x.u_1\sigma']\ .\end{aligned}$$

Comme $x \notin \mathcal{D}om(\sigma) \cup FV(\sigma)$, on a $\lambda x.(u_1\sigma') = (\lambda x.u_1)\sigma'$, et on en déduit que

$$(\lambda x.t_1)\sigma = [\lambda x.u_1][\sigma'] = [(\lambda x.u_1)\sigma'] = [\lambda x.(u_1\sigma')] = \lambda x.(t_1\sigma)\ .$$

Maintenant, soit $z \notin \mathcal{D}om(\sigma) \cup FV(\sigma) \cup FV(t_1)$. Alors, grâce à ce qui précède, on a $\lambda z.(t_1[z/x]\sigma) = (\lambda z.t_1[z/x])\sigma$. Si on montre que $\lambda z.t_1[z/x] = \lambda x.t_1$, alors on obtient $\lambda z.(t_1[z/x]\sigma) = (\lambda x.t_1)\sigma$, ce qui conclut la démonstration.

Grâce au lemme 1.2.10, il existe $t'_1 \in t_1$ tel que

$$BV(t'_1) \cap (\{z\} \cup FV(\sigma) \cup \mathcal{D}om(\sigma)) \quad = \quad \emptyset\ .$$

Alors, la proposition 1.3.2 implique que $t'_1[z/x] = t'_1[x \mapsto z] \in t_1[z/x]$.
Comme $z \notin FV(t'_1) \cup BV(t'_1)$, en utilisant la proposition 1.1.18, on en déduit que pour toute variable z', $t'_1[x \mapsto z'] = t'_1[z/x][z \mapsto z']$. Donc

$$\begin{aligned}\lambda x.t_1 &= \lambda x.[t'_1] \\ &= \{\lambda y.u_1 \mid u_1[y \mapsto z'] =_\alpha t'_1[x \mapsto z'] \quad \forall\, z' \text{ sauf un nombre fini}\} \\ &= \{\lambda y.u_1 \mid u_1[y \mapsto z'] =_\alpha t'_1[z/x][z \mapsto z'] \quad \forall\, z' \text{ sauf un nombre fini}\} \\ &= \lambda z.[t'_1[z/x]] = \lambda z.t_1[z/x]\ .\end{aligned}$$

(ii) Soit $\sigma' \in \sigma$ et $\theta' \in \theta$ telle que $Dom(\sigma) \cap (BV(\theta')) = \emptyset$ et $t' \in t$ tel que $(t'\sigma')\theta'$ soit défini.
Comme $V(\theta') \subseteq (FV(\theta') \cup BV(\theta'))$ et $Dom(\sigma') \cap (Dom(\theta') \cup FV(\theta') \cup BV(\theta')) = \emptyset$, on a $Dom(\sigma') \cap (Dom(\theta') \cup V(\theta')) = \emptyset$, donc $\theta'(\sigma'(t')) = \theta' \circ \sigma'(\theta'(t'))$ par le lemme 1.1.17. Or, la proposition 1.3.2 implique que $(t'\sigma')\theta' = \theta'(\sigma'(t'))$ et $t'\theta'(\theta' \circ \sigma') = \theta' \circ \sigma'(\theta'(t'))$.
On en déduit que $[(t'\sigma')\theta'] = [t'\theta'(\theta' \circ \sigma')]$, soit $(t\sigma)\theta = t\theta(\theta \circ \sigma)$. □

Remarque 1.3.10 *Le lemme 1.3.9.(ii) implique qu'étant donné un terme $\lambda x.t$ et une substitution σ, on peut toujours se ramener à un cas où $x \notin Dom(\sigma) \cup FV(\sigma)$ tout en restant dans la même classe d'α-équivalence : si $x \in Dom(\sigma) \cup FV(\sigma)$, alors pour tout $z \notin Dom(\sigma) \cup FV(\sigma) \cup FV(t)$, on a $(\lambda x.t)\sigma = \lambda z.(t[z/x]\sigma) = (\lambda z.t[z/x])\sigma$ dans $\Lambda(\Sigma)$.*

Pour finir, voici comment définir une substitution sur un sous-ensemble L de $\Lambda(\Sigma)$, et comment relier deux substitutions par une relation et une fonction.

Définition 1.3.11 *Soit $L \subseteq \Lambda(\Sigma)$. Si $\sigma : \mathcal{X} \to \Lambda(\Sigma)$ est une substitution telle que $t\sigma \in L$ pour tout $t \in L$, alors c'est* une L-substitution, *notée $\sigma : \mathcal{X} \to L$.*

Définition 1.3.12 *Étant donnés $L \subseteq \Lambda(\Sigma)$, une relation $R \subseteq L \times L$ et deux substitutions $\sigma, \sigma' : \mathcal{X} \to L$, on note $\sigma\ R\ \sigma'$ si $Dom(\sigma) = Dom(\sigma')$ et si $\sigma(x)\ R\ \sigma'(x)$ pour tout $x \in Dom(\sigma)$.*
Dans le cas où $\sigma\ F\ \sigma'$ pour une fonction $F : L \to L$, on pose $F(\sigma) =_{def} \sigma'$.

1.4 Termes du premier ordre parmi les lambda-termes

Nous nous intéressons, dans cette section, aux sous-ensembles de $\Lambda(\Sigma)$ constitués des termes applicatifs et des termes « du premier ordre ». Nous montrons que notre notion d'algèbre $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ libre pour une injection $i : \mathcal{X} \to A$ donnée nous permet de voir ces ensembles respectivement comme une $(\Sigma \cup \{\_ \cdot \_\})$-algèbre et comme une Σ-algèbre, toutes les deux libres sur $\mathcal{X}$.

Définition 1.4.1 *On définit par $\Sigma_{App} =_{def} \Sigma \uplus \{\_ \cdot \_\}$ l'extension applicative d'une signature Σ.*

Nous commençons par le cas des termes applicatifs, qui donnent lieu à une Σ_{App}-algèbre libre sur $\mathcal{X}$.

Définition 1.4.2 (Termes applicatifs)

(i) L'ensemble $\mathcal{L}_{\mathrm{App}}(\Sigma)$ *des* λ-termes bruts applicatifs *sur* $(\Sigma, \mathcal{X})$ *est l'ensemble des termes générés par la grammaire suivante :*

$$t, u \in \mathcal{L}_{\mathrm{App}}(\Sigma) \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n) \quad | \quad t{\cdot}u$$

où $x \in \mathcal{X}$ *et* $f \in \Sigma_n$.

(ii) L'ensemble $\Lambda_{\mathrm{App}}(\Sigma)$ *des* λ-termes applicatifs *sur* $(\Sigma, \mathcal{X})$ *est l'ensemble*

$$\Lambda_{\mathrm{App}}(\Sigma) \quad =_{def} \quad \{[t] \mid t \in \mathcal{L}_{\mathrm{App}}(\Sigma)\} .$$

Les λ-termes applicatifs sont des singletons.

Proposition 1.4.3 *Si* t *est un terme brut applicatif, alors* $[t] = \{t\}$.

Notre notion d'algèbre libre sur $\mathcal{X}$ nous permet de voir $\Lambda_{\mathrm{App}}(\Sigma)$ comme une Σ_{App}-algèbre libre pour l'injection $x \in \mathcal{X} \mapsto [x] = \{x\} \in \Lambda_{\mathrm{App}}(\Sigma)$.

Théorème 1.4.4 *Soit* Σ_{App} *l'ensemble constitué des opérations suivantes, définies en 1.2.12 :*

— $\_ \cdot \_ : \Lambda(\Sigma) \times \Lambda(\Sigma) \to \Lambda(\Sigma)$,

— $f(\_, \ldots, \_) : \Lambda(\Sigma)^n \to \Lambda(\Sigma)$ *pour chaque* $f \in \Sigma_n$.

Alors $(\Lambda_{\mathrm{App}}(\Sigma), \Sigma_{\mathrm{App}})$ *est une* Σ_{App}*-algèbre libre sur* $\mathcal{X}$ *pour l'injection* $x \in \mathcal{X} \mapsto [x]$.

De ce fait, toutes les constructions de la section 1.1 sont définies sur $\Lambda_{\mathrm{App}}(\Sigma)$. De plus, elles correspondent à celles sur $\mathcal{L}_{\mathrm{App}}(\Sigma)$:

— Pour tout $t \in \mathcal{L}_{\mathrm{App}}(\Sigma)$, on a $\mathsf{V}([t]) = \mathsf{V}(t)$, de plus $\mathsf{V}([t]) = \mathsf{FV}([t])$.

— Pour tout $t, u \in \mathcal{L}_{\mathrm{App}}(\Sigma)$, et $[v] \in \Lambda(\Sigma)$ on a $\mathcal{P}\mathsf{os}([t]) = [\mathcal{P}\mathsf{os}(t)]$, $[t]|_p = [t|_p]$, $[t][[v]]_p = [t[v]_p]$ et $\mathcal{O}\mathsf{cc}([t], [u]) = \mathcal{O}\mathsf{cc}(t, u)$.

— Pour toute substitution du premier ordre $\sigma(\_) : \mathcal{X} \to \mathcal{L}_{\mathrm{App}}(\Sigma)$, on définit $\_\sigma : \mathcal{X} \to \Lambda_{\mathrm{App}}(\Sigma)$ par $x\sigma =_{\mathrm{def}} [\sigma(x)]$ pour tout $x \in \mathcal{D}\mathsf{om}(\sigma(\_))$. On a alors $[u]\sigma = [\sigma(u)]$ pour tout $u \in \mathcal{L}_{\mathrm{App}}(\Sigma)$.

Nous écrivons $\mathcal{T}\mathsf{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ pour désigner $(\Lambda_{\mathrm{App}}(\Sigma), \Sigma_{\mathrm{App}})$. Il est facile d'en déduire une Σ-algèbre libre sur $\mathcal{X}$ en utilisant la propriété suivante.

Proposition 1.4.5 *Soient* Σ *et* Σ' *deux signatures telles que* $\Sigma' \subseteq \Sigma$ *et* $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ *une* Σ*-algèbre. Alors* $\mathcal{A}' =_{def} (A, \{f_{\mathcal{A}} \mid f \in \Sigma'\})$ *est une* Σ'*-algèbre, qui de plus est libre sur* $\mathcal{X}$ *pour* i *si* $\mathcal{A}$ *est libre sur* $\mathcal{X}$ *pour* i.

Nous notons $\mathcal{T}\mathsf{er}(\Sigma, \mathcal{X})$ la Σ-algèbre issue de $\mathcal{T}\mathsf{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ par la proposition 1.4.5. Elle est donc libre sur $\mathcal{X}$ pour l'injection $x \in \mathcal{X} \mapsto [x] = \{x\} \in \Lambda_{\mathrm{App}}(\Sigma)$.

Remarque 1.4.6 *Les contextes sur* $L \subseteq \mathcal{T}\mathsf{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ *définis en 1.1.19 sont isomorphes aux* λ*-contextes sur* $L \subseteq \mathcal{T}\mathsf{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ *définis en 1.2.18 : tout* λ*-contexte sur* $L \subseteq \mathcal{T}\mathsf{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ *peut être vu comme un terme* $C \in L$ *contenant exactement une occurrence de la variable* $[\ ]$. *On a alors* $C[t] = C[t/[\ ]]$ *pour tout* $t \in \Lambda(\Sigma)$.

Inversement, tout contexte C *sur* $L \subseteq \mathcal{T}\mathsf{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ *définit un unique* λ*-contexte* $C[\ ]$ *sur* L *tel que* $C[t/[\ ]] = C[t]$ *pour tout* $t \in \Lambda(\Sigma)$.

1.5 Convention de Barendregt locale

Revenons sur l'exemple 1.2.8.(iii), qui dit que

$$\lambda x.\lambda x.x \quad =_\alpha \quad \lambda x.\lambda y.x\ .$$

Cet exemple contredit la notion d'α-conversion utilisée dans [Bar84], qui est en fait la relation d'équivalence identifiant les termes ayant même représentant de de Bruijn.

Le problème vient du fait que dans $\lambda x.\lambda x.x$, la variable x liée par le premier λ est aussi liée par le second λ. Il faudrait choisir laquelle de ces deux abstractions lie la variable de tête de $\lambda x.\lambda x.x$, et sur ce point, l'α-conversion de Krivine ne coïncide pas avec l'identification des termes ayant même représentation de de Bruijn.

Afin d'avoir une correspondance entre la notion d'α-conversion formelle de Krivine et l'identification des λ-termes par leur représentants de de Bruijn, nous pouvons ne pas considérer les termes, tels que $\lambda x.\lambda x.x$, qui ont un sous-terme de la forme $\lambda x.u$ dans lequel $x \in \mathrm{BV}(u)$.

Cette restriction est un sous-cas de la « convention de Barendregt ». Telle qu'énoncée en 2.1.13 dans [Bar84], c'est une condition globale sur les λ-termes apparaissant dans les énoncés, preuves, etc, qui dit que les variables libres et liées sont toujours différentes.

En fait, nous n'avons pas besoin d'une méta-condition globale sur tous les termes apparaissant dans le texte. Nous utilisons une version locale de la convention de Barendregt, qui s'énonce comme une restriction de l'ensemble de termes considéré, et dont on peut prouver la correction.

Définition 1.5.1 (Convention de Barendregt locale)

- *Un terme* $t \in \mathcal{L}(\Sigma)$ vérifie la convention de Barendregt locale *si pour tout sous-terme* $\lambda x.u$ *de* t*, on a* $x \notin \mathrm{BV}(u)$.
- *L'ensemble des termes qui vérifient la convention de Barendregt locale est noté* $\mathcal{L}_{\mathrm{B}}(\Sigma)$.
- *On dit qu'une substitution partielle sans capture* σ vérifie la convention de Barendregt locale *si* $\sigma(x) \in \mathcal{L}_{\mathrm{B}}(\Sigma)$ *pour tout* $x \in \mathcal{D}\mathrm{om}(\sigma)$.

Lorsque le contexte le permet, on dit « convention de Barendregt » au lieu de « convention de Barendregt locale ».

Voyons maintenant que l'on peut se restreindre aux termes vérifiant cette propriété.

Proposition 1.5.2 *Soient un terme* $t \in \mathcal{L}(\Sigma)$ *et* σ *une substitution partielle sans capture. Alors,*

(i) il existe $t' =_\alpha t$ *tel que* $t' \in \mathcal{L}_{\mathrm{B}}(\Sigma)$*,*

(ii) si t *et* σ *vérifient la convention de Barendregt et sont tels que* $t\sigma$ *soit défini, et si de plus* $\mathrm{BV}(t) \cap \mathrm{BV}(\sigma) = \emptyset$*, alors* $t\sigma$ *vérifie la convention de Barendregt,*

(iii) il existe t' *et* σ' *vérifiant la convention de Barendregt, respectivement* α*-équivalents à* t *et* σ *tels que* $t'\sigma'$ *soit défini et vérifie la convention de Barendregt.*

Preuve. En utilisant le lemme 1.2.10. □

Exemple 1.5.3 *Le terme* $\lambda x.\lambda x.x$ *qui ne vérifie pas la convention de Barendregt est* α*-équivalent au terme* $\lambda x.\lambda y.x$ *qui appartient à* $\mathcal{L}_B(\Sigma)$.

1.6 Termes de de Bruijn

Afin de pouvoir passer facilement des termes bruts du λ-calcul, c'est à dire non quotientés par $=_\alpha$, aux λ-termes, il s'avère très pratique d'utiliser des représentants canoniques de ses derniers. En fait, nous ne choisissons pas vraiment un représentant *dans* chaque classe d'α-équivalence ; au lieu de cela, nous utilisons des termes d'une autre syntaxe qui est isomorphe à $\Lambda(\Sigma)$, ce sont les termes de de Bruijn.

Dans cette section, nous montrons que l'α-conversion de Krivine identifie exactement les termes vérifiant la convention de Barendregt qui ont même représentant de de Bruijn. Ce résultat est à notre connaissance nouveau.

Il permet, une fois définie une traduction inverse des termes de de Bruijn vers les λ-termes vérifiant la convention de Barendregt, de montrer que ces deux ensembles sont isomorphes modulo $=_\alpha$.

Définition 1.6.1 (Termes de de Bruijn) *Étant donnée une signature* Σ*, l'ensemble* $\mathcal{T}_{\mathtt{dB}}(\Sigma, \mathcal{X})$ *des* termes de de Bruijn *sur* Σ *est défini par la grammaire suivante :*

$$t, u \in \mathcal{T}_{\mathtt{dB}}(\Sigma, \mathcal{X}) \quad ::= \quad x \;\mid\; m \;\mid\; f(t_1, \ldots, t_n) \;\mid\; t{\cdot}u \;\mid\; \lambda t$$

où $x \in \mathcal{X}$*,* $m \in \mathbb{N}$ *et* $f \in \Sigma_n$.

L'ensemble $\mathcal{T}_{\mathtt{dB}}(\Sigma, \mathcal{X})$ est donc un ensemble de termes sur $(\Sigma_{\mathtt{dB}}, \mathcal{X})$, où $\Sigma_{\mathtt{dB}}$ est la signature $\Sigma \uplus \{\lambda\_, \_ \cdot \_\} \uplus \{m \mid m \in \mathbb{N}\}$.

Habituellement, les termes de de Bruijn (et c'est tout leur intérêt) ne contiennent pas de variables de termes ; elles sont encodées dans les indices. Cependant, pour notre utilisation des termes de de Bruijn, on gagne en souplesse si on autorise des termes avec des variables libres. Cela nous sera utile à la section 4.3 lorsque nous utilisons les termes de de Bruijn pour étudier la correspondance entre la réécriture sur $\Lambda(\Sigma)$ et la réécriture sur $\mathcal{L}(\Sigma)$. Notons que ces variables ne sont pas utilisées de la même manière que les métavariables d'unification de [DHK00].

La notion habituelle de variable libre correspond à la notion d'indice libre pour les termes de de Bruijn.

Définition 1.6.2 *La hauteur de de Bruijn d'une occurrence* p *dans* $t \in \mathcal{T}_{\mathtt{dB}}(\Sigma, \mathcal{X})$*, notée* $|p|_t$*, est définie inductivement comme suit :*

$$\begin{array}{lll} |\varepsilon|_t & =_{def} & 0 \\ |1.p|_{\lambda t} & =_{def} & |p|_t + 1 \\ |i.p|_{f(t_1,\ldots,t_n)} & =_{def} & |p|_{t_i} \\ |i.p|_{t_1\, t_2} & =_{def} & |p|_{t_i} \ . \end{array}$$

Une occurrence p *d'un indice* m *dans* t *est dite* libre *si* $m > |p|_t$.

Précisons, pour ne pas faire d'ambiguïté, que pour les termes de de Bruijn, nous utilisons l'expression « terme clos » de la même manière que pour les termes du premier ordre : les termes de de Bruijn clos sont ceux de $\mathcal{T}\mathtt{dB}(\Sigma)$.

Voyons maintenant comment les termes de de Bruijn représentent les classes d'α-équivalence. Pour cela, nous utilisons la traduction habituelle des λ-termes vers les termes de de Bruijn.

Définition 1.6.3 *Un référentiel pour un terme $t \in \mathcal{L}_B(\Sigma)$ est une liste R de variables distinctes telles que $FV(t) \subseteq R$ et $BV(t) \cap R = \emptyset$.*

Si R est un référentiel pour un terme, et $m \in \mathbb{N}$ est tel que $m \leq |R|$, alors $R[m]$ est la m-ème variable de R.

Définition 1.6.4 *Étant donné un terme $t \in \mathcal{L}_B(\Sigma)$ et un référentiel R pour t, le représentant de de Bruijn de t dans R est le terme $t_R \in \mathcal{T}\mathtt{dB}(\Sigma)$ défini inductivement comme suit :*

$$\begin{array}{lll} x_R & =_{def} & m \qquad \text{où} \quad R[m] = x \\ f(\vec{t})_R & =_{def} & f(\vec{t_R}) \\ (t_1 t_2)_R & =_{def} & t_{1R}\, t_{2R} \\ (\lambda x.t_1)_R & =_{def} & \lambda(t_{1\, x\cdot R}) \end{array}$$

où $x \cdot R$ dénote la liste de tête x et de queue R.

Remarquons que la définition est correcte car on suppose que t vérifie la convention de Barendregt. Autrement, dans le cas d'une abstraction $(\lambda x.t_1)_R$, on ne pourrait pas assurer l'hypothèse d'induction que $x \cdot R$ est fait de variables distinctes.

La condition de linéarité des référentiels permet d'obtenir la proposition suivante, qui dit que la traduction de de Bruijn ne varie pas lorsqu'une variable libre est renommée par une variable fraîche, dès lors que ce renommage est aussi effectué dans le référentiel. On note $\vec{x} \cdot \vec{y}$ pour la concaténation des listes $\vec{x}$ et $\vec{y}$. Si R est la liste linéaire $\vec{x} \cdot x \cdot \vec{y}$, alors $R[x \mapsto z]$ est la liste $\vec{x} \cdot z \cdot \vec{y}$.

Proposition 1.6.5 (Invariance par renommage) *Soit $t \in \mathcal{L}_B(\Sigma)$ et R un référentiel pour t. Alors, pour tout $z \notin FV(t) \cup BV(t) \cup R$ et tout $x \notin FV(t)$, on a*

$$(t[x \mapsto z])_{R[x \mapsto z]} \quad = \quad t_R \ .$$

Preuve. On montre, par induction sur t, que si $\vec{x} \cdot x \cdot \vec{y}$ est un référentiel pour t et $z \notin BV(t) \cup FV(t) \cup \vec{x} \cdot x \cdot \vec{y}$ alors $(t[x \mapsto z])_{\vec{x} \cdot z \cdot \vec{y}} = t_{\vec{x} \cdot x \cdot \vec{y}}$.

$t = y \in \mathcal{X}$. Si $x = y$, alors $t[x \mapsto z]_{\vec{x} \cdot z \cdot \vec{y}} = z_{\vec{x} \cdot z \cdot \vec{y}}$. On a $z_{\vec{x} \cdot z \cdot \vec{y}} = x_{\vec{x} \cdot x \cdot \vec{y}}$ car $z, x \notin \vec{x} \cdot \vec{y}$.
Sinon, $t[x \mapsto z]_{\vec{x} \cdot z \cdot \vec{y}} = y_{\vec{x} \cdot z \cdot \vec{y}} = y_{\vec{x} \cdot x \cdot \vec{y}}$ car $x, z \neq y$.

$t = f(\vec{t})$. Comme $z \notin FV(t_i) \cup BV(t_i) \cup \vec{x} \cdot x \cdot \vec{y}$ pour tout $i \in \{1, \ldots, n\}$, par hypothèse d'induction on a $t_i[x \mapsto z]_{\vec{x} \cdot z \cdot \vec{y}} = t_{i\vec{x} \cdot x \cdot \vec{y}}$, d'où le résultat recherché.

$t = t_1 t_2$. Idem.

$t = \lambda y.t_1$. Alors, comme $z \notin BV(t) \cup FV(t) \cup \vec{x} \cdot x \cdot \vec{y}$, on a $z \notin BV(t_1) \cup FV(t_1) \cup y \cdot \vec{x} \cdot x \cdot \vec{y}$, donc par hypothèse d'induction, comme $y \notin \vec{x} \cdot x \cdot \vec{y}$ et comme $y \notin BV(t_1)$ (par la convention de Barendregt) on a $t_1[x \mapsto z]_{y \cdot \vec{x} \cdot z \cdot \vec{y}} = t_{1\, y \cdot \vec{x} \cdot x \cdot \vec{y}}$, d'où $(\lambda y.t_1)[x \mapsto z]_{\vec{x} \cdot z \cdot \vec{y}} = (\lambda y.t_1)_{\vec{x} \cdot x \cdot \vec{y}}$. □

On en déduit que sur les termes vérifiant la convention de Barendregt locale, l'α-conversion de Krivine identifie exactement les termes ayant même traduction de de Bruijn.

Lemme 1.6.6 (Lemme fondamental des termes de de Bruijn) *Pour tout* $t, u \in \mathcal{L}_B(\Sigma)$ *et pour tout référentiel* R *pour* t *et* u*, on a*

$$t =_\alpha u \qquad \textit{si et seulement si} \qquad t_R = u_R \ .$$

Preuve. On raisonne par induction sur la taille de t.

$t = x \in \mathcal{X}$. On a $t =_\alpha u$ si et seulement si $t = u$ si et seulement si $t_R = u_R$.

$t = f(\vec{t})$. On a $t =_\alpha u$ si et seulement si $u = f(u_1, \dots, u_n)$ avec $t_i =_\alpha u_i$ pour tout $i \in \{1, \dots, n\}$, donc, par hypothèse d'induction, si et seulement si $(t_i)_R = (u_i)_R$ pour tout $i \in \{1, \dots, n\}$, si et seulement si $t_R = u_R$.

$t = t_1 t_2$. Idem.

$t = \lambda x.t_1$. Supposons que $t =_\alpha u$. Dans ce cas, $u = \lambda y.u_1$, et il existe $Z \subseteq_{fin} \mathcal{X}$ tel que $t_1[x \mapsto z] =_\alpha u_1[y \mapsto z]$ pour tout $z \in \mathcal{X} \backslash Z$. Donc, si $z \notin Z \cup FV(t_1, u_1) \cup BV(t_1, u_1)$, alors par la proposition 1.6.5, on a $t_1[x \mapsto z]_{z \cdot R} = t_{1\, x \cdot R}$ et $u_1[y \mapsto z]_{z \cdot R} = u_{1\, y \cdot R}$. Or, par hypothèse d'induction, comme $t_1[x \mapsto z] =_\alpha u_1[y \mapsto z]$, on a $t_1[x \mapsto z]_{z \cdot R} = u_1[y \mapsto z]_{z \cdot R}$. On en déduit que $(\lambda x.t_1)_R = (\lambda y.u_1)_R$.

Réciproquement, supposons que $t_R = u_R$. On a donc $t = \lambda x.t_1$ et $u = \lambda y.u_1$ avec $(t_1)_{x \cdot R} = (u_1)_{y \cdot R}$. Donc, par la proposition 1.6.5, on a $(t_1[x \mapsto z])_{z \cdot R} = (u_1[y \mapsto z])_{z \cdot R}$ pour tout $z \notin (BV(t, u) \cup FV(t, u) \cup R)$. Par hypothèse d'induction (t_1 et $t_1[x \mapsto z]$ ont même taille), on obtient $t_1[x \mapsto z] =_\alpha u_1[y \mapsto z]$ pour toute variable z non dans l'ensemble fini $BV(t, u) \cup FV(t, u) \cup R$, c'est-à-dire $\lambda x.t_1 =_\alpha \lambda y.u_1$. □

Remarque 1.6.7 *Le lemme 1.6.6 n'est en général pas vrai sur les termes qui ne vérifient pas la convention de Barendregt. Les référentiels construits à partir de ces termes peuvent contenir des doublons. Par exemple,* $(\lambda x.\lambda x.x). = (\lambda x.x)_x = x_{x \cdot x}$*. Il faut alors choisir, pour le terme* $x_{x \cdot x}$*, l'indice qui correspond à* x *dans la liste* $x \cdot x$*. La sémantique habituelle (voir par exemple [DHK00]) est de prendre le premier :* $x_{x \cdot x} = 1$*. Ceci correspond à l'idée que dans le sous-terme* $\lambda x.x$ *de* $\lambda x.\lambda x.x$*, toute occurrence de* x *est liée par l'abstraction de* $\lambda x.x$*. Le terme* $\lambda x.\lambda x.x$ *devrait donc être* α*-équivalent à* $\lambda y.\lambda x.x$*.*

C'est pour cela que nous utilisons la convention de Barendregt locale.

Afin de pouvoir passer de $\mathcal{T}dB(\Sigma)$ à $\mathcal{L}_B(\Sigma)$, nous utilisons l'inverse de l'opération $(\_)_R$ définie en 1.6.4. À notre connaissance, cette définition est originale.

Définition 1.6.8 *Soit* $t \in \mathcal{T}dB(\Sigma, \mathcal{X})$*. Une liste* R *de variables distinctes est un* référentiel pour t *si sa longueur est plus grande que le plus grand indice libre dans* t*.*

Étant donné un référentiel R *pour* $t \in \mathcal{T}dB(\Sigma)$ *et une énumération* $(x_i)_{i\in\mathbb{N}}$ *de* $\mathcal{X}$*, on définit* $t^R \in \mathcal{L}_B(\Sigma)$ *par induction sur* t *comme suit :*

$$\begin{array}{ll} m^R & =_{def} R[m] \\ (f(t_1,\dots,t_n))^R & =_{def} f(t_1^R,\dots,t_n^R) \\ (t_1\ t_2)^R & =_{def} t_1^R\ t_2^R \\ (\lambda t)^R & =_{def} \lambda x_{|R|+1}.(t^{x_{|R|+1}\cdot R}) \end{array}$$

où $|R|$ *est le plus grand indice de variable apparaissant dans* R *pour l'énumération* $(x_i)_{i\in\mathbb{N}}$ *(voir 1.1.11).*

On vérifie aisément que $FV(t^R) \subseteq R$ et donc que $t^R \in \mathcal{L}_B(\Sigma)$ pour tout $t \in \mathcal{T}dB(\Sigma)$. Les opérations $(\_)_R$ et $(\_)^R$ sont inverses l'une de l'autre *modulo* α-conversion.

Théorème 1.6.9 (Isomorphismes de de Bruijn)

(i) Pour tout $\nu \in \mathcal{T}dB(\Sigma)$*, si* R *est un référentiel pour* ν*, on a* $(\nu^R)_R = \nu$*.*

(ii) Pour tout $\nu \in \mathcal{L}_B(\Sigma)$*, si* R *est un référentiel pour* ν*, on a* $(\nu_R)^R =_\alpha \nu$*.*

Preuve.

(i) Par induction sur $\nu \in \mathcal{T}dB(\Sigma)$.

$\nu = m$. On a $(\nu^R)_R = (R[m])_R = m$.

$\nu = f(t_1,\dots,t_n)$, $\nu = t_1\ t_2$. Par hypothèse d'induction.

$\nu = \lambda t$. On a $(\nu^R)_R = (\lambda x_{|R|+1}.(t^{x_{|R|+1}\cdot R}))_R = \lambda((t^{x_{|R|+1}\cdot R})_{x_{|R|+1}\cdot R})$. Donc, par hypothèse d'induction, $(\nu^R)_R = \lambda t = \nu$.

(ii) On déduit de (i) que pour tout $\nu \in \mathcal{L}_B(\Sigma)$, on a $((\nu_R)^R)_R = \nu_R$, d'où $(\nu_R)^R =_\alpha \nu$ par le lemme fondamental des termes de de Bruijn (1.6.6). □

Pour finir, voici deux propriétés « évidentes » des termes de de Bruijn.

Remarque 1.6.10

(i) Si $t \in \mathcal{T}dB(\Sigma,\mathcal{X})$ *est applicatif et* σ *est une substitution telle que* $t\sigma \in \mathcal{T}dB(\Sigma)$*, alors* $(t\sigma)^R = t(\sigma^R)$*.*

(ii) Si $t \in \mathcal{L}(\Sigma)$ *est applicatif et* σ *est une substitution, alors* $(t\sigma)_R = t(\sigma_R)$*.*

Chapitre 2

Réductions

Dans ce chapitre, nous présentons les notions et notations de base pour manier des relations de réécriture et des systèmes de réécriture.

Les relations de réécriture sont des relations binaires sur un ensemble de termes qui sont stables par contexte et substitution. Les relations de réécriture qui nous intéressent sont générées par des ensembles de règles de transformations locales sur les termes, appelés systèmes de réécriture.

Nous commençons par rappeler les propriétés des relations de réécriture qui peuvent s'énoncer au niveau des relations binaires. Nous rappelons ensuite les relations de réécriture, et insistons sur les relations de réécriture parallèles qui nous seront très utiles à la partie II. Enfin, nous rappelons la notion de système de réécriture, ainsi que les relations de réécritures qui en sont issues.

2.1 Systèmes de réduction abstraits

Nous présentons ici certaines notions et propriétés utiles pour les relations de réécriture, mais qui peuvent être comprises dans le cadre plus abstrait des relations binaires.

Définition 2.1.1 (Systèmes de réduction abstraits) *Un* système de réduction abstrait *(ou* ARS*) est une relation binaire notée* $(A, \to)$.

L'intérêt de la notion d'ARS n'est donc pas sa définition, mais plutôt qu'elle permet de distinguer les propriétés fondamentales des relations de réécriture qui peuvent être exprimées pour des relations binaires arbitraires.

Étant donné un ARS $(A, \to)$, nous écrivons $a \to b$ pour $(a, b) \in \to$. Lorsque le contexte le permet, nous désignons l'ARS $(A, \to)$ par $\to$.

Remarque 2.1.2 *Dans certains contextes il peut être intéressant d'avoir une définition plus fine de système de réduction abstrait. C'est par exemple le cas dans [Ter03], où un système de réduction abstrait est un ensemble équipé d'une famille de relations binaires* $(\to_i)_{i \in I}$.

Définition 2.1.3 (Réductions) *Étant donné un ARS* $(A, \to)$*, une* $(A, \to)$*-réduction est une famille* $(a_i)_{i \in I}$ *d'éléments de* A*, indexée par un ensemble non vide* $I \subseteq \mathbb{N}$ *et telle que pour tout* $i \in \mathbb{N}$*, si* $i + 1 \in I$ *alors* $i \in I$ *et* $a_i \to a_{i+1}$.

Une $(A, \to)$*-réduction* $(a_i)_{i \in I}$ *est dite* finie *si* I *est fini, et* issue de a *si* $a_0 = a$.

Bien entendu, si $(a_i)_{i\in I}$ est une $\to$-réduction, alors $0 \in I$.

Définition 2.1.4 (Relations dérivées) *Étant donné un ARS* $(A, \to)$*, définissons*
— *sa* clôture transitive $(A, \to^+)$ *par*

$$\to^+ \quad =_{def} \quad \{(a_0, a_i) \mid (a_i)_{i\in I} \textit{ est une } \to\textit{-réduction et } i \in I\} \ ,$$

— *sa* clôture réflexive $(A, \to^=)$ *par* $\to^= \ =_{def} \ \to \cup\{(a, a) \mid a \in A\}$,
— *sa* clôture réflexive et transitive $(A, \to^*)$ *par* $\to^* \ =_{def} \ \to^= \cup \to^+$,
— *son* symétrique $(A, \leftarrow)$ *par* $\leftarrow \ =_{def} \{(a, b) \mid b \to a\}$,
— *sa* clôture symétrique $(A, \leftrightarrow)$ *par*

$$\leftrightarrow \quad =_{def} \quad \{(a_0, a_i) \mid (a_i)_{i\in I} \textit{ est une } (\to \cup \leftarrow)\textit{-réduction et } i \in I\} \ ,$$

— *sa* clôture par joignabilité $(A, \downarrow)$ *par* $\downarrow \ =_{def} \{(a, b) \mid \exists c \ (a \to^* c \leftarrow^* b)\}$,
— *se* clôture en au plus $n \in \mathbb{N}$ pas $(A, \to^n)$ *par*

$$\to^n \quad =_{def} \quad \{(a_0, a_m) \mid (a_i)_{0\le i\le n} \textit{ est une } \to\textit{-réduction et } m \le n\} \ .$$

Avec ces notations, $\leftrightarrow^*$ désigne la clôture réflexive et transitive de $\leftrightarrow$, c'est donc la clôture réflexive, symétrique et transitive de $\to$. Il nous arrive de la dénoter aussi par $=$.

Définition 2.1.5 (Réduits et formes normales) *Soit un ARS* $(A, \to)$ *et* $a \in A$.
— *L'*ensemble des $(A, \to)$-réduits en un pas de $a \in A$ *est* $(a)_\to =_{def} \{b \mid a \to b\}$.
— *L'*ensemble des $(A, \to)$-réduits de $a \in A$ *est* $(a)^*_\to =_{def} \{b \mid a \to^* b\}$.
Un élément $a \in A$ *est* en forme normale pour $(A, \to)$ *(ou encore, est* en forme $(A, \to)$-normale*) si* $(a)_\to = \emptyset$*. L'ensemble des éléments en forme normale pour* $(A, \to)$ *est dénoté par* $\mathcal{NF}_{(A,\to)}$*. Un élément* a *de* A a une forme $(A, \to)$-normale *si* $(a)^*_\to$ *contient une forme* $(A, \to)$*-normale.*

Lorsque le contexte le permet, nous écrivons « forme normale » au lieu de « forme normale pour $(A, \to)$ » et $\mathcal{NF}$ au lieu de $\mathcal{NF}_\to$.

Définition 2.1.6 (Normalisation) *Soit un ARS* $(A, \to)$.

(i) L'ensemble des éléments fortement normalisants *de* $(A, \to)$ *est le plus petit ensemble* $\mathcal{SN}_{(A,\to)}$ *tel que pour tout* $a \in A$*,*

$$\big(\forall b \in A.\ a \to b \quad \Longrightarrow \quad b \in \mathcal{SN}_{(A,\to)}\big) \qquad \Longrightarrow \qquad a \in \mathcal{SN}_{(A,\to)} \ .$$

(ii) L'ensemble des éléments faiblement normalisants *de* A *est le plus petit ensemble* $\mathcal{WN}_{(A,\to)}$ *contenant* $\mathcal{NF}_{(A,\to)}$ *et tel que pour tout* $a \in A$*,*

$$\big(\exists b \in A.\ a \to b \quad \wedge \quad b \in \mathcal{WN}_{(A,\to)}\big) \qquad \Longrightarrow \qquad a \in \mathcal{WN}_{(A,\to)} \ .$$

Un ARS $(A, \to)$ *est dit* fortement normalisant *si* $A = \mathcal{SN}_{(A,\to)}$ *et* faiblement normalisant *si* $A = \mathcal{WN}_{(A,\to)}$.

Lorsque le contexte le permet, nous écrivons $\mathcal{SN}_{\to}$ au lieu de $\mathcal{SN}$ et $\mathcal{WN}_{\to}$ au lieu de $\mathcal{WN}$.

Proposition 2.1.7 *Un élément* a *d'un ARS* $(\mathsf{A}, \to)$ *est*
- *faiblement normalisant si et seulement si il a une forme normale,*
- *fortement normalisant si et seulement si toutes les réductions issues de* a *sont finies.*

Définition 2.1.8 (Confluence) *Un ARS* $(\mathsf{A}, \to)$
- *est* confluent *si pour tout* $\mathsf{a}, \mathsf{b}, \mathsf{c} \in \mathsf{A}$*, si* $\mathsf{a} \leftarrow^* \mathsf{b} \to^* \mathsf{c}$ *alors il existe* $\mathsf{d} \in \mathsf{A}$ *tels que* $\mathsf{a} \to^* \mathsf{d} \leftarrow^* \mathsf{c}$*,*
- *est* localement confluent *si pour tout* $\mathsf{a}, \mathsf{b}, \mathsf{c} \in \mathsf{A}$*, si* $\mathsf{a} \leftarrow \mathsf{b} \to \mathsf{c}$ *alors il existe* $\mathsf{d} \in \mathsf{A}$ *tels que* $\mathsf{a} \to^* \mathsf{d} \leftarrow^* \mathsf{c}$*,*
- *vérifie la* propriété du diamant *si pour tout* $\mathsf{a}, \mathsf{b}, \mathsf{c} \in \mathsf{A}$*, si* $\mathsf{a} \leftarrow \mathsf{b} \to \mathsf{c}$ *alors il existe* $\mathsf{d} \in \mathsf{A}$ *tels que* $\mathsf{a} \to \mathsf{d} \leftarrow \mathsf{c}$*.*

Ces différentes formes de confluence sont dépeintes sur la figure 2.1.

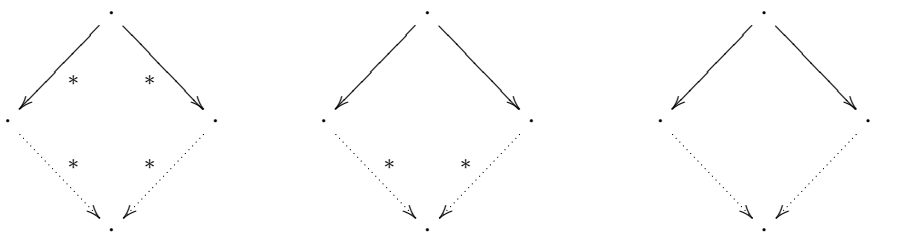


FIG. 2.1: Confluence

Il est clair que la propriété du diamant implique la confluence, qui à son tour implique la confluence locale. Il est bien connu que la confluence locale n'implique pas, en général, la confluence.

Lemme 2.1.9 (Lemme de Newman) *Un ARS fortement normalisant est confluent si et seulement s'il est localement confluent.*

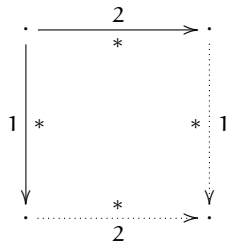
Ceci est faux si on remplace « fortement normalisant » par « faiblement normalisant ».

Proposition 2.1.10 (Propriété de Church-Rosser) *Un ARS* $(\mathsf{A}, \to)$ *est confluent si et seulement s'il vérifie la* propriété de Church-Rosser *:* $\mathsf{a} \leftrightarrow^* \mathsf{b}$ *implique* $\mathsf{a} \downarrow \mathsf{b}$*.*

Définition 2.1.11 (Consistance) *Un ARS* $(\mathsf{A}, \to)$ *est dit* consistant *s'il existe deux éléments* $\mathsf{a}, \mathsf{b} \in \mathsf{A}$ *tels que* $\mathsf{a} \not\leftrightarrow^* \mathsf{b}$*.*

Remarque 2.1.12 *Il est clair qu'un ARS confluent qui a au moins deux formes normales distinctes est consistant.*

Définition 2.1.13 (Commutation) *Deux ARS* $(A, \to_1)$ *et* $(A, \to_2)$ commutent *si pour tout* $a, b, c \in A$ *tels que* $a \leftarrow_1^* b \to_2^* c$*, il existe* $d \in A$ *tel que* $a \to_2^* d \leftarrow_1^* c$*. En image :*



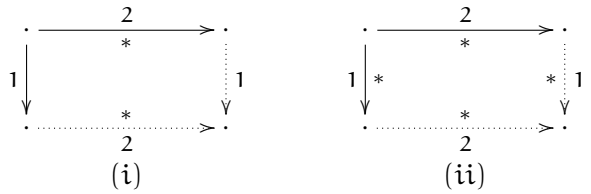
La commutation de relations permet d'obtenir la confluence de l'union de relations de réécriture.

Théorème 2.1.14 (Lemme de Hindley-Rosen — Théorème 3.5 dans [Ros73]) *Soit* $(A, \to_i)_{i \in I}$ *une famille d'ARS telle que* $\to_i$ *et* $\to_j$ *commutent pour tout* $i, j \in I$*. Alors, l'ARS* $(A, \bigcup_{i \in I} \to_i)$ *est confluent.*

Voici un lemme simple et particulièrement utile pour montrer la commutation de deux relations

Lemme 2.1.15 *Soient deux ARS* $(A, \to_1)$ *et* $(A, \to_2)$ *tels que pour tout* $a, b, c \in A$*, si* $a \leftarrow_1 b \to_2 c$ *alors il existe* $d \in A$ *tel que* $a \to_2^* d \leftarrow_1 c$*. Alors* $(A, \to_1)$ *et* $(A, \to_2)$ *commutent. En image :*

Preuve. Par induction sur $\to_2^*$ on montre (i), et on en déduit (ii) par induction sur $\to_1^*$.



□

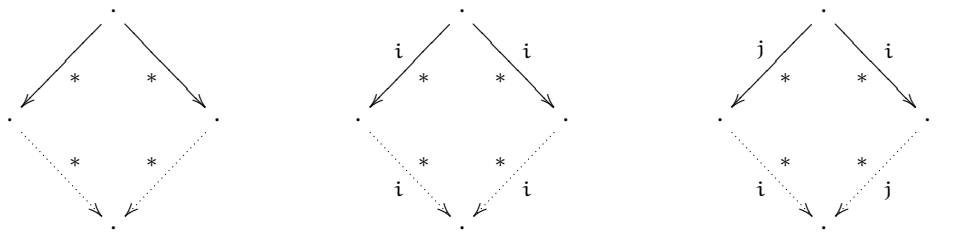
La définition suivante rappelle la notion d'ARS *stratifié*, qui est très utile pour traiter la réécriture conditionnelle (présentée au chapitre 4).

Définition 2.1.16 (Stratification) *Soit un ARS* $(A, \to)$ *et* $(A, \to_i)_{i \in I}$ *une famille d'ARS. Si* $\to = \bigcup\{\to_i \mid i \in I\}$*, on dit que* $(A, \to)$ *est* stratifié *et que* $(A, \to_i)_{i \in I}$ *est une* stratification *de* $(A, \to)$*.*

Les stratifications donnent lieu à des notions de confluence plus fines que celles de la définition 2.1.8.

Définition 2.1.17 (Confluences stratifiées) *Soit un ARS* $(A, \to)$ *et sa stratification* $(A, \to_i)_{i \in I}$. *On dit que* $(A, \to)$ *est*
- confluent par niveaux *lorsque* $(A, \to_i)$ *est confluent pour tout* $i \in I$,
- confluent en profondeur[1] *lorsque pour tout* $i, j \in I$, $\to_i$ *et* $\to_j$ *commutent.*

Les notions de confluence, confluence par niveaux et confluence profondes sont représentées figure. 2.2.



Confluence — Confluence par niveaux — Confluence profonde

FIG. 2.2: Confluences pour $\to = \bigcup\{\to_i \mid i \in I\}$

2.2 Relations de réécriture et congruences

La définition de la notion de relation de réécriture et de congruence dépend de l'ensemble des termes considéré. Pour cela, nous utilisons les notions de contexte et de substitution spécialisées à un sous-ensemble $L \subseteq \Lambda(\Sigma)$ donné, définies en 1.2.19 et 1.3.11 respectivement.

Définition 2.2.1 (Relations de réécriture) *Soit* $L \subseteq \Lambda(\Sigma)$.
- *Une relation binaire* R *sur* L *est une* congruence sur L *si pour tout* $t, u \in L$, $t \mathrel{R} u$ *implique* $C[t] \mathrel{R} C[u]$ *pour tout contexte* $C[\,] : L \to L$.
- *Une relation binaire* R *sur* L *est* stable par substitution sur L *si pour toute substitution* $\sigma : \mathcal{X} \to L$, *pour tout* $t, u \in L$, $t \mathrel{R} u$ *implique* $t\sigma \mathrel{R} u\sigma$.
- *Une relation binaire* R *sur* L *est une* relation de réécriture sur L *si c'est une congruence sur* L *qui est stable par substitution sur* L.

Une relation qui est une congruence est aussi dite « stable par contexte ».

Remarque 2.2.2

[1] *shallow confluence* en Anglais

— *Les relations de réécriture sur* $\Lambda(\Sigma)$, $\mathcal{T}er(\Sigma,\mathcal{X})$ *et* $\mathcal{T}er(\Sigma)$ *correspondent évidemment aux cas* $L=\Lambda(\Sigma)$, $L=\mathcal{T}er(\Sigma,\mathcal{X})$ *et* $L=\mathcal{T}er(\Sigma)$ *respectivement.*
— *La* Σ*-algèbre* $\mathcal{T}er(\Sigma,\mathcal{X})$ *étant libre sur* $\mathcal{X}$*, on peut définir une relation de réécriture sur une* Σ*-algèbre* $\mathcal{T}er(\Sigma_{\mathcal{A}},\mathcal{X}_{\mathcal{A}})$ *libre sur* $\mathcal{X}$ *comme étant une relation binaire* R *sur* $\mathcal{T}er(\Sigma_{\mathcal{A}},\mathcal{X}_{\mathcal{A}})$ *telle que* $\{(i(t),i(u)) \mid t\ R\ u\}$ *est une relation de réécriture sur* $\mathcal{T}er(\Sigma,\mathcal{X})$*, où* i *est l'isomorphisme canonique de* $\mathcal{T}er(\Sigma_{\mathcal{A}},\mathcal{X}_{\mathcal{A}}) \to \mathcal{T}er(\Sigma,\mathcal{X})$.
— *Il est équivalent de dire que* R *est une congruence sur* L *si pour tout* $t,u \in L$, $t\,R\,u$ *implique*
  – *pour tout* $x \in \mathcal{X}$*, si* $\lambda x.t, \lambda x.u \in L$*, alors* $\lambda x.t\ R\ \lambda x.u$,
  – *pour tout* $v \in L$*, si* $tv, uv \in L$*, alors* $tv\ R\ uv$,
  – *pour tout* $v \in L$*, si* $vt, vu \in L$*, alors* $vt\ R\ vu$,
  – *pour tout* $f \in \Sigma_n$*, pour tout* $v_1,\dots,v_n \in L$*, pour tout* $i \in \{1,\dots,n\}$*, si*

$$f(v_1,\dots,v_{i-1},t,v_{i+1},\dots,v_n), f(v_1,\dots,v_{i-1},u,v_{i+1},\dots,v_n) \in L$$

  alors

$$f(v_1,\dots,v_{i-1},t,v_{i+1},\dots,v_n)\ \ R\ \ f(v_1,\dots,v_{i-1},u,v_{i+1},\dots,v_n)\ .$$

Lemme 2.2.3 (Clôture par contexte) *Étant donnée une relation binaire* R *sur* $L \subseteq \Lambda(\Sigma)$*, soit* $\overline{R}$ *la plus petite relation binaire sur* L *satisfaisant aux règles suivantes :*

$$(\mathrm{R})\ \frac{t\ R\ u}{t\ R_{ctx}\ u} \qquad (\textsc{Abs})\ \frac{t\ R_{ctx}\ u}{\lambda x.t\ R_{ctx}\ \lambda x.u}$$

$$(\textsc{App})\ \frac{t_1\ R_{ctx}\ u_1 \qquad t_2\ R_{ctx}\ u_2}{t_1t_2\ R_{ctx}\ u_1u_2} \qquad (\textsc{Symb})\ \frac{t_1\ R_{ctx}\ u_1 \qquad \dots \qquad t_n\ R_{ctx}\ u_n}{f(t_1,\dots,t_n)\ R_{ctx}\ f(u_1,\dots,u_n)}$$

où, dans la règle (Symb)*,* f *est un symbole d'arité* n*. Alors,*

(i) $\overline{R}$ *est la plus petite relation sur* L *qui contient* R *et qui est stable par contexte ;*

(ii) si R *est stable par substitution sur* L*, alors* $\overline{R}$ *est une relation de réécriture sur* L.

2.3 Réduction parallèle

Lorsque l'on travaille sur la commutation de deux relations de réécritures, comme c'est souvent le cas en section 6.1 et au chapitre 7, on a besoin de la stabilité par substitution des relations de réécriture et d'une autre propriété, apparentée, qui correspond plutôt à la réduction *dans* les substitutions :

Stabilité par substitution : $t \to u$ implique $t[v/x] \to u[v/x]$.

Réécriture dans les substitutions : $t \to u$ implique $v[t/x] \to v[u/x]$.

La propriété de « réécriture dans les substitutions » n'est en général pas vérifiée par les relations de réécriture, et ce pour deux raisons :

(i) le terme v peut ne pas contenir la variable x, et dans ce cas on a $v[t/x] = v[u/x]$,
(ii) le terme v peut dupliquer la variable x, et dans ce cas on a $v[t/x] \to^+ v[u/x]$.

La propriété de « réécriture dans les substitutions » est satisfaite par les relations de réécritures réflexives et transitives.

Lemme 2.3.1 *Soit* $\to$ *une relation de réécriture sur* $L \subseteq \Lambda(\Sigma)$ *et* $t, u, v \in L$. *Si* $t \to^* u$ *alors* $v[t/x] \to^* v[u/x]$.

Preuve. Par induction sur v. □

Cependant, les relations transitives ne correspondent pas précisément à cette propriété : elles permettent de rassembler plusieurs pas de réécriture en un seul, mais sans distinguer entre des pas « disjoints », comme c'est le cas dans $v[t/x] \to^+ v[u/x]$ et des pas successifs comme dans la réduction $a \to b \to c$ de la relation de réécriture $\{a \mapsto b, b \mapsto c\}$.

Il existe une notion générique de relation de réécriture qui correspond très bien à la réécriture dans les substitutions. Il s'agit de la *réécriture parallèle*.

Définition 2.3.2 (Relation de réécriture parallèle) *Une relation* $\rhd$ *de réécriture sur* $L \subseteq \Lambda(\Sigma)$ *est dite* parallèle *si*

— *pour tout* $x \in \mathcal{X}$, *si* $x \in L$ *alors* $x \rhd x$,

— *si* $t_1 t_2, u_1 u_2 \in L$ *alors*

$$(t_1 \rhd u_1 \quad \textit{et} \quad t_2 \rhd u_2) \qquad \textit{implique} \qquad t_1 t_2 \rhd u_1 u_2$$

— *si* $f(t_1, \dots, t_n), f(u_1, \dots, u_n) \in L$ *alors*

$$(\forall i \in \{1, \dots, n\}.\ t_i \rhd u_i) \qquad \textit{implique} \qquad f(t_1, \dots, t_n) \rhd f(u_1, \dots, u_n)$$

Remarque 2.3.3 *Les relations de réécriture parallèles sont réflexives.*

Proposition 2.3.4 *Soit* R *une relation binaire sur* $L \subseteq \Lambda(\Sigma)$ *stable par substitution, et* $\to_{\|R}$ *la plus petite relation sur* L *contenant* R *et satisfaisant aux règles suivantes :*

$$(\rhd\textsc{Var})\ \frac{}{x \ \to_{\|R}\ x} \qquad (\rhd\textsc{Abs})\ \frac{t_1 \ \to_{\|R}\ u_1}{\lambda x.t_1 \ \to_{\|R}\ \lambda x.u_1}$$

$$(\rhd\textsc{App})\ \frac{t_1 \ \to_{\|R}\ u_1 \quad t_2 \ \to_{\|R}\ u_2}{t_1 t_2 \ \to_{\|R}\ u_1 u_2} \qquad (\rhd\textsc{Symb})\ \frac{t_1 \ \to_{\|R}\ u_1 \quad \dots \quad t_n \ \to_{\|R}\ u_n}{f(t_1, \dots, t_n) \ \to_{\|R}\ f(u_1, \dots, u_n)}$$

où, dans la règles (Symb), $f \in \Sigma_n$. *Alors,*

(i) la relation $\to_{\|R}$ *est une relation de réécriture parallèle sur* L,

(ii) on a $R \subseteq \to_{\|R} \subseteq R^*$, *(où* R^* *est la clôture réflexive et transitive de* R, *définie en 2.1.4).*

Preuve.

(i) Il suffit de montrer que $\to_{\|R}$ est une relation de réécriture. Elle est stable par contexte par (ii). Il reste à montrer qu'elle est stable par substitution.
Soit σ une L-substitution et $t, u \in L$ tels que $t \to_{\|R} u$. On montre que $t\sigma \to_{\|R} u\sigma$ par induction sur $t \to_{\|R} u$. Le cas de la règle ($\rhd$Var) suit de la réflexivité de $\to_{\|R}$, et celui des règles ($\rhd$Abs), ($\rhd$App) et ($\rhd$Symb) de l'hypothèse d'induction. Dans le cas où $t\ R\ u$, on conclut grâce à l'hypothèse que R est stable par substitution.

(ii) On a $R \subseteq \to_{\|R}$ par définition et on montre $\to_{\|R} \subseteq R^*$ par induction sur les règles de la définition 2.3.2. □

Ainsi, on peut définir la *clôture parallèle* d'une relation sur $L \subseteq \Lambda(\Sigma)$.

Définition 2.3.5 *Soit* R *une relation binaire sur* $L \subseteq \Lambda(\Sigma)$. *La* clôture parallèle *de* R *sur* L *est la relation* $\to_{\|R_{subst}}$ *où* R_{subst} *est la clôture de* R *par* L*-substitutions.*

Lorsque le contexte le permet, on écrit $\to_{\|R}$ pour désigner $\to_{\|R_{subst}}$. La propriété de réécriture dans les substitutions est facilement obtenue. On la présente ici dans une forme un peu plus générale, qui sera très utile par la suite. Rappelons que d'après la définition 1.3.12, si R est une relation binaire sur $\Lambda(\Sigma)$ et si σ, σ' sont deux substitutions, alors $\sigma \, R \, \sigma'$ signifie que $\mathcal{D}om(\sigma) = \mathcal{D}om(\sigma')$ et que $\sigma(x) \, R \, \sigma'(x)$ pour tout $x \in \mathcal{D}om(\sigma)$.

Lemme 2.3.6 *Soit* R *une relation binaire sur* $L \subseteq \Lambda(\Sigma)$. *Alors,* $\sigma \, R \, \sigma'$ *implique* $t\sigma \to_{\|R} t\sigma'$ *pour toute* L*-substitution* σ *et pour tout* $t \in L$.

Preuve. Par induction sur t. □

2.4 Systèmes de réécriture

Dans cette section, nous rappelons la définition de système de réécriture de termes. Notre définition est un peu plus générale que la définition habituelle. Elle permet de rendre compte harmonieusement de la réécriture usuelle, de la réécriture conditionnelle et du λ-calcul. Nous reprenons la terminologie de [Ter03].

Définition 2.4.1 (Systèmes de réécriture de termes) *Un* système de réécriture de termes *(ou TRS) sur* $L \subseteq \Lambda(\Sigma)$ *est une relation binaire* $\mathcal{R}$ *sur* L *telle que* $l \, \mathcal{R} \, r$ *implique* $l \notin \mathcal{X}$ *et* $FV(r) \subseteq FV(l)$.

Une paire $(l, r) \in \mathcal{R}$ *est une* règle de réécriture *de* $\mathcal{R}$, *aussi notée* $l \mapsto_{\mathcal{R}} r$.

À un TRS $(L, \mathcal{R})$ *on associe l'ARS* $(L, \to_{\mathcal{R}})$ *où* $\to_{\mathcal{R}}$ *est la plus petite relation de réécriture sur* L *qui contient* $\mathcal{R}$.

Remarque 2.4.2

— *Comme pour les relations de réécriture (voir remarque. 2.2.2), la notion de TRS a pour cas particulier les TRS sur* $\Lambda(\Sigma)$, $\mathcal{T}er(\Sigma, \mathcal{X})$ *et* $\mathcal{T}er(\Sigma)$.

— *De même, un TRS sur une* Σ*-algèbre libre* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *est une relation binaire* $\mathcal{R}$ *sur* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *telle que* $\{(i(t), i(u)) \mid t \, \mathcal{R} \, u\}$ *est un TRS sur* $\mathcal{T}er(\Sigma, \mathcal{X})$, *où* i *est l'isomorphisme canonique de* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}}) \to \mathcal{T}er(\Sigma, \mathcal{X})$.

— *Si* $(\Lambda(\Sigma), \mathcal{R})$ *est un système de réécriture où* $\mathcal{R}$ *est stable par substitution, alors par le lemme 2.2.3,* $(\Lambda(\Sigma), \overline{\mathcal{R}})$ *est une relation de réécriture (où* $\overline{\mathcal{R}}$ *est la clôture de* $\mathcal{R}$ *par contexte, définie en 2.2.3).*

— *On écrit* $(L, \mapsto_{\mathcal{R}\mathcal{R}'})$ *pour l'union des TRS* $(L, \mapsto_{\mathcal{R}})$ *et* $(L, \mapsto'_{\mathcal{R}})$, *et* $(L, \to_{\mathcal{R}\cup\mathcal{R}'})$ *pour la relation de réécriture qui en est issue.*

— *Selon les définitions 2.4.1 et 2.3.2, les relations de réécriture parallèles ne forment pas des TRS car ce sont des relations réflexives, qui contiennent donc* $\{(x, x) \mid x \in \mathcal{X}\}$.

Exemple 2.4.3

— *Si* $(\Lambda(\Sigma), \mathcal{R})$ *est un TRS, alors il en est de même pour* $(\Lambda(\Sigma), \to_{\mathcal{R}})$.
— *La notion de système de réécriture correspond à l'idée de règle de réécriture :* $\mathcal{R}$ *est vu comme un ensemble de règles* $l \mapsto_{\mathcal{R}} r$.

Proposition 2.4.4 (Stabilité par renommage) *Un* renommage *est une bijection sur* $\mathcal{X}$. *Si* $\mathcal{R}$ *est un TRS sur* $L \subseteq \Lambda(\Sigma)$ *et* ρ *est un renommage tel que* $\rho(FV(\mathcal{R})) \subseteq L$, *alors* $\to_{\mathcal{R}} = \to_{\mathcal{R}\rho}$, *où* $\mathcal{R}\rho =_{def} \{(l\rho, r\rho) \mid l \mapsto_{\mathcal{R}} r\}$.

Chapitre 3

Réécriture et lambda-calcul

Dans ce chapitre, nous articulons les objets d'étude centraux de cette thèse : la réécriture, le λ-calcul et leur combinaison.

Nous partons de la réécriture du premier ordre sur $\mathcal{T}er(\Sigma, \mathcal{X})$ et de son extension aux termes applicatifs $\mathcal{T}er(\Sigma_{App}, \mathcal{X})$. Nous en considérons deux variantes : la réécriture algébrique, qui est l'extension naturelle de la notion de premier ordre « sémantique » aux termes applicatifs, et la réécriture applicative à droite, qui autorise n'importe quel terme de $\mathcal{T}er(\Sigma_{App}, \mathcal{X})$ dans les membres droits de règles. Nous donnons quelques exemples de systèmes algébriques, comme les paires surjectives, et de systèmes applicatifs, comme l'itérateur et le récurseur sur les entiers.

Nous passons ensuite au λ-calcul. Nous commençons par le cas non typé puis présentons deux calculs typés : le λ-calcul simplement typé et le système F. Ce dernier système permet de coder certaines des fonctions présentées dans le cadre de la réécriture, par exemple le schéma d'itération sur les entiers de Church. Cependant, les possibilités de codage du λ-calcul ne semblent pas suffisantes, en particulier pour les paires surjectives et pour le schéma de récursion sur les entiers. Il faudrait donc étendre la β-réduction. Pour mieux comprendre quelles sont les possibilités d'extensions du λ-calcul, un résultat important est le théorème de Böhm : sur les λ-termes purs et β-normalisants, il n'y a pas d'extension stricte consistante de la βη-conversion.

Cela suggère d'accompagner les extensions de la β(η)-réduction par des extensions de la *syntaxe* du λ-calcul. Une possibilité, à la laquelle est consacrée cette thèse, est d'étendre la β-réduction par des règles de réécriture opérant sur des λ-termes étendus avec symboles algébriques, c'est-à-dire sur des termes de $\Lambda(\Sigma)$. Nous voyons ensuite comment adapter le λ-calcul typé à la réécriture, ce qui nous permettra de considérer brièvement les combinaisons du λ-calcul et de la réécriture pour quelques exemples, dont les paires surjectives et le schéma de récursion sur les entiers.

3.1 Réécriture au premier ordre

Dans cette section, nous rappelons les définitions concernant la réécriture *au* premier ordre : termes algébriques, systèmes de réécriture algébriques et systèmes de réécriture applicatifs. Nous donnons aussi quelques exemples de tels systèmes. La réécriture *du* premier ordre usuelle est la réécriture sur les termes du premier ordre.

Définition 3.1.1 (TRS du premier ordre) *Un TRS du premier ordre est un TRS sur* $\mathcal{T}er(\Sigma, \mathcal{X})$.

Il est souvent utile d'isoler les symboles qui sont à la tête de règles de réécriture. Ils sont dits « définis », alors que les autres symboles sont des « constructeurs ».

Définition 3.1.2 (Symboles définis et constructeurs) *Soit $\mathcal{R}$ un TRS du premier ordre sur $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$. Un symbole $f \in \Sigma$ est* défini dans $\mathcal{R}$ *s'il existe une règle $f(\vec{l}) \mapsto_{\mathcal{R}} r$. Les symboles qui ne sont pas définis dans $\mathcal{R}$ sont des* constructeurs de $\mathcal{R}$.

Voici une définition des booléens et des paires comme TRS du premier ordre.

Exemple 3.1.3 (Booléens) *Les booléens sont définis par $\Sigma_{\mathsf{Bool}} =_{def} \{\mathsf{true}, \mathsf{false}\}$. Il sont éliminés par l'itérateur $\mathsf{iter}_{\mathsf{Bool}}(\_,\_,\_)$, qui est défini par le système de réécriture*

$$\mathsf{iter}_{\mathsf{Bool}}(x, y, \mathsf{true}) \quad \mapsto_{\mathsf{iter}_{\mathsf{Bool}}} \quad x \qquad\qquad \mathsf{iter}_{\mathsf{Bool}}(x, y, \mathsf{false}) \quad \mapsto_{\mathsf{iter}_{\mathsf{Bool}}} \quad y \ .$$

Le terme $\mathsf{iter}_{\mathsf{Bool}}(u, v, b)$ correspond à la construction « if b then u else v » des langages de programmation. On peut la définir par les règles suivantes :

$$\mathsf{ite}(\mathsf{true}, x, y) \quad \mapsto_{\mathsf{ite}} \quad x \qquad \textit{et} \qquad \mathsf{ite}(\mathsf{false}, x, y) \quad \mapsto_{\mathsf{ite}} \quad y \ .$$

Exemple 3.1.4 (Paires) *Les paires sont construites par le symbole binaire $(\_,\_)$ et éliminées par les symboles unaires $\pi_1\_$ et $\pi_2\_$ définis par le système de réécriture*

$$\pi_1(x, y) \quad \mapsto_{\pi} \quad x \qquad\qquad \pi_2(x, y) \quad \mapsto_{\pi} \quad y \ .$$

Parfois, il est aussi intéressant d'ajouter la règle suivante, dite de surjectivité, qui implante une forme d'extensionalité pour le constructeur $(\_,\_)$:

$$(\pi_1 t, \pi_2 t) \quad \mapsto_{\mathsf{SP}} \quad t \ .$$

Les termes du premier ordre contiennent aussi les termes applicatifs. Ainsi, un TRS sur $\mathcal{T}\mathrm{er}(\Sigma_{\mathrm{App}}, \mathcal{X})$ est aussi un TRS du premier ordre. Rappelons que les termes applicatifs sur une signature Σ sont générés par la grammaire :

$$t, u \in \mathcal{T}\mathrm{er}(\Sigma_{\mathrm{App}}, \mathcal{X}) \quad ::= \quad x \ \mid \ f(t_1, \ldots, t_n) \ \mid \ t\,u$$

où $x \in \mathcal{X}$ et $f \in \Sigma_n$.

Exemple 3.1.5 (Fonction identité) *Voici une définition de la fonction identité sur les termes applicatifs :*

$$\mathsf{id} \cdot x \quad \mapsto_{\mathsf{id}} \quad x \ .$$

On a $\mathsf{id} \cdot t \to_{\mathsf{id}} t$ pour tout $t \in \Lambda(\Sigma)$.

Dans l'exemple 3.1.5, on peut voir la variable x comme étant une variable « du premier ordre ». Mais avec un symbole d'application, on peut aussi avoir des variables ayant « une sémantique d'ordre supérieur », c'est-à-dire qui ont « vocation » être substituées par des « fonctions ».

Exemple 3.1.6 (Traitement des erreurs) *On représente les « erreurs » par un symbole* err *défini par la règle*

$$\mathsf{err} \cdot x \quad \mapsto_{\mathsf{err}} \quad \mathsf{err}\ .$$

On peut lui adjoindre la règle suivante, qui force les fonctions à être « strictes » vis-à-vis de la propagation d'erreurs :

$$x \cdot \mathsf{err} \quad \mapsto_{\mathsf{strict}} \quad \mathsf{err}\ .$$

Dans ce système, la variable x *a une « sémantique d'ordre supérieur » : on peut avoir*

$$\mathsf{err} \quad \leftarrow_{\mathsf{id}} \quad \mathsf{id} \cdot \mathsf{err} \quad \rightarrow_{\mathsf{strict}} \quad \mathsf{err}\ .$$

Ainsi, lorsque l'on dispose d'un symbole d'application, les notions « syntaxiques » et « sémantiques » de premier ordre ne coïncident plus. Dans le terme $x \cdot \mathsf{err}$, la variable x a une « sémantique d'ordre supérieur » (nous disons qu'elle est *active*), alors que le terme $x \cdot \mathsf{err}$ est syntaxiquement un terme du premier ordre : c'est un élément de $\mathcal{T}er(\Sigma_{App}, \mathcal{X})$.

La notion « sémantique » naturelle de termes *du* premier ordre nous semble être celle de terme algébrique, c'est-à-dire de terme applicatif sans variable active. Nous parlons de termes *au* premier ordre pour désigner les éléments de $\mathcal{T}er(\Sigma_{App}, \mathcal{X})$. Ils correspondent à la notion syntaxique de premier ordre.

Définition 3.1.7 (Termes algébriques)

— *Une variable* x *est* active *dans un terme* t *si* t *a un sous-terme la forme* $x\,u$.
— *Un terme applicatif est* algébrique *s'il ne contient pas de variable active.*

Les termes algébriques de $\Lambda(\Sigma)$ ont une forme très agréable, ils sont construits selon la grammaire suivante :

$$t \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n)t_{n+1} \ldots t_{n+m}$$

où $x \in \mathcal{X}$, $f \in \Sigma_n$ et $m \geq 0$. Les membres gauches des règles des systèmes de réécriture que nous utilisons sont très souvent algébriques (bien que ce ne soit pas le cas de $\mapsto_{\mathsf{strict}}$).

Définition 3.1.8 (Systèmes de réécriture applicatifs et algébriques) *Soit* $\mathcal{R}$ *un TRS sur* $L \subseteq \Lambda(\Sigma)$.

— *On dit que* $\mathcal{R}$ *est* algébrique à gauche *(resp.* à droite*) si* l *(resp.* r*) est algébrique pour tout* $l \mapsto_{\mathcal{R}} r$.
— *On dit que* $\mathcal{R}$ *est* algébrique *s'il est algébrique à gauche et à droite.*
— *On dit que* $\mathcal{R}$ *est* applicatif à gauche *(resp.* à droite*) si* l *(resp.* r*) est applicatif pour tout* $l \mapsto_{\mathcal{R}} r$.

Par exemple, le système $\mapsto_{\mathsf{err}}$ de l'exemple 3.1.6 est algébrique alors que le système $\mapsto_{\mathsf{strict}}$ ne l'est pas.

Un TRS algébrique sur $\Lambda(\Sigma)$ est donc un TRS sur $\mathcal{T}er(\Sigma_{App}, \mathcal{X})$. Lorsque que l'on considère la relation de réécriture sur $\Lambda(\Sigma)$ issue d'un TRS applicatif, on obtient un TRS qui n'est plus applicatif, mais qui conserve certaines propriétés intéressantes provenant de sa parenté applicative.

Définition 3.1.9 Un TRS $\mathcal{R}$ est issu d'un TRS $\mathcal{R}'$ *si pour tout* $l \mapsto_{\mathcal{R}} r$ *il existe une substitution* σ *et deux termes* $l' \mapsto_{\mathcal{R}'} r'$ *tels que* $l = l'\sigma$ *et* $r = r'\sigma$.

L'intérêt des TRS issus de TRS applicatifs est qu'ils préservent « l'applicativité » des termes.

Proposition 3.1.10 *Si* $\mathcal{R}$ *est un TRS issu d'un TRS applicatif à droite, alors*

$$(t \in \mathcal{T}er(\Sigma_{App}, \mathcal{X}) \quad \wedge \quad t \to_{\mathcal{R}} u) \qquad \Longrightarrow \qquad u \in \mathcal{T}er(\Sigma_{App}, \mathcal{X}) \ .$$

Il est possible d'avoir des fonctions définies en utilisant à la fois le symbole d'application et des symboles de fonction d'arité non nulle. C'est ainsi que nous définissons les entiers et les listes.

Exemple 3.1.11 (Entiers) *Les entiers unaires de Peano sont construits par le symbole* 0 *et le symbole unaire* $S(\_)$. *On pose* $\Sigma_{Nat} =_{def} \{0, S(\_)\}$.

Par exemple, la fonction « strictement supérieur », des entiers vers les booléens, peut être représentée par le symbole $>$ *et les règles suivantes :*

$$\begin{array}{lll} > \ 0 \ y & \mapsto_{>} & \mathsf{false} \\ > \ S(x) \ 0 & \mapsto_{>} & \mathsf{true} \\ > \ S(x) \ S(y) & \mapsto_{>} & > \ x \ y \ . \end{array}$$

On peut définir la soustraction entière sur la signature $\Sigma_{minus} =_{def} \Sigma_{Nat} \cup \{\mathsf{minus}\}$ *par les règles suivantes :*

$$\begin{array}{lll} \mathsf{minus} \ x \ 0 & \mapsto_{\mathsf{minus}} & x \\ \mathsf{minus} \ 0 \ y & \mapsto_{\mathsf{minus}} & 0 \\ \mathsf{minus} \ S(x) \ S(y) & \mapsto_{\mathsf{minus}} & \mathsf{minus} \ x \ y \ . \end{array}$$

Elles peuvent être complétées par les règles non-linéaires suivantes :

$$\mathsf{minus} \ x \ x \quad \mapsto_{\mathsf{minus}} \quad 0 \qquad\qquad \mathsf{minus} \ S(x) \ x \quad \mapsto_{\mathsf{minus}} \quad S(0) \ .$$

Exemple 3.1.12 (Listes) *Les listes que nous utilisons sont construites sur la signature* $\Sigma_{List} =_{def} \{[\,], (\_ :: \_)\}$, *où* $[\,]$ *représente la liste vide et* $(x :: xs)$ *représente la liste* xs *sur laquelle est empilé l'élément* x. *Voici quelques opérations de base :*

$$\begin{array}{lll} \mathsf{car} \ (x :: l) & \mapsto_{\mathsf{car}} & x \\ \mathsf{car} \ [\,] & \mapsto_{\mathsf{car}} & \mathsf{err} \end{array} \qquad\qquad \begin{array}{lll} \mathsf{cdr} \ (x :: l) & \mapsto_{\mathsf{cdr}} & l \\ \mathsf{cdr} \ [\,] & \mapsto_{\mathsf{cdr}} & \mathsf{err} \end{array}$$

$$\begin{array}{lll} \mathsf{get} \ l \ 0 & \mapsto_{\mathsf{get}} & \mathsf{car} \ l \\ \mathsf{get} \ l \ S(n) & \mapsto_{\mathsf{get}} & \mathsf{get} \ (\mathsf{cdr} \ l) \ n \end{array} \qquad\qquad \begin{array}{lll} \mathsf{length} \ [\,] & \mapsto_{\mathsf{length}} & 0 \\ \mathsf{length} \ (x :: l) & \mapsto_{\mathsf{length}} & S(\mathsf{length} \ l) \end{array}$$

Cependant, dans beaucoup d'utilisations des termes applicatifs, l'application est le seul symbole d'arité non nulle. Ce sont des systèmes *curryfiés*.

Exemple 3.1.13 (Curryfication) *On peut faire correspondre aux termes sur* Σ_{Nat} *et* Σ_{List} *respectivement un sous ensemble des termes construits sur les signatures* $(\Sigma_{\mathsf{Nat}\mathcal{C}})_{\mathrm{App}}$ *et* $(\Sigma_{\mathsf{List}\mathcal{C}})_{\mathrm{App}}$*, où*

$$\Sigma_{\mathsf{Nat}\mathcal{C}} \quad =_{def} \quad \{0_{\mathcal{C}}, \mathsf{S}_{\mathcal{C}}\} \qquad et \qquad \Sigma_{\mathsf{List}\mathcal{C}} \quad =_{def} \quad \{\mathsf{nil}, \mathsf{cons}\}\ .$$

Par exemple,

$$\begin{array}{rrll} & \mathsf{S}(0) & \textit{correspond à} & \mathsf{S}_{\mathcal{C}} \cdot 0_{\mathcal{C}} \\ et & (\mathsf{S}(0) :: [\,]) & \textit{correspond à} & (\mathsf{cons} \cdot (\mathsf{S}_{\mathcal{C}} \cdot 0_{\mathcal{C}})) \cdot \mathsf{nil} \end{array}$$

La traduction canonique de $\mathcal{T}er(\Sigma_{\mathsf{Nat}} \cup \Sigma_{\mathsf{List}}, \mathcal{X})$ vers $\mathcal{T}er((\Sigma_{\mathsf{Nat}\mathcal{C}} \cup \Sigma_{\mathsf{List}\mathcal{C}})_{\mathrm{App}}, \mathcal{X})$ s'appelle *curryfication*. Nous l'étudions de façon plus précise au chapitre 8.

Remarque 3.1.14 (Fonctions strictes) *Le système* $\mapsto_{\mathsf{err}\cup\mathsf{strict}}$ *semble avoir des caractéristiques intéressantes pour les systèmes curryfiés : les fonctions définies par des systèmes curryfiés confluents dont la combinaison avec* $\mapsto_{\mathsf{err}\cup\mathsf{strict}}$ *reste confluente peuvent être vues comme des fonctions strictes vis-à-vis de la propagation des erreurs.*

C'est le cas, pas exemple, de la version curryfiée du système présenté à l'exemple 3.1.3 :

$$\mathsf{ite}_{\mathcal{C}} \cdot \mathsf{true} \cdot x \cdot y \quad \mapsto_{\mathsf{ite}_{\mathcal{C}}} \quad x \qquad\qquad \mathsf{ite}_{\mathcal{C}} \cdot \mathsf{true} \cdot x \cdot y \quad \mapsto_{\mathsf{ite}_{\mathcal{C}}} \quad y\ .$$

Un exemple de fonction non stricte est la fonction eat *définie par la règle*

$$\mathsf{eat} \cdot x \quad \mapsto_{\mathsf{eat}} \quad \mathsf{eat}\ .$$

On alors un pic injoignable :

$$\mathsf{err} \quad \leftarrow_{\mathsf{strict}} \quad \mathsf{eat} \cdot \mathsf{err} \quad \rightarrow_{\mathsf{eat}} \quad \mathsf{eat}\ .$$

Une telle caractérisation des fonctions strictes serait plus difficile avec des systèmes du premier ordre. En effet, les règles prenant en compte err *dépendraient de la signature : on aurait une règle*

$$\mathsf{f}(x_1, \ldots, \mathsf{err}, \ldots, x_n) \quad \mapsto \quad \mathsf{err}$$

pour chaque $\mathsf{f} \in \Sigma_n$ *et chaque* $i \in \{1, \ldots, n\}$*.*

Ainsi, le symbole d'application donne aux systèmes curryfiés, ou de manière plus générale aux systèmes au premier ordre, une forme différente de celle des systèmes du premier ordre. En particulier, dans le système $\mapsto_{\mathsf{id}}$ de l'exemple 3.1.5, le symbole d'application est un symbole défini au sens de la définition 3.1.2, alors que le symbole id ne l'est pas. Une notion plus générale de symbole défini semble nécessaire.

Définition 3.1.15 (Symboles définis et constructeurs algébriques) *Soit* $\mathcal{R}$ *un système algébrique à gauche sur* $\Lambda(\Sigma)$*. Un symbole* $\mathsf{f} \in \Sigma$ *est* algébriquement défini par $\mathcal{R}$ *s'il existe une règle de la forme* $\mathsf{f}(\vec{l})\ \vec{l'} \mapsto_{\mathcal{R}} r$*. Les symboles qui ne sont pas algébriquement définis dans* $\mathcal{R}$ *sont des* constructeurs algébriques de $\mathcal{R}$*.*

La notion de règle effondrante joue un rôle important dans l'étude de la combinaison de relations de réécriture.

Définition 3.1.16 (Règle effondrante) *Une règle de réécriture* $l \mapsto r$ *est* effondrante *si* r *est une variable.*

Voyons enfin un exemple utilisant des variables actives dans les membres droits.

Exemple 3.1.17 (Itérateurs et récurseurs pour les entiers) *Il existe des éliminateurs canoniques pour les entiers, ce sont l'itérateur et le récurseur. Ces systèmes de réécriture sont applicatifs mais pas algébriques : ils ont une « sémantique d'ordre supérieur » tout en étant, syntaxiquement, des systèmes du premier ordre.*

*(i) L'*itérateur iter *est défini par les règles suivantes :*

$$\mathsf{iter}\ u\ v\ 0 \quad \mapsto_{\mathsf{iter}} \quad u \qquad\qquad \mathsf{iter}\ u\ v\ \mathsf{S}(n) \quad \mapsto_{\mathsf{iter}} \quad v\ (\mathsf{iter}\ u\ v\ n)\ .$$

Notons que dans la seconde règle, le terme v *n'a pas d'accès direct au terme* n *qui est en train de se faire éliminer. De ce fait,* iter *ne permet pas de coder certains algorithmes, comme par exemple le calcul du prédécesseur en temps constant.*

(ii) Le récurseur rec *est défini par les règles suivantes :*

$$\mathsf{rec}\ u\ v\ 0 \quad \mapsto_{\mathsf{rec}} \quad u \qquad\qquad \mathsf{rec}\ u\ v\ \mathsf{S}(n) \quad \mapsto_{\mathsf{rec}} \quad v\ (\mathsf{rec}\ u\ v\ n)\ n\ .$$

C'est le système de réécriture sous-jacent au système T de Gödel (voir 3.7.2). Les fonctions calculées par les systèmes rec *et* iter *sont les mêmes, bien que certains algorithmes exprimables dans* rec *ne le soient pas dans* iter*, comme par exemple le calcul du prédécesseur en temps constant.*

3.2 Lambda-calcul non typé

Un formalisme naturel pour pouvoir manipuler des variables actives et des fonctions d'ordre supérieur est le λ-calcul. Dans cette section, nous présentons certaines notions importantes dans le cas non typé. Commençons par rappeler la définition de la β-réduction et certaines de ses propriétés.

Définition 3.2.1 (Béta-réduction) *La* notion de β-réduction *est la plus petite relation* $\mapsto_\beta$ *telle que pour tout* $t, u \in \Lambda(\Sigma)$*,*

$$(\lambda x.t)u \quad \mapsto_\beta \quad t[u/x]\ .$$

La relation de β-réduction*, notée* $\to_\beta$*, est la clôture par contexte de* $\mapsto_\beta$*.*

Proposition 3.2.2 *La relation de* β*-réduction est une relation de réécriture.*

Preuve. Grâce au lemme 2.2.3, il suffit de montrer que pour tout $t, u \in \Lambda(\Sigma)$, pour toute substitution σ, $((\lambda x.t)u)\sigma \mapsto_\beta (t[u/x])\sigma$.

Comme noté en 1.3.10, on peut supposer que $x \notin \mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$, c'est à dire que $\mathcal{D}om([u/x])$ est disjoint de $(\mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma))$. Par le lemme 1.3.9, on a alors $(t[u/x])\sigma = t\sigma(\sigma \circ [u/x]) = t\sigma[u\sigma/x]$. D'autre part, comme $x \notin \mathcal{D}om(\sigma) \cup \mathsf{FV}(\sigma)$, on a $((\lambda x.t)u)\sigma = (\lambda x.t\sigma)(u\sigma)$, ce qui conclut la démonstration. □

Remarque 3.2.3 *Il s'en suit que* $(\Lambda(\Sigma), \to_\beta)$ *est la relation de réécriture issue du TRS* $(\Lambda(\Sigma), \mapsto_\beta)$*. C'est donc aussi un TRS.*

Une propriété importante de la β-réduction est sa confluence.

Théorème 3.2.4 *La relation* $\to_\beta$ *est confluente.*

Une des preuves de ce résultat est présentée en 5.1.2.

Exemple 3.2.5 (Fonction identité) *Un des termes les plus simples que l'on peut former est la fonction identité :*

$$\mathrm{Id} \quad =_{def} \quad \lambda x.x \ .$$

Pour tout $t \in \Lambda(\Sigma)$*, on a* $\mathrm{Id}\ t \to_\beta t$.

On écrit $\lambda\_.t$ pour $\lambda x.t$ avec $x \notin \mathrm{FV}(t)$. D'autre part, $\lambda x, y.t$ et $\lambda x\, y.t$ désignent $\lambda x.\lambda y.t$.

Le λ-calcul dispose d'un moyen d'évaluation privilégié : la stratégie de réduction dite « normale », qui trouve toujours la forme β-normale d'un terme lorsqu'elle existe. Cette stratégie est basée sur la notion de réduction de tête. Dans cette section, nous rappelons des définitions et résultats sur ces notions.

Lemme 3.2.6 (Notation de Wadsworth) *Tout terme* $t \in \Lambda(\Sigma)$ *s'écrit de manière unique sous une des trois formes suivantes :*

$$\lambda x_1 \dots x_p.(\lambda x.t)u t_1 \dots t_n \tag{a}$$
$$\lambda x_1 \dots x_p.x\ t_1 \dots t_n \tag{b}$$
$$\lambda x_1 \dots x_p.f(t_1, \dots, t_n)\ t_{n+1} \dots t_{n+m} \tag{c}$$

où $n, m, p \geq 0$*,* $x \in \mathcal{X}$ *et* $f \in \Sigma_n$.

Dans les cas (b) et (c), on dit que t *est en* forme β-normale de tête.

Définition 3.2.7 (Réduction de tête, réduction normale)

— *La* réduction faible de tête *est la plus petite relation* $\to_{wh_\beta}$ *telle que pour tout* $n \geq 0$*, pour tout* $t, u, t_1, \dots, t_n \in \Lambda(\Sigma)$ *on ait,*

$$(\lambda x.t)u t_1 \dots t_n \quad \to_{wh_\beta} \quad t[u/x]t_1 \dots t_n \ .$$

— *Le* réduction de tête *est la plus petite relation* $\to_h$ *telle que pour tout* $t, u \in \Lambda(\Sigma)$ *on ait,*

$$\lambda\vec{x}.t \quad \to_h \quad \lambda\vec{x}.u \qquad si \qquad t \quad \to_{wh_\beta} \quad u \ .$$

Un terme est en forme normale de tête *s'il est en forme* $\to_h$*-normale.*

— *La* réduction normale *est la plus petite relation* $\to_n$ *telle que pour tout terme* t*,*

(i) si $t \to_h u$ *alors* $t \to_n u$*,*

(ii) sinon t *est de la forme* $\lambda\vec{x}.h\ t_1 \dots t_n$ *où* h *n'est pas une application et*

– *si* h *est en forme* β*-normale, et* t_j *est en forme* β*-normale pour tout* $j < i$*, alors* $t_i \to_n t'_i$ *implique*

$$\lambda\vec{x}.h\, t_1 \dots t_{i-1}\, t_i\, t_{i+1} \dots t_n \quad \to_n \quad \lambda\vec{x}.h\, t_1 \dots t_{i-1}\, t'_i\, t_{i+1} \dots t_n \ ,$$

– *sinon, on est dans le cas (c) du lemme 3.2.6 et il existe* $i \leq n$ *tel que* t_j *est en forme* β*-normale pour tout* $j < i$*; alors* $t_i \to_n t'_i$ *implique*

$$\lambda\vec{x}.f(t_1, \dots, t_{i-1},\ t_i,\ t_{i+1}, \dots, t_n)\, t_{n+1} \dots t_{n+m} \quad \to_n$$
$$\lambda\vec{x}.f(t_1, \dots, t_{i-1},\ t'_i,\ t_{i+1}, \dots, t_p)\, t_{n+1} \dots t_{n+m} \ .$$

La réduction normale est une réduction *standard*.

Remarque 3.2.8 *On a* $\to_{wh_\beta} \subseteq \to_h \subseteq \to_n \subseteq \to_\beta$.

Théorème 3.2.9 (Normalisation — Théorème 13.2.2 dans [Bar84]) *Si* t *a une forme* β*-normale, alors la réduction normale termine sur* t.

Le λ-calcul étant un formalisme Turing complet, il existe des λ-termes qui admettent des β-dérivations infinies. Nous les appelons des « termes infinis ».

Exemple 3.2.10 (Termes « infinis ») *Considérons les termes*

$$\omega \quad =_{def} \quad \lambda x.xx \qquad \text{et} \qquad \Omega \quad =_{def} \quad \omega\omega \ .$$

Le terme Ω *se réduit en lui même :*

$$\Omega \quad \to_\beta \quad \omega\omega \quad \to_\beta \quad \dots \ .$$

On peut aussi former des combinateurs de point fixe. Par exemple, pour tout $t \in \Lambda(\Sigma)$*, posons*

$$\omega_t \quad =_{def} \quad \lambda x.t(xx) \qquad \text{et} \qquad Y_t \quad =_{def} \quad \omega_t\omega_t \ .$$

On a alors

$$Y_t \quad \to_\beta \quad t\, Y_t \quad \to_\beta \quad t\, (t\, Y_t) \quad \to_\beta \quad \dots \ .$$

On étend les termes ω_t *et* Y_t *aux symboles unaires* $f(\_) \in \Sigma_1$ *comme suit :*

$$\omega_f \quad =_{def} \quad \lambda x.f(xx) \qquad \text{et} \qquad Y_f \quad =_{def} \quad \omega_f\omega_f \ .$$

Cela donne

$$Y_f \quad \to_\beta \quad f(Y_f) \quad \to_\beta \quad f(f(Y_f)) \quad \to_\beta \quad \dots \ .$$

3.3 Lambda-calcul typé

Un des moyens de sélectionner les λ-termes en fonction de leur comportement (p. ex. d'éviter d'avoir des termes infinis) est d'imposer des contraintes sur leur construction. Cela peut être fait en utilisant du typage.

Nous rappelons ici des systèmes de types pour le λ-calcul issus de l'isomorphisme de Curry-Howard. Nous nous intéressons essentiellement aux systèmes *à la Curry* : les abstractions sont celles du λ-calcul pur, elles n'ont pas d'annotations de types. Les systèmes *à la Church*, dans lesquels les abstractions possèdent une information de type sont évoqués en 3.3.3. Cette présentation est bien sûr très succincte, voir [Bar92, GLT89, Kri90, Miq01] pour plus de détails.

3.3.1 Lambda-calcul pur simplement typé

Les types du λ-calcul simplement typé correspondent usuellement aux formules de la logique intuitionniste minimale : ils sont construits à partir d'un ensemble de types de base et du constructeur binaire $\_ \Rightarrow \_$. Nous gagnons en souplesse en autorisant les types de base à avoir une arité.

Définition 3.3.1 (Types simples) *Étant donnée une signature $\mathcal{B} = (\mathcal{B}_i)_{i\in\mathbb{N}}$, l'ensemble $\mathcal{T}_\Rightarrow(\mathcal{B})$ des* types simples *sur $\mathcal{B}$ est l'ensemble engendré par la grammaire suivante :*

$$T, U \in \mathcal{T}_\Rightarrow(\mathcal{B}) \quad ::= \quad B(T_1, \dots, T_n) \quad | \quad T \Rightarrow U$$

où $B \in \mathcal{B}_n$.

L'ensemble $\mathcal{T}_\Rightarrow(\mathcal{B})$ est bien entendu identique à $\mathcal{T}er(\mathcal{B} \cup \{\_ \Rightarrow \_\})$. On écrit $\mathcal{T}_\Rightarrow$ pour désigner $\mathcal{T}_\Rightarrow(\emptyset)$. Lorsque le contexte le permet, on désigne $\mathcal{T}_\Rightarrow(\mathcal{B})$ par $\mathcal{T}(\mathcal{B})$.

Les termes du λ-calcul simplement typé sont ceux du λ-calcul pur :

$$t, u \in \Lambda_\Rightarrow \quad ::= \quad x \quad | \quad \lambda x.u \quad | \quad tu$$

où $x \in \mathcal{X}$. Nous les considérons égaux modulo α-conversion, comme défini en 1.2.

Rappelons qu'une valuation est une fonction de domaine fini.

Définition 3.3.2 (Contextes de typage) *Un* contexte de typage *pour $\mathcal{T}_\Rightarrow(\mathcal{B})$ est une valuation $\Gamma : \mathcal{X} \to \mathcal{T}_\Rightarrow(\mathcal{B})$.*

Nous écrivons $x : T \in \Gamma$ lorsque $\Gamma(x) = T$ et $x \in \Gamma$ lorsque $x \in \mathcal{D}om(\Gamma)$. Quand $x \notin \Gamma$, on désigne par $\Gamma, x : T$ le contexte tel que $(\Gamma, x : T)(x) = T$ et $(\Gamma, x : T)(y) = \Gamma(y)$ pour tout $y \neq x$.

Les règles de typage sont les suivantes.

$$(\mathrm{Ax})\ \frac{}{\Gamma, x : T \vdash x : T}$$

$$(\Rightarrow\mathrm{I})\ \frac{\Gamma \vdash t : U \Rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T} \qquad\qquad (\Rightarrow\mathrm{E})\ \frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \Rightarrow T}$$

En mettant tout ensemble, obtient le système $\boldsymbol{\lambda}_\Rightarrow(\mathcal{B})$.

Définition 3.3.3 (Lambda-calcul simplement typé) *On désigne par $\boldsymbol{\lambda}_\Rightarrow(\mathcal{B})$* le λ-calcul simplement typé avec types de base dans $\mathcal{B}$, *constitué*

- *de l'ensemble de termes $\Lambda_\Rightarrow$,*
- *de l'ensemble de types $\mathcal{T}_\Rightarrow(\mathcal{B})$,*
- *de la relation de typage dénotée par $\Gamma \vdash_\Rightarrow t : T$, définie comme étant la plus petite relation $\_ \vdash \_ : \_$ close par les règles de typage* (Ax), ($\Rightarrow$I) *et* ($\Rightarrow$E),
- *de la règle de réduction*

$$(\lambda x.t)u \quad \mapsto_\beta \quad t[u/x]\ .$$

Remarque 3.3.4 (Isomorphisme de Curry-Howard) *Les* dérivations de typage *de ce calcul sont isomorphes aux dérivations de la logique propositionnelle minimale : c'est* l'isomorphisme de Curry-Howard.

Notons que, pour autant, les termes *de ne sont pas isomorphes aux dérivations, et ce pour deux raisons :*

(i) d'une part, le typage à la Curry *implique que le terme* $\lambda x.x$ *est une preuve de la proposition* $A \Rightarrow A$ *pour toute proposition* A *;*

(ii) d'autre part, les affaiblissements n'étant pas pris en compte de manière explicite, le terme $\lambda x.x$ *est typable dans n'importe quel contexte* Γ*, en particulier lui correspondent les dérivations suivantes :*

$$\frac{A \vdash A}{\vdash A \Rightarrow A} \qquad \frac{B, A \vdash A}{B \vdash A \Rightarrow A}$$

Exemple 3.3.5 *Pour tout* $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B})$*, on a*

$$\vdash_{\Rightarrow} \mathrm{Id} : T \Rightarrow T \ .$$

3.3.2 Système F

Le système F a été introduit indépendamment par Girard et Reynolds dans les années 70. Girard l'a utilisé pour résoudre positivement, avec des outils de la « théorie de la preuve », la « conjecture de Takeuti » selon laquelle la logique du second ordre admet l'élimination de coupures [Gal89] ; et Reynolds l'a utilisé comme formalisation de la notion de polymorphisme paramétrique.

On considère deux versions du système F. Pour la première, la quantification de types $\forall X.\_$ ne laisse pas de trace au niveau des termes, elle est dite *implicite*. Pour la seconde, au contraire, les introductions et éliminations de la quantification $\forall X.\_$ sont reflétées au niveau des termes par des constructions correspondantes, cette version est dite *explicite*. Dans le cadre du calcul des constructions, les différences entre quantifications implicites et explicites sont étudiées dans [Miq01].

D'un point de vue pratique, la différence essentielle entre le système F explicite à la Church et le système F implicite à la Curry est que la vérification du typage est décidable dans le premier alors qu'elle est indécidable dans le second [Wel99].

3.3.2.(a) Types du second ordre

Les types du système F usuel correspondent aux formules de la logique propositionnelle minimale du second ordre. Comme pour le λ-calcul simplement typé, nous utilisons des types de base avec arité.

Comme ils contiennent un lieur, les types du second ordre sont en fait des quotients pour une relation d'α-conversion. Le traitement est très similaire à celui effectué pour les λ-termes en 1.2 et 1.3.

Définition 3.3.6 (Types du second ordre) *Soit une signature* $\mathcal{B} = (\mathcal{B}_i)_{i \in \mathbb{N}}$*, et* $\mathcal{V}$ *un ensemble dénombrable de* variables de types.

— *L'ensemble* $\Theta_2(\mathcal{B})$ *des* types bruts du second ordre *est l'ensemble généré par la grammaire suivante*

$$T, U \in \Theta_2(\mathcal{B}) \quad ::= \quad X \quad | \quad B(T_1, \ldots, T_n) \quad | \quad T \Rightarrow U \quad | \quad \forall X.T$$

où $B \in \mathcal{B}_n$ *et* $X \in \mathcal{V}$.

— *L'ensemble* $\mathcal{T}_2(\mathcal{B})$ *des* types du second ordre *est l'ensemble* $\Theta_2(\mathcal{B})/=_{\alpha\mathcal{T}}$, *où* $=_{\alpha\mathcal{T}}$ *est la plus petite relation d'équivalence sur* $\Theta_2(\mathcal{B})$ *qui satisfait*

$$\begin{array}{rcll} X & =_{\alpha\mathcal{T}} & X\ , & \\ B(\vec{T}) & =_{\alpha\mathcal{T}} & B(\vec{U}) & \textit{si } \vec{T} =_{\alpha\mathcal{T}} \vec{U}\ , \\ T_1 \Rightarrow T_2 & =_{\alpha\mathcal{T}} & U_1 \Rightarrow U_2 & \textit{si } T_1 =_{\alpha\mathcal{T}} U_1 \textit{ et } T_2 =_{\alpha\mathcal{T}} U_2\ , \\ \forall X.T & =_{\alpha\mathcal{T}} & \forall X.U & \textit{si } T[X \mapsto Z] =_{\alpha\mathcal{T}} U[Y \mapsto Z] \\ & & & \textit{pour tout } Z \in \mathcal{V} \textit{ sauf un nombre fini}\ . \end{array}$$

Nous désignons par $[T]_{\mathcal{T}}$ *la classe d'*$\alpha\mathcal{T}$*-équivalence de* $T \in \Theta_2(\mathcal{B})$.

— *Les ensembles des variables libres et liées d'un type* T, *respectivement notés* $FV(T)$ *et* $BV(T)$, *sont définis inductivement comme suit :*

$$\begin{array}{lll} \textit{Si } T = X \in \mathcal{V} & \textit{alors } FV(T) = \{X\} & \textit{et } BV(T) = \emptyset\ , \\ \textit{Si } T = B(T_1, \ldots, T_n) & \textit{alors } FV(T) = \bigcup_{1 \leq i \leq n} FV(T_i) & \textit{et } BV(T) = \bigcup_{1 \leq i \leq n} BV(T_i)\ , \\ \textit{Si } T = T_1 \Rightarrow T_2 & \textit{alors } FV(T) = FV(T_1) \cup FV(T_2) & \textit{et } BV(T) = BV(T_1) \cup BV(T_2)\ , \\ \textit{Si } T = \forall X.T_1 & \textit{alors } FV(T) = FV(T_1) \setminus \{X\} & \textit{et } BV(T) = BV(T_1) \cup \{X\}\ . \end{array}$$

Notons, dans la définition 3.3.6, que $\Theta_2(\mathcal{B})$ est l'ensemble

$$\mathcal{T}er(\mathcal{B} \cup \{\_ \Rightarrow \_\} \cup \{\forall X.\_ \mid X \in \mathcal{V}\}, \mathcal{V})\ .$$

De ce fait, l'opération de substitution simple $T[X \mapsto U]$ est définie en 1.1.14.

Nous définissons la substitution sans capture exactement comme en 1.3.

Définition 3.3.7 (Substitution de type sans capture) *Étant donnée une valuation* θ *sur* $\Theta_2(\mathcal{B})$, *la* substitution partielle sans capture $\_\theta$ *est la fonction partielle* $\Theta_2(\mathcal{B}) \to \Theta_2(\mathcal{B})$ *définie comme suit :*

$$\begin{array}{lll} X\theta & =_{def} \theta(X) & \textit{si } X \in \mathcal{D}om(\theta) \\ X\theta & =_{def} X & \textit{sinon} \\ B(T_1, \ldots, T_n)\theta & =_{def} B(T_1\theta, \ldots, T_n\theta) & \textit{si } B \in \mathcal{B}_n \\ (T_1 \Rightarrow T_2)\theta & =_{def} T_1\theta \Rightarrow T_2\theta & \\ (\forall X.T_1)\theta & =_{def} \forall X.(T_1\theta) & \textit{si } X \notin \mathcal{D}om(\theta) \cup FV(\theta) \end{array}$$

Étant donnée une valuation $[\theta]_{\mathcal{T}} : \mathcal{V} \to \mathcal{T}_2(\mathcal{B})$, *la* substitution sans capture *sur* $\mathcal{T}_2(\mathcal{B})$, *notée* $\_[\theta] : \mathcal{T}_2(\mathcal{B}) \to \mathcal{T}_2(\mathcal{B})$, *est définie par* $[T]_{\mathcal{T}}\theta =_{def} [T'\theta']_{\mathcal{T}}$ *pour* $\theta' \in [\theta]_{\mathcal{T}}$ *et* $T' =_{\alpha\mathcal{T}} T$ *tel que* $T'\theta'$ *est défini.*

La propriété importante qui assure la correction de la définition 3.3.7 est la suivante.

Proposition 3.3.8 *Soit une valuation* $\theta : \mathcal{V} \to \mathcal{T}_2(\mathcal{B})$. *La relation qui à chaque* $[T]_{\mathcal{T}} \in \mathcal{T}_2(\mathcal{B})$ *associe l'ensemble des* $[T'\theta']_{\mathcal{T}}$ *pour* $T' =_{\alpha\mathcal{T}} T$ *et* $\theta' \in [\theta]_{\mathcal{T}}$ *tels que* $T'\theta'$ *soit défini est une fonction totale de* $\mathcal{T}_2(\mathcal{B})$ *vers* $\mathcal{T}_2(\mathcal{B})$.

Preuve. Voir la proposition 1.3.7. □

Si θ est la substitution sans capture sur $\mathcal{T}_2(\mathcal{B})$ issue de la valuation qui associe T_i à X_i pour $i \in \{1, \dots, n\}$, alors $[[T_1]_{\mathcal{T}}/X_1, \dots, [T_n]_{\mathcal{T}}/X_n]$ désigne la substitution issue de θ et $[T]_{\mathcal{T}}[[T_1]_{\mathcal{T}}/X_1, \dots, [T_n]_{\mathcal{T}}/X_n]$ désigne son application au type $[T]_{\mathcal{T}} \in \mathcal{T}_2(\mathcal{B})$. Enfin, $\theta[U/X]$ est la substitution de domaine $\mathcal{D}om(\theta) \cup \{X\}$ qui vaut U en X et qui est égale à θ sur $\mathcal{D}om(\theta) \setminus \{X\}$.

Rappelons quelques propriétés des substitutions sans capture prouvées à la section 1.3.

Proposition 3.3.9

(i) *Si* θ *et* θ' *sont deux substitutions telles que* $FV(T) \cap \mathcal{D}om(\theta) = FV(T) \cap \mathcal{D}om(\theta')$ *et* $\theta(X) = \theta'(X)$ *pour tout* $X \in \mathcal{D}om(\theta) \cap FV(T)$ *alors* $T\theta = T\theta'$.

(ii) *Si* $FV(T) \cap \mathcal{D}om(\theta) = \emptyset$ *alors* $T\theta = T$.

(iii) *Si* $X \notin \mathcal{D}om(\theta) \cup FV(\theta)$ *alors* $T(\theta[U/X]) = (T\theta)[U/X]$.

Le raisonnement de 1.4 s'applique aussi ici. Ainsi, $\mathcal{T}_{\Rightarrow}(\mathcal{B})$ est isomorphe en tant que $(\mathcal{B} \cup \{\_ \Rightarrow \_\})$-algèbre à son quotient par $=_{\alpha\mathcal{T}}$, ce qui permet de voir les types simples comme un sous-ensemble des types du second ordre. Par convention, on écrit $T \in \mathcal{T}_2(\mathcal{B})$ pour désigner un type du second ordre.

Définition 3.3.10 Les contextes de typage pour $\mathcal{T}_2(\mathcal{B})$ *sont les valuations* $\Gamma : \mathcal{X} \to \mathcal{T}_2(\mathcal{B})$.

Voyons maintenant les versions implicites et explicites du système F.

3.3.2.(b) Système F implicite

Les termes du système F implicite sont ceux du λ-calcul :

$$t, u \in \Lambda_2 \quad ::= \quad x \quad | \quad tu \quad | \quad \lambda x.t$$

où $x \in \mathcal{X}$.

Ses règles de typage sont celles de $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B})$ plus les règles $(\forall\,\mathrm{I})$ et $(\forall\,\mathrm{E})$ suivantes :

$$(\forall\,\mathrm{I})\ \frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T}\ (X \notin FV(\Gamma)) \qquad\qquad (\forall\,\mathrm{E})\ \frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : T[U/X]}$$

Définition 3.3.11 (Système F implicite) *On désigne par* $\boldsymbol{\lambda}_2(\mathcal{B})$ le système F implicite avec types de base dans $\mathcal{B}$, *constitué*

— *de l'ensemble de termes* Λ_2,
— *de l'ensemble de types* $\mathcal{T}_2(\mathcal{B})$,
— *de la relation de typage dénotée par* $\Gamma \vdash_2 t : T$, *définie comme étant la plus petite relation* $\_ \vdash \_ : \_$ *close par les règles de typage de* $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B})$ *plus* $(\forall\,\mathrm{I})$ *et* $(\forall\,\mathrm{E})$,

— *de la règle de réduction*

$$(\lambda x.t)u \quad \mapsto_\beta \quad t[u/x] \ .$$

Exemple 3.3.12 *La fonction* Id *a un type très naturel dans le système F implicite :*

$$\vdash_2 \mathrm{Id} : \forall X.X \Rightarrow X \ .$$

3.3.2.(c) Système F explicite

Les termes du système F explicite contiennent l'information de typage de manière explicite :

$$\begin{array}{rcccccc} t, u \in \Lambda_{2\wedge}(\mathcal{B}) & ::= & x & | & t\,u & | & \lambda x.t \\ & | & t \cdot_\wedge T & | & \wedge X.t & & \end{array}$$

où $x \in \mathcal{X}$, $X \in \mathcal{V}$ et $T \in \mathcal{T}_2(\mathcal{B})$. Ses règles de typage sont celles de $\boldsymbol{\lambda}_\Rightarrow(\mathcal{B})$ plus les règles $(\forall\,\mathrm{I})$ et $(\forall\,\mathrm{E})$ suivantes :

$$(\forall_\wedge \mathrm{I})\ \frac{\Gamma \vdash t : T}{\Gamma \vdash \wedge X.t : \forall X.T}\ (X \notin \mathrm{FV}(\Gamma)) \qquad (\forall_\wedge \mathrm{E})\ \frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t \cdot_\wedge U : T[U/X]}$$

Lorsque le contexte le permet, les applications $t \cdot_\wedge U$ qui correspondent à l'instantiation de la quantification universelle du second ordre, sont écrites $t \cdot U$ ou même tU.

Nous n'explicitons pas le traitement de l'α-conversion induit par le lieur $\wedge X.\_$.

Définition 3.3.13 (Système F explicite) *On désigne par* $\boldsymbol{\lambda}_{2\wedge}(\mathcal{B})$ le système F implicite avec types de base dans $\mathcal{B}$, *constitué*

— *de l'ensemble de termes* $\Lambda_{2\wedge}(\mathcal{B})$,
— *de l'ensemble de types* $\mathcal{T}_2(\mathcal{B})$,
— *de la relation de typage dénotée par* $\Gamma \vdash_{2\wedge} t : T$, *définie comme étant la plus petite relation* $\_ \vdash \_ : \_$ *close par les règles de typage de* $\boldsymbol{\lambda}_\Rightarrow(\mathcal{B})$ *plus* $(\forall_\wedge \mathrm{I})$ *et* $(\forall_\wedge \mathrm{E})$,
— *des règles de réduction*

$$(\lambda x.t)u \quad \mapsto_\beta \quad t[u/x] \qquad\qquad (\wedge X.t) \cdot_\wedge U \quad \mapsto_\beta \quad t[U/X] \ .$$

Exemple 3.3.14 *La fonction* Id *est typable de la manière suivante dans le système F explicite :*

$$\vdash_{2\wedge} \wedge X.\mathrm{Id} : \forall X.X \Rightarrow X \ .$$

Sauf précisé explicitement, nous manipulons toujours le système implicite. D'autre part, il est parfois commode d'écrire $\boldsymbol{\lambda}_{\Rightarrow\wedge}(\mathcal{B})$ pour désigner $\boldsymbol{\lambda}_\Rightarrow(\mathcal{B})$.

3.3.2.(d) Propriétés

Voici deux propriétés importantes du système F. Elles sont vraies pour les systèmes implicite et explicite.

Lemme 3.3.15 (Préservation du type par réduction) *Pour tout* $ty \in \{\Rightarrow, 2, 2\wedge\}$, *si* $\Gamma \vdash_{ty} t : T$ *et* $t \to_\beta u$ *alors* $\Gamma \vdash_{ty} u : T$.

Théorème 3.3.16 (Normalisation forte) *Pour tout* $ty \in \{\Rightarrow, 2, 2\Lambda\}$*, si* $\Gamma \vdash_{ty} t : T$ *alors* $t \in \mathcal{SN}$*.*

Au chapitre 9, nous revenons en détail sur les preuves par réductibilité de la normalisation forte du λ-calcul typé. Une preuve du théorème 3.3.16 est donnée à la section 9.2.2.

3.3.3 Lambda-calculs avec abstractions à la Church

Les λ-calculs typés peuvent être définis en utilisant une syntaxe différente pour les termes, qui fait apparaître le type des variables sous les abstractions. Il s'agit d'abstractions *à la Church*. Voir [Bar92] pour une comparaison des systèmes *à la Curry* et des systèmes *à la Church*.

La syntaxe des termes doit être adaptée. Voici par exemple ceux du λ-calcul simplement typé à la Church :

$$t, u \quad ::= \quad x \quad | \quad tu \quad | \quad \lambda x : T.u \tag{3.1}$$

où $x \in \mathcal{X}$ et $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B})$.

Le fait d'avoir une annotation de type dans les abstractions implique en particulier que les variables liées ont un type unique, qui de plus est donné par la construction du terme.

Il existe un moyen simple d'avoir des λ-termes avec ces propriétés tout en restant dans la syntaxe du calcul à la Curry : au lieu de marquer le type des variables dans les abstractions, on assigne à chaque variable un unique type, dont on exige le respect par les contextes de typage. Ainsi, pour $ty \in \{\Rightarrow, 2\}$, on suppose que $\mathcal{X} = (\mathcal{X}_T)_{T \in \mathcal{T}_{ty}(\mathcal{B})}$ où les $\mathcal{X}_T$ sont des ensembles infinis dénombrables et deux à deux disjoints. On écrit x^T pour dire $x \in \mathcal{X}_T$. Comme les variables ont un type, on doit adapter la notion d'α-conversion (voir définition 1.2.6). Seul le cas de l'abstraction est modifié :

$$\lambda x^U.t \quad =_\alpha \quad \lambda y^U.u \quad \text{si } t[x \mapsto z] =_\alpha u[y \mapsto z] \text{ pour tout } z \in \mathcal{X}_U \text{ sauf un nombre fini .}$$

Définition 3.3.17 (Lambda-calculs à la Church) *Soit* $ty \in \{\Rightarrow, 2\}$*. On désigne par* $\boldsymbol{\lambda}^{Ch}_{ty}(\mathcal{B})$ *le calcul dont*

- *les types sont ceux de* $\mathcal{T}_{ty}(\mathcal{B})$ *;*
- *les termes sont ceux de* $\Lambda_{ty}(\mathcal{B})$*, construits à partir de* $\mathcal{X} = (\mathcal{X}_T)_{T \in \mathcal{T}_{ty}(\mathcal{B})}$*, dont l'ensemble est désigné par* $\Lambda^{Ch}_{ty}(\mathcal{B})$ *;*
- *les règles de typage sont celles de* $\boldsymbol{\lambda}_{ty\wedge}(\mathcal{B})$*, restreintes aux contextes de typage* Γ *tels que* $x : T \in \Gamma$ *implique* $x \in \mathcal{X}_T$*.*

On note $\Gamma \vdash^{Ch}_{ty} t : T$ *lorsque* $\Gamma \vdash t : T$ *est dérivable avec les règles de* $\boldsymbol{\lambda}^{Ch}_{ty}(\mathcal{B})$*.*

Remarque 3.3.18 *Les versions usuelles du système F explicite (voir p. ex. [GLT89, Gal89]) utilisent aussi des abstractions à la Church pour les variables de terme.*

Exemple 3.3.19 *Dans le système F* à la Church*, le terme* Id *n'existe pas en tant que tel. Le terme qui lui correspond est* $\mathrm{Id}_{Ch} =_{def} \Lambda X.\lambda x^X.x$*, typé de la manière suivante :*

$$\vdash^{Ch}_2 \mathrm{Id}_{Ch} : \forall X.X \Rightarrow X \ .$$

Les calculs à la Church usuels (tels que 3.1) vérifient la propriété *d'unicité du typage* : pour tout contexte Γ et tout terme t, il existe au plus un type tel que $\Gamma \vdash t : T$.

Les systèmes *à la Church* tels que nous les définissons ne sont en fait pas exactement équivalents aux systèmes à la Church usuels. En particulier, l'unicité du typage est plus forte : puisque le type des variables est fixé a priori, le type d'un terme est indépendant du contexte de typage.

Lemme 3.3.20 (Unicité du typage) *Soit* $ty \in \{\Rightarrow, 2\}$ *et* $t \in \Lambda_{ty}^{Ch}$. *Alors*

$$\left(\Gamma \vdash t : T \quad \textit{et} \quad \Gamma' \vdash t : T'\right) \qquad \textit{implique} \qquad T = T' \ .$$

Remarque 3.3.21 *Pour les systèmes* à la Church *tels que nous les définissons, si* $\Gamma \vdash t : T$ *et* $\Gamma' \vdash t : T$ *alors* Γ *et* Γ' *coïncident sur* $FV(t)$. *Lorsque le contexte le permet, nous écrivons* $t : T$ *au lieu de* $\Gamma \vdash t : T$.

Les propriétés énoncées par 3.3.15 et 3.3.16 sont aussi satisfaites par les systèmes *à la Church*.

Lemme 3.3.22 (Préservation du type par réduction) *Pour tout* $ty \in \{\Rightarrow, 2\}$, *si* $\Gamma \vdash_{ty}^{Ch} t : T$ *et* $t \to_\beta u$ *alors* $\Gamma \vdash_{ty}^{Ch} u : T$.

Théorème 3.3.23 (Normalisation forte) *Pour tout* $ty \in \{\Rightarrow, 2\}$, *si* $\Gamma \vdash_{ty}^{Ch} t : T$ *alors* $t \in \mathcal{SN}_\beta$.

3.4 Types de données dans le système F

Le système F est suffisamment puissant pour encoder beaucoup de types de données intéressants. On peut trouver dans [GLT89, Kri90] plus de détails à ce sujet.

Dans cette section, nous voyons comment encoder dans le système F les types de données présentés à la section 1.1 dans le cadre de la réécriture au premier ordre. Les explications sur les codages imprédicatifs mélangent volontairement les constructions algébriques et les λ-termes.

3.4.1 Booléens

Les booléens algébriques sont construits à partir de deux constructeurs true et false. Le type des booléens est donc le plus petit type T tel qu'on a un élément de T dès lors qu'on a true ou false. On pose

$$T_{\mathsf{Bool}} \quad =_{\text{def}} \quad \forall X.\ X \Rightarrow X \Rightarrow X \ .$$

On représente true et false par les deux habitants les plus simples possibles de T_{Bool} :

$$\mathtt{True} \quad =_{\text{def}} \quad \lambda xy.x \qquad\qquad \mathtt{False} \quad =_{\text{def}} \quad \lambda xy.y \ .$$

On les élimine avec un terme $\mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ t$ tel que $\mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ \mathsf{True}$ et $\mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ \mathsf{False}$ simulent les comportements respectifs de $\mathsf{iter}_{\mathsf{Bool}}(u, v, \mathsf{true})$ et de $\mathsf{ite}_{\mathsf{Bool}}(u, v, \mathsf{false})$. C'est-à-dire :

$$\mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ \mathsf{True} \quad \to_\beta^* \quad u \qquad \text{et} \qquad \mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ \mathsf{False} \quad \to_\beta^* \quad v\ .$$

Il doit donc pouvoir être typé comme suit :

$$u, v : U, b : \mathsf{T}_{\mathsf{Bool}} \quad \vdash_2 \quad \mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ b : U$$

Ces deux contraintes sont satisfaites en posant $\mathsf{Iter}_{\mathsf{Bool}}\ u\ v\ t =_{\text{def}} t\ u\ v$.

3.4.2 Paires

Les paires algébriques sont construites à partir d'un constructeur binaire $(\_, \_)$. Donc le type des paires $A \times B$ est le plus petit type T construit à partir d'une fonction dans $A \Rightarrow B \Rightarrow T$. On pose

$$\mathsf{T}_{A\times B} \quad =_{\text{def}} \quad \forall X.\ (A \Rightarrow B \Rightarrow X) \Rightarrow X\ .$$

On peut alors représenter $(\_, \_)$ par

$$\mathsf{Pair} \quad =_{\text{def}} \quad \lambda x \lambda y \lambda p.p\ x\ y \quad : \quad A \Rightarrow B \Rightarrow \mathsf{T}_{A\times B}\ .$$

Le terme (a, b) correspond à $\mathsf{Pair}\ a\ b$, dont la forme $\to_{wh_\beta}$-normale est $\lambda p.p\ a\ b$.

Les deux projections sont codées à partir d'un itérateur de paires. C'est un terme $\mathsf{Iter}_{\_\times\_}\ u\ p$ tel que

$$\mathsf{Iter}_{\_\times\_}\ u\ (\mathsf{Pair}\ a\ b) \quad \to_\beta \quad u\ a\ b\ ,$$

et qui doit donc être typé de la manière suivante :

$$u : A \Rightarrow B \Rightarrow U, p : \mathsf{T}_{A\times B} \quad \vdash_2 \quad \mathsf{Iter}_{\_\times\_}\ u\ p : U\ .$$

On pose $\mathsf{Iter}_{\_\times\_}\ u\ p =_{\text{def}} p\ u$ et on code les projections par

$$\mathsf{Fst}\ t \quad =_{\text{def}} \quad \mathsf{Iter}_{\_\times\_}\ \mathsf{True}\ t \qquad \text{et} \qquad \mathsf{Snd}\ t \quad =_{\text{def}} \quad \mathsf{Iter}_{\_\times\_}\ \mathsf{False}\ t\ .$$

Alors on a

$$\begin{array}{rcccccl} \mathsf{Fst}\ (\mathsf{Pair}\ a\ b) & = & (\lambda p.p\ a\ b)\ \mathsf{True} & \to_\beta & (\lambda xy.x)\ a\ b & \to_\beta & a \\ \mathsf{Snd}\ (\mathsf{Pair}\ a\ b) & = & (\lambda p.p\ a\ b)\ \mathsf{False} & \to_\beta & (\lambda xy.y)\ a\ b & \to_\beta & b \end{array}$$

ainsi que

$$\begin{array}{rcl} a : A, b : B & \vdash_2 & \mathsf{Fst}\ (\mathsf{Pair}\ a\ b)\ :\ A \\ a : A, b : B & \vdash_2 & \mathsf{Snd}\ (\mathsf{Pair}\ a\ b)\ :\ B\ . \end{array}$$

Remarque 3.4.1 *Ce typage utilise* True *et* False *avec les types* $A \Rightarrow B \Rightarrow A$ *et* $A \Rightarrow B \Rightarrow B$ *respectivement. Ce n'est pas leur type en tant que booléens. Dans le système F* à la Church *nous ne pourrions pas utiliser ces termes comme sélecteurs dans les paires : dans ce système* True *et* False *sont les termes* $\Lambda X.\lambda x^X y^X.x$ *et* $\Lambda X.\lambda x^X y^X.y$ *respectivement, qui ne peuvent avoir pour type ni* $A \Rightarrow B \Rightarrow A$ *ni* $A \Rightarrow B \Rightarrow B$.

3.4.3 Entiers

Le type des entiers est le plus petit type T construit à partir d'une constante et d'un symbole de fonction unaire. On pose donc

$$\mathsf{T}_{\mathsf{Nat}} \quad =_{\text{def}} \quad \forall X.\, X \Rightarrow (X \Rightarrow X) \Rightarrow X\ .$$

Les termes

$$\mathsf{Zero} \quad =_{\text{def}} \quad \lambda xf.x \qquad \text{et} \qquad \mathsf{Succ} \quad =_{\text{def}} \quad \lambda n.\lambda xf.f(n\ x\ f)$$

sont de type $\mathsf{T}_{\mathsf{Nat}}$ et représentant respectivement les constructeurs 0 et $\mathsf{S}(\_)$. Pour tout $n \in \mathbb{N}$, le terme $\mathsf{S}^n(0)$ est représenté par le terme $\mathsf{Succ}^n\mathsf{Zero}$, dont la forme β-normale est l'entier de Church $\lambda xf.f^n x$ de type $\mathsf{T}_{\mathsf{Nat}}$.

Le système F permet aussi de coder l'itérateur $\mathsf{iter}(u, v, n)$: avec

$$\mathrm{Iter}\ u\ v\ n \quad =_{\text{def}} \quad n\ u\ v\ ,$$

on a

$$\begin{array}{lcllcl} \mathrm{Iter}\ u\ v\ \mathrm{Zero} & = & (\lambda xf.x)\ u\ v & \to_\beta & u & \\ \mathrm{Iter}\ u\ v\ (\mathrm{Succ}\ n) & = & (\lambda xf.f(n\ x\ f))\ u\ v & \to_\beta & v\ (n\ u\ v) & = \quad v\ (\mathrm{Iter}\ u\ v\ n) \end{array}$$

et

$$u : U, v : U \Rightarrow U, n : \mathsf{T}_{\mathsf{Nat}} \quad \vdash_2 \quad \mathrm{Iter}\ u\ v\ n : U\ .$$

3.4.4 Vers une extension du lambda-calcul

Nous avons vu comment coder dans le λ-calcul des types de données et leur schémas d'itération. De plus, ces codages sont typables grâce aux types imprédicatifs du système F. Cependant, ils souffrent de certaines limitations.

3.4.4.(a) Paires surjectives

Certaines interprétations catégorielles du λ-calcul valident les équations suivantes sur les paires et les projections : étant donnés deux objets A_1,A_2, on a deux morphismes $\pi_i : A_1 \times A_2 \to A_i$ ($i \in \{1,2\}$) tels que pour tout objet B, pour tout $f_i : A_i \to B$ ($i \in \{1,2\}$) et tout $h : B \to A_1 \times A_2$,

$$\begin{array}{rcl} \pi_1 \circ (f_1, f_2) & = & f_1 \\ \pi_2 \circ (f_1, f_2) & = & f_2 \\ (\pi_1 \circ h, \pi_2 \circ h) & = & h\ . \end{array} \tag{3.2}$$

On a vu en 3.1.4 une définition algébrique des paires et projections dans laquelle les deux premières équations sont implantées par les règles $\mapsto_\pi$ et la dernière par la règle $\mapsto_{\mathsf{SP}}$. En 3.4.2, on a vu comment coder le système de réécriture $\to_\pi$ dans la syntaxe du λ-calcul pur et comment typer ce codage grâce aux types imprédicatifs du système F. Cependant, Barendregt a montré que la règle $\mapsto_{\mathsf{SP}}$ n'est pas codable par la β-conversion dans le λ-calcul (voir [Bar84]).

3.4.4.(b) Schéma de récursion

Le récurseur rec présenté à l'exemple 3.1.17.(ii) n'est pas définissable en temps borné dans le système F : il n'y a pas de terme Rec tel qu'il existe $k \in \mathbb{N}$ tel que pour tout terme n,

$$\mathsf{Rec}\ u\ v\ 0 \quad \to_\beta^k \quad u \qquad \text{et} \qquad \mathsf{Rec}\ u\ v\ (\mathsf{Succ}\ n) \quad \to_\beta^k \quad v\ (\mathsf{Rec}\ u\ v\ n)\ n\ . \tag{3.3}$$

Cela provient d'un résultat de [Par89] disant qu'il n'y a pas, dans le λ-calcul (donc *à fortiori* dans le système F), de terme calculant le prédécesseur d'un entier de Church en temps borné. Or, un tel terme serait définissable s'il existait un terme Rec vérifiant (3.3). En effet :

$$\mathsf{Rec}\ 0\ (\lambda\_.\lambda x.x)\ 0 \quad \to_\beta^k \quad 0 \qquad \text{et} \qquad \mathsf{Rec}\ 0\ (\lambda\_.\lambda x.x)\ (\mathsf{Succ}\ n) \quad \to_\beta^{k+2} \quad n\ .$$

3.5 Extensionnalité et théorème de Böhm

Les limitations des codages dans le λ-calcul suggèrent d'enrichir la β-réduction. Dans cette section, nous rappelons le théorème de Böhm : sur les λ-termes purs et normalisants, il n'y a pas d'extension stricte consistante de la conversion du λ-calcul extensionnel. Cela suggère d'étudier des extensions de $\to_\beta$ sur $\Lambda(\Sigma)$ avec Σ non vide.

Nous commençons par quelques rappels sur le λ-calcul extensionnel.

Définition 3.5.1 (Éta-réduction) *La* notion d'η-réduction *est la plus petite relation* $\mapsto_\eta$ *telle que pour tout* $t \in \Lambda(\Sigma)$,

$$\lambda x.tx \quad \mapsto_\eta \quad t \qquad \textit{si} \qquad x \notin \mathsf{FV}(t)\ .$$

La relation d'η-réduction, *notée* $\to_\eta$ *est la clôture par contexte de* $\mapsto_\eta$.

Remarque 3.5.2

- *Sur* $\Lambda(\Sigma)$, *la relation* $\to_\eta$ *est confluente et fortement normalisante.*
- *De même que pour* $\to_\beta$, *comme* $\mapsto_\eta$ *est stable par substitution,* $\to_\eta$ *est une relation de réécriture par le lemme 2.2.3.*
- *Comme pour* $\to_\beta$, *il s'en suit que* $(\Lambda(\Sigma), \to_\eta)$ *est la relation de réécriture issue du TRS* $(\Lambda(\Sigma), \mapsto_\eta)$. *C'est donc aussi un TRS.*

Le résultat suivant dit que $=_{\beta\eta}$ correspond exactement à l'extensionnalité de l'application.

Théorème 3.5.3 (Curry — Théorème 2.1.29 dans [Bar84]) *Soit* $=_{\beta\mathrm{EXT}}$ *la plus petite congruence sur* Λ *contenant* $\mapsto_\beta$ *et close pour la règle suivante :*

$$\frac{\forall v.\ tv =_{\beta\mathrm{EXT}} uv}{t =_{\beta\mathrm{EXT}} u}$$

Pour tout $t, u \in \Lambda$, *on a* $t =_{\beta\mathrm{EXT}} u$ *si et seulement si* $t =_{\beta\eta} u$.

Théorème 3.5.4 (Confluence — [Bar84, Kri90]) *La relation $\to_{\beta\eta}$ est confluente.*

Le lemme suivant dit que l'η-réduction ne crée pas de β-rédexes dans les termes β-normaux.

Lemme 3.5.5 (Lemme 6 p. 52 dans [Kri90]) *Si t est un terme β-normal tel que $t \to_\eta u$, alors u est β-normal aussi.*

Venons en au théorème de Böhm, qui dit que le seul moyen consistant d'étendre $=_{\eta\beta}$ sur les termes β-normalisables est de rajouter des symboles de fonctions au λ-calcul. Ce résultat est bien entendu faux sur $\Lambda(\Sigma)$ avec Σ non vide.

Théorème 3.5.6 (Théorème de Böhm — Corollaire 2 p. 70 dans [Kri90]) *Soit $\equiv$ une congruence sur Λ qui contient $=_\beta$. S'il existe deux termes β-normalisables $t, u \in \Lambda$ tels que $t \equiv u$ mais $t \neq_{\beta\eta} u$, alors $\equiv$ est inconsistante sur Λ.*

Remarque 3.5.7 *Soit $\mathcal{R}$ un TRS sur $\Lambda(\Sigma)$. S'il existe deux termes* purs *β-normalisables $t, u \in \Lambda$ tels que $t =_{\mathcal{R}\beta} u$, mais $t \neq_{\beta\eta} u$, alors $\to_{\mathcal{R}\beta}$ est bien évidemment non confluente : les formes β-normales respectives de t, u sont des termes $\to_{\mathcal{R}\beta}$-normaux distincts mais $=_{\mathcal{R}\beta}$-convertibles (comme ils sont* purs, *t et u ne contiennent pas de symbole de Σ, ils sont donc en forme $\mathcal{R}$-normale, ainsi que tous leurs β-réduits).*

Ce que dit le théorème 3.5.6, c'est que $=_{\mathcal{R}\beta}$ n'est de surcroît pas consistante, ce qui est strictement plus fort : tous les termes de Λ *sont identifiés par $=_{\mathcal{R}\beta}$, en particulier* $\mathtt{True} =_{\mathcal{R}\beta} \mathtt{False}$*, ce qui permet d'identifier* tous les termes de $\Lambda(\Sigma)$.

3.6 Réécriture dans le lambda-calcul typé

Nous avons vu que le λ-calcul nécessite certaines extensions pour pouvoir y accommoder de façon satisfaisante des types de données. Le théorème de Böhm (3.5.6) nous dit que les extensions de la βη-réduction, pour être consistantes, doivent s'accompagner d'extensions de la syntaxe du λ-calcul pur. Une possibilité est l'étude d'extensions du λ-calcul par des signatures algébriques et des systèmes de réécriture de termes utilisant ces signatures.

Dans cette section, nous donnons les définissions permettant de parler rigoureusement de réécriture dans le λ-calcul typé.

Définition 3.6.1 (Typage des constantes) *Soit $\mathrm{ty} \in \{\Rightarrow, 2\}$.*

- *Une* signature typée *est une paire (Σ, τ) où Σ est une signature et $\tau = (\tau_n)_{n\in\mathbb{N}}$ est une famille de fonctions $\tau_n : \Sigma_n \to \mathcal{P}(\mathcal{T}_{\mathrm{ty}}(\mathcal{B})^{n+1})$.*
- *On dit que $\tau_n(f)$ est un* typage algébrique *lorsque $\tau_n(f)$ est singleton de la forme*

$$\{(B_1, \dots, B_n, B_{n+1} \Rightarrow \cdots \Rightarrow B_{n+m} \Rightarrow B)\} ,$$

  où $m \geq 0$ et $B_1, \dots, B_{n+m} \in \mathcal{B}_0$.
- *Si pour tout $n \in \mathbb{N}$, $\tau_n(f)$ est algébrique pour tout $f \in \Sigma_n$, alors (Σ, τ) est une* signature algébrique.

En particulier, $\tau_n(f)$ peut être une *fonction partielle* de $\mathcal{T}_{ty}(\mathcal{B})^n$ dans $\mathcal{T}_{ty}(\mathcal{B})$. Lorsque le contexte le permet, on écrit τ pour $\bigcup_{n\in\mathbb{N}} \tau_n$. Pour $ty \in \{\Rightarrow, 2, 2\Lambda\}$, on étend de manière directe les termes de $\Lambda_{ty}(\mathcal{B})$ à $\Lambda_{ty}(\mathcal{B}, \Sigma)$:

$$\Lambda_{ty}(\mathcal{B}, \Sigma) \quad ::= \quad \ldots \quad | \quad f(t_1, \ldots, t_n)$$

où $f \in \Sigma_n$. Les symboles de la signature sont typés de la manière suivante :

$$(\textsc{Symb I})\ \frac{\Gamma \vdash t_1 : T_1 \quad \ldots \quad \Gamma \vdash t_n : T_n}{\Gamma \vdash f(t_1, \ldots, t_n) : T}\ (T_1, \ldots, T_n, T) \in \tau(f)$$

Remarque 3.6.2 *Lorsque $\tau_n(f)$ est une fonction de $\mathcal{T}_{ty}(\mathcal{B})^n$ dans $\mathcal{T}_{ty}(\mathcal{B})$ telle que*

$$\tau(f)(T_1, \ldots, T_n) = T[T_1/X_1, \ldots, T_n/X_n]$$

pour tout $T_1, \ldots, T_n \in \mathcal{T}_{ty}(\mathcal{B})$, alors la règle (Symb I) *peut être écrite*

$$\frac{\Gamma \vdash t_1 : T_1 \quad \ldots \quad \Gamma \vdash t_n : T_n}{\Gamma \vdash f(t_1, \ldots, t_n) : T[T_1/X_1, \ldots, T_n/X_n]}$$

Définition 3.6.3 (Système de type avec constantes) *Étant donné $ty \in \{\Rightarrow, 2\}$, on désigne par $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$ le système constitué du λ-calcul typé dont*

- *les termes sont ceux de $\Lambda_{ty}(\mathcal{B}, \Sigma)$,*
- *les types sont ceux de $\mathcal{T}_{ty}(\mathcal{B})$,*
- *la signature (Σ, τ) est typée dans $\mathcal{T}_{ty}(\mathcal{B})$.*
- *les règles de typage sont celles de $\lambda_{ty}(\mathcal{B})$, auxquelles on ajoute la règle* (Symb I).

On écrit $\Gamma \vdash_{ty\tau} t : T$ lorsque $\Gamma \vdash t : T$ est dérivable dans $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$. Pour tout Γ à valeurs dans $\mathcal{T}_{ty}(\mathcal{B})$ et $T \in \mathcal{T}_{ty}(\mathcal{B})$, on pose $[\Gamma \vdash_{ty\tau} T] =_{def} \{t \mid \Gamma \vdash_{ty\tau} t : T\}$.

Lorsque le contexte le permet, on écrit $\lambda_{ty}(\mathcal{B}, \Sigma)$ pour désigner $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$, Σ étant vue comme la signature typée (Σ, τ). Dans ce cas, $\Gamma \vdash_{ty} t : T$ désigne $\Gamma \vdash_{ty\tau} t : T$.

Une des propriétés intéressantes avec le typage est la suivante, dont la preuve est une induction triviale :

Proposition 3.6.4 *Soit $ty \in \{\Rightarrow, 2, 2\Lambda\}$ et (Σ, τ) une signature algébrique. Si $t \in \Lambda_{ty}(\Sigma)$ est un terme β-normal tel qu'il existe Γ à valeurs dans $\mathcal{B}_0$ et $B \in \mathcal{B}_0$ tels que $\Gamma \vdash t : B$, alors t est algébrique.*

La propriété 3.6.4 ne caractérise pas les termes algébriques si on n'impose pas que les symboles algébriques soient complètement appliqués : le terme f de la signature typée $\{f : B \Rightarrow B\}$ est algébrique mais n'est pas typable par un type de base.

Définition 3.6.5 *Étant donnée une signature typée (Σ, τ) algébrique sur $\lambda_2(\mathcal{B}, \Sigma, \tau)$, les termes algébriquement typés sont définis inductivement comme suit :*

- *si Γ est un contexte à valeurs dans $\mathcal{B}_0$ et $B \in \mathcal{B}_0$, alors x est algébriquement typé dans $\Gamma, x : B$ de type B,*

— *si* $\tau(f) = \{(B_1, \dots, B_n, B_{n+1} \Rightarrow \cdots \Rightarrow B_{n+m} \Rightarrow B)\}$ *et, pour tout* $i \in \{1, \dots, n+m\}$, t_i *est algébriquement typé dans* Γ *de type* B_i, *alors* $f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m}$ *est algébriquement typé dans* Γ *de type* B.

La propriété 3.6.4 caractérise les termes algébriquement typés. En utilisant la curryfication (chapitre 8), on peut ainsi construire une algèbre libre.

Remarque 3.6.6 *Prenons une signature* Σ, *que l'on curryfie en* $\Sigma_{\mathcal{C}}$ *(voir l'exemple 3.1.13 et le chapitre 8). On associe donc à chaque* $f \in \Sigma_n$ *un* $f_{\mathcal{C}} \in \Sigma_{\mathcal{C}0}$. *On peut refléter cette opération aux niveau des types, en associant à* $f_{\mathcal{C}}$ *un type*

$$\tau_{\mathcal{C}}(f_{\mathcal{C}}) \quad = \quad \underbrace{B \Rightarrow \cdots \Rightarrow B}_{n \text{ fois}} \Rightarrow B \ .$$

Les termes algébriquement typés dans $\lambda_{ty}(\{B\}, \Sigma_{\mathcal{C}0}, \tau_{\mathcal{C}})$ *forment une* Σ*-algèbre libre sur* $\mathcal{X}$.

Il s'en suit que pour toute signature (Σ_0, τ) *algébriquement typée, les termes algébriquement typés de* $\lambda_{ty}(\{B\}, \Sigma_0, \tau)$ *forment une* Σ_τ*-algèbre sur* $\mathcal{X}$ *où*

$$\Sigma_\tau \quad =_{def} \quad \left(\left\{ f \mid \tau(f) \quad = \quad \underbrace{B \Rightarrow \cdots \Rightarrow B}_{n \text{ fois}} \Rightarrow B \right\} \right)_{n \in \mathbb{N}}$$

Tout ceci s'étend au cas multisorté, ce qui permet de prendre en compte le typage avec plusieurs types de base.

Définition 3.6.7 (Système de réécriture typé) *Étant donné* $ty \in \{\Rightarrow, 2\}$, *un système de réécriture* $\mathcal{R}$ *sur* Σ *algébrique à gauche est* un TRS typé dans $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$ *si pour toute règle* $l \mapsto_{\mathcal{R}} r$,

(i) l *est typable dans* $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$ *et*

(ii) pour toute substitution σ, *tout type* $T \in \mathcal{T}_{ty}(\mathcal{B})$ *et tout contexte* Γ *à valeurs dans* $\mathcal{T}_{ty}(\mathcal{B})$,

$$\Gamma \vdash l\sigma : T \qquad \textit{implique} \qquad \Gamma \vdash r\sigma : T \ .$$

Un TRS typé dans $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$ *est* algébriquement typé dans $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$ *si ses règles sont faites de termes algébriquement typés.*

Notons qu'un TRS algébriquement typé est nécessairement algébrique.

Remarque 3.6.8 *Soit* $ty \in \{\Rightarrow, 2\}$ *et* $\mathcal{R}$ *un TRS algébriquement typé dans* $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$. *Comme les membres gauches des règles de* $\mathcal{R}$ *sont des termes algébriques qui ne sont pas des variables, pour toute règle* $l \mapsto_{\mathcal{R}} t$, *il existe un* unique *type* $T \in \mathcal{T}_{ty}(\mathcal{B})$ *tel que pour toute substitution* σ, *si* $l\sigma$ *est typable dans* $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$, *alors il existe un contexte* Γ *à valeurs dans* $\mathcal{T}_{ty}(\mathcal{B})$ *tel que* $\Gamma \vdash l\sigma : T$.

Ces définitions s'adaptent facilement aux systèmes *à la Church* définis en 3.3.17 : si $ty \in \{\Rightarrow, 2\}$, le système $\lambda^{\mathcal{C}h}_{ty}(\mathcal{B}, \Sigma, \tau)$ est le système $\lambda_{ty}(\mathcal{B}, \Sigma, \tau)$ dont les termes sont ceux de $\Lambda^{\mathcal{C}h}_{ty}(\mathcal{B}, \Sigma)$ au lieu de $\Lambda_{ty}(\Sigma)$.

Le cas (ii) de la définition 3.6.7 permet d'étendre la préservation du typage par réduction à $\rightarrow_{\beta\mathcal{R}}$ (voir 3.3.15).

Lemme 3.6.9 (Préservation du type) *Soit* $\mathsf{ty} \in \{\Rightarrow, 2, 2\Lambda\}$ *et* $\mathcal{R}$ *un TRS typé dans* $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}, \Sigma, \tau)$. *Si* $\Gamma \vdash_{\mathsf{ty}} t : T$ *et* $t \to_{\beta\mathcal{R}} u$ *alors* $\Gamma \vdash_{\mathsf{ty}} u : T$.

La définition suivante est une notation pour les congruences entre les termes typés de certains systèmes. Rappelons que par la remarque 3.6.6, les termes algébriquement typés de $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}, \Sigma_0, \tau)$ forment une Σ_τ-algèbre libre sur $\mathcal{X}$.

Définition 3.6.10 (Congruence typée) *Pour tout* $\mathsf{ty} \in \{\Rightarrow, 2, 2\Lambda\}$, *pour toute signature* Σ *typée dans* $\mathcal{T}_{\mathsf{ty}}(\mathcal{B})$ *et pour toute congruence* $\equiv$ *sur* $\mathcal{T}_{\mathsf{ty}}(\mathcal{B}, \Sigma, \tau)$, *on note*

$$\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}, \Sigma) \vdash_{\mathsf{ty}} t \equiv u$$

lorsqu'il existe Γ *à valeurs dans* $\mathcal{T}_{\mathsf{ty}}(\mathcal{B})$ *et* $T \in \mathcal{T}_{\mathsf{ty}}(\mathcal{B})$ *tel que* $t \equiv u$ *dans* $[\Gamma \vdash_{\mathsf{ty}} T]$.

Si $\equiv$ *est une congruence sur* $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0, \tau)$, *on note*

$$\mathcal{T}\mathrm{er}(\Sigma_\tau, \mathcal{X}) \vdash_{\mathsf{ty}} t \equiv u$$

lorsque $t \equiv u$ *sur les termes algébriquement typés de* $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}, \Sigma_0, \tau)$.

3.7 Exemples de combinaisons

Voici quelques systèmes dans lesquels sont combinés le λ-calcul et un système de réécriture particulier. Nous revenons en particulier sur les paires surjectives (3.4.4.(a)) et sur le récurseur des entiers (3.4.4.(b)).

3.7.1 Lambda-calcul avec produits finis

Nous avons vu que les paires surjectives n'étaient pas définissables dans le λ-calcul pur. Voyons maintenant comment combiner le λ-calcul au système présenté en 3.1.4.

Définition 3.7.1 *L'*extension produit *d'une signature* $\mathcal{B}$ *est* $\mathcal{B}_\times =_{def} \mathcal{B} \uplus \{\_ \times \_\}$.

Les symboles de la signature $\Sigma_\pi =_{\text{def}} \{(\_, \_), \pi_1\_, \pi_2\_\}$ sont typés de la manière suivante :

$$\begin{array}{lcll} \tau_\times((\_, \_)) & : & (A, B) \in \mathcal{T}_{\mathsf{ty}}(\mathcal{B}_\times) \times \mathcal{T}_{\mathsf{ty}}(\mathcal{B}_\times) & \mapsto \quad A \times B \\ \tau_\times(\pi_1) & : & A \times B \in \mathcal{T}_{\mathsf{ty}}(\mathcal{B}_\times) & \mapsto \quad A \\ \tau_\times(\pi_2) & : & A \times B \in \mathcal{T}_{\mathsf{ty}}(\mathcal{B}_\times) & \mapsto \quad B\ . \end{array}$$

Il est aisé de voir que le système $\{\mapsto_\pi, \mapsto_{\mathsf{SP}}\}$ est typé dans $(\Sigma_\pi, \tau_\times)$. De plus, les instances de la règle (Symb) pour $\tau_\times$ correspondent aux règles de typage usuelles du produit :

$$(\times\mathrm{I})\ \frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash (t_1, t_2) : T_1 \times T_2} \qquad (\times\mathrm{E})\ \frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash \pi_i t : T_i}\ (i \in \{1, 2\})$$

Définition 3.7.2 (λ-calcul typé avec produits finis) *Soit* $\mathsf{ty} \in \{\Rightarrow, 2, 2\Lambda\}$.

— *On désigne* $\mathcal{T}_{\mathsf{ty}}(\mathcal{B}_\times)$ *par* $\mathcal{T}_{\mathsf{ty}\times}(\mathcal{B})$.

— *On appelle* λ-calcul typé avec produits *et on désigne par* $\boldsymbol{\lambda}_{\mathsf{ty}\times}(\mathcal{B})$ *le système* $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_\times, \Sigma_\pi, \tau_\times)$ *dont les règles de réductions sont* $\mapsto_{\beta\pi}$.

— *On appelle* λ-calcul typé avec produits et paires surjectives *et on désigne par* $\boldsymbol{\lambda}_{\mathsf{ty}\times\mathsf{SP}}(\mathcal{B})$ *le système* $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_\times, \Sigma_\pi, \tau_\times)$ *dont les règles de réductions sont* $\mapsto_{\beta\pi\mathsf{SP}}$.
— *Étant donnée un signature typée* (Σ, τ), *on appelle* λ-calcul typé avec produits sur (Σ, τ) *et on désigne par* $\boldsymbol{\lambda}_{\mathsf{ty}\times}(\mathcal{B}, \Sigma, \tau)$ *le système* $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_\times, \Sigma \uplus \Sigma_\pi, \tau \uplus \tau_\times)$.

Nous écrivons aussi $\boldsymbol{\lambda}_{\mathsf{ty}\times}(\mathcal{B})$ pour désigner $\boldsymbol{\lambda}_{\times\mathsf{ty}}(\mathcal{B})$.

Études non typées. Le système de réécriture $\mapsto_{\times\mathsf{SP}}$ a aussi été étudié dans un cadre non typé. Sa non confluence a été montrée par Klop dans sa thèse [Klo80] (voir aussi [Bar84] page 403 pour une courte histoire de la question).

Il est important de noter que la conversion reste consistante. Cela a été montré par Scott [Sco75] en utilisant une méthode sémantique (voir l'exercice 18.4.19 dans [Bar84]). Enfin, l'article [Vri89] montre avec des moyens syntaxiques que l'ajout des paires surjectives à la syntaxe du λ-calcul est conservatif : pour tout $t, u \in \Lambda$, si $t =_{\beta\pi\mathsf{SP}} u$ alors $t =_\beta u$. En particulier, cela entraîne la consistance de $=_{\beta\pi\mathsf{SP}}$.

3.7.2 Système T de Gödel

Nous avons vus en 3.4.4.(b) que le récurseur rec sur les entiers (voir l'exemple 3.1.17.(ii)) n'était pas définissable en temps borné dans le λ-calcul. Il semble donc approprié d'ajouter le système $\mapsto_{\mathsf{rec}}$ au λ-calcul.

Définition 3.7.3 L'extension entière *d'une signature* $\mathcal{B}$ *est* $\mathcal{B}_{\mathsf{Nat}} =_{def} \mathcal{B} \uplus \{\mathsf{Nat}\}$.

On type la signature Σ_{Nat} de la manière suivante :

$$\tau_{\mathsf{Nat}}(0) \quad = \quad \{\mathsf{Nat}\} \qquad\qquad \tau_{\mathsf{Nat}}(\mathsf{S}) \quad = \quad \{(\mathsf{Nat}, \mathsf{Nat})\}\ ,$$

et pour rec, on pose

$$\tau_{\mathsf{Nat}}(\mathsf{rec}) \quad =_{\mathrm{def}} \quad \{\mathsf{T} \Rightarrow (\mathsf{T} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{T}) \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{T} \mid \mathsf{T} \in \mathcal{T}_\Rightarrow(\mathcal{B}_{\mathsf{Nat}})\}\ ,$$

ce qui permet de typer les règles de réduction :

$$\begin{array}{rcl} x : \mathsf{T}, y : \mathsf{T} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{T} & \vdash_{\mathsf{Nat}} & \mathsf{rec}\ x\ y\ 0 \quad \mapsto_{\mathsf{rec}} \quad x \qquad\qquad : \mathsf{T} \\ z : \mathsf{Nat}, x : \mathsf{T}, y : \mathsf{T} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{T} & \vdash_{\mathsf{Nat}} & \mathsf{rec}\ x\ y\ \mathsf{S}(z) \quad \mapsto_{\mathsf{rec}} \quad y\ (\mathsf{rec}\ x\ y\ z)\ z\ : \mathsf{T} \end{array}$$

Nous verrons à l'exemple 5.1.5 que la relation $\to_{\mathsf{rec}}$ est confluente sur $\Lambda(\Sigma)$; de plus, les termes bien typés du système T sont fortement normalisants (voir par exemple [GLT89]).

Cette combinaison des récurseurs de type arbitraire sur les entiers au λ-calcul simplement typé est le *système T de Gödel*. Il permet de coder toutes les fonctions prouvablement totales de l'arithmétique de Peano du premier ordre [GLT89, Bar84].

Par exemple, la fonction $>$ définie en 3.1.11 peut être codée dans le système T :

$$\begin{array}{rcll} & >_{\mathsf{rec}}\ x\ y & =_{\mathrm{def}} & \mathsf{rec}\ \mathsf{false}\ (\mathsf{rec}\ (\lambda\_, \_.\mathsf{true})\ >_{\mathsf{aux}}\ y)\ x \\ \text{où} & >_{\mathsf{aux}} & =_{\mathrm{def}} & \lambda r.\lambda\_, \_.\lambda z.\ \mathsf{rec}\ \mathsf{false}\ r\ z \end{array}$$

On vérifie aisément que

$$\begin{array}{lll} >_{\mathsf{rec}}\ 0\ y & \to^{+}_{\beta\mathsf{rec}} & \mathsf{false} \\ >_{\mathsf{rec}}\ S(x)\ 0 & \to^{+}_{\beta\mathsf{rec}} & \mathsf{true} \\ >_{\mathsf{rec}}\ S(x)\ S(y) & \to^{+}_{\beta\mathsf{rec}} & >_{\mathsf{rec}}\ x\ y\ . \end{array}$$

3.7.3 Des co-produits à la réécriture d'ordre supérieur

Les duaux catégoriques des produits sont les *co-produits*. Les produits correspondent, par l'isomorphisme de Curry-Howard, au « et » logique, et les co-produits correspondent au « ou » intuitionniste.

Définition 3.7.4 *L'*extension co-produit *d'une signature* $\mathcal{B}$ *est* $\mathcal{B}_+ =_{def} \mathcal{B} \uplus \{\_ + \_\}$.

Voici comment définir le système. Soit $\mathsf{ty} \subseteq \{\Rightarrow, 2, \Lambda, \mathsf{Nat}, \times\}$.

— Les termes sont enrichis des constructions suivantes :

$$t, u \in \Lambda_{\mathsf{ty}}(\mathcal{B}_+, \Sigma) \quad ::= \quad \ldots \quad | \quad \mathsf{inj}_1 t \quad | \quad \mathsf{inj}_2 t \quad | \quad \mathsf{case}(t, x_1.t_1, x_2.t_2)\ .$$

Notons que dans $\mathsf{case}(t, x_1.t_1, x_2.t_2)$, les variables x_1 et x_2 sont liées respectivement dans t_1 et t_2. Les termes sont bien entendu quotientés par $=_\alpha$, dont la définition est celle de 1.2.6 augmentée du cas suivant :

$$\begin{array}{l} \mathsf{case}(t, x_1.t_1, x_2.t_2) \quad =_\alpha \quad \mathsf{case}(u, y_1.u_1, y_2.u_2) \\ \qquad\qquad \text{si } t =_\alpha u \text{ et pour tout } i \in \{1,2\}, \\ \qquad\qquad t_i[x_i \mapsto z] =_\alpha u_i[y_i \mapsto z] \text{ pour tout } z \in \mathcal{X} \text{ sauf un nombre fini.} \end{array}$$

— Les règles de typage sont les suivantes :

$$\frac{\Gamma \vdash t : T_i}{\Gamma \vdash \mathsf{inj}_i t : T_1 + T_2}\ (i \in \{1,2\}) \qquad \frac{\Gamma \vdash t : T_1 + T_2 \qquad \begin{array}{c}\Gamma, x : T_1 \vdash t_1 : U \\ \Gamma, x : T_2 \vdash t_2 : U\end{array}}{\Gamma \vdash \mathsf{case}(t, x_1.t_1, x_2.t_2) : U}$$

— Les règles de réduction sont les suivantes :

$$\begin{array}{lll} \mathsf{case}(\mathsf{inj}_1 u, x_1.t_1, x_2.t_2) & \mapsto_+ & t_1[u/x_1] \\ \mathsf{case}(\mathsf{inj}_2 u, x_1.t_1, x_2.t_2) & \mapsto_+ & t_2[u/x_2]\ . \end{array}$$

Les réductions des co-produits posent une question difficile à la réécriture : la définition d'un formalisme cadre pour les systèmes de réécriture avec variables liées. Cette ligne de recherche a donné lieu à de nombreux travaux :

— Les « Combinatory Reduction Systems » de Klop [Klo80] (voir aussi [KOR93]).
— La réécriture d'ordre supérieure de Wolfram [Wol93] et Nipkow [Nip91, NP98], étendue à la réécriture conditionnelle d'ordre supérieur [ALS94].
— Les systèmes d'ordre supérieur de van Oostrom et van Raamsdonk [OR94, Oos94, Raa96].
— Les « Expression Reduction Systems » de Z. Khasidashvili [GKK05]

L'enjeu de ces systèmes est de fournir un mécanisme de filtrage prenant en compte des variables liées dans les membres gauches de règles et permettant de gérer les paires critiques dues aux variables actives dans les membres gauche de règles. Nous ne nous sommes pas intéressés à ce problème difficile.

Chapitre 4

Réécriture conditionnelle

Ce chapitre concerne la réécriture conditionnelle. Nous commençons par les définitions de règle conditionnelle et de relation de réécriture conditionnelle et donnons quelques exemples. Ensuite, en utilisant les termes de de Bruijn, nous montrons que la réécriture conditionnelle sur $\Lambda(\Sigma)$ est le quotient par $=_\alpha$ de la réécriture conditionnelle sur $\mathcal{L}(\Sigma)$ modulo $=_\alpha$.

4.1 Définitions

Nous commençons par un exemple qui introduit les idées principales de la réécriture conditionnelle. On aimerait définir, par réécriture, une fonction filter paramétrée par une fonction booléenne $\mathtt{p}$, qui se comporte ainsi :

— $\mathsf{filter}\ \mathtt{p}\ [\,]$ se réécrit en $[\,]$,

— $\mathsf{filter}\ \mathtt{p}\ (\mathtt{t} :: \mathtt{ts})$ se réécrit en $\mathtt{t} :: (\mathsf{filter}\ \mathtt{p}\ \mathtt{ts})$ si $\mathtt{p}\ \mathtt{t}$ se réécrit en true,

— $\mathsf{filter}\ \mathtt{p}\ (\mathtt{t} :: \mathtt{ts})$ se réécrit en $\mathsf{filter}\ \mathtt{p}\ \mathtt{ts}$ si $\mathtt{p}\ \mathtt{t}$ se réécrit en false.

Cette spécification peut être écrite en utilisant des règles de réécriture conditionnelle ($\supset$ se lit « implique ») :

$$\begin{array}{lllll} & & \mathsf{filter}\ \mathtt{p}\ [\,] & \mapsto & [\,] \\ \mathtt{p}\ \mathtt{x} = \mathsf{true} & \supset & \mathsf{filter}\ \mathtt{p}\ (\mathtt{x} :: \mathtt{xs}) & \mapsto & \mathtt{x} :: (\mathsf{filter}\ \mathtt{p}\ \mathtt{xs}) \\ \mathtt{p}\ \mathtt{x} = \mathsf{false} & \supset & \mathsf{filter}\ \mathtt{p}\ (\mathtt{x} :: \mathtt{xs}) & \mapsto & \mathsf{filter}\ \mathtt{p}\ \mathtt{xs} \end{array} \tag{4.1}$$

Si nous essayons de définir une relation de réécriture $\to$ satisfaisant notre spécification, cela donne

$$\mathsf{filter}\ \mathtt{p}\ (\mathtt{t} :: \mathtt{ts}) \quad \to \quad \mathtt{t} :: (\mathsf{filter}\ \mathtt{p}\ \mathtt{ts}) \qquad \text{si} \qquad \mathtt{p}\ \mathtt{t} \ \to^*\ \mathsf{true}\ .$$

En d'autres termes, lors de la définition de $\to$ dans le pas

$$\mathsf{filter}\ \mathtt{p}\ (\mathtt{t} :: \mathtt{ts}) \quad \to \quad \mathtt{t} :: (\mathsf{filter}\ \mathtt{p}\ \mathtt{ts})\ ,$$

nous avons besoin de savoir si $\mathtt{p}\ \mathtt{t} \to^* \mathsf{true}$, donc d'utiliser la relation $\to$. Cette circularité est résolue par une définition inductive de la réécriture conditionnelle, dans laquelle la relation $\to$ est stratifiée en relations $(\to_\mathtt{i})_{\mathtt{i}\in\mathbb{N}}$, dont la correction est assurée par le théorème du point fixe de Tarski. On pose $\to =_{\text{def}} \bigcup_{\mathtt{i}\in\mathbb{N}} \to_\mathtt{i}$, avec, dans notre exemple, $\to_0 =_{\text{def}} \emptyset$ et $\to_{\mathtt{i}+1}$ définie en fonction de $\to_\mathtt{i}$ par

$$\mathsf{filter}\ \mathtt{p}\ (\mathtt{t} :: \mathtt{ts}) \ \to_{\mathtt{i}+1}\ \mathtt{t} :: (\mathsf{filter}\ \mathtt{p}\ \mathtt{ts}) \qquad \text{si} \qquad \mathtt{p}\ \mathtt{t} \ \to_\mathtt{i}^*\ \mathsf{true}\ .$$

Venons en aux définitions formelles. Nous autorisons, dans l'évaluation des conditions, l'utilisation d'une relation auxiliaire quelconque. De ce fait, dans le cas où $L \subseteq \mathcal{T}er(\Sigma, \mathcal{X})$, les définitions présentées ici sont des extensions de celles présentées dans [DO90, Ohl02]. De plus, notre définition de réécriture conditionnelle normale est plus large que celles que nous avons pu trouver dans la littérature.

Définition 4.1.1 (Règles de réécriture conditionnelle) *Soit* $L \subseteq \Lambda(\Sigma)$. *Une* règle de réécriture conditionnelle sur L *est un tuple écrit sous la forme*

$$d_1 = c_1 \ \wedge \ \ldots \ \wedge \ d_n = c_n \ \supset \ l \ \mapsto \ r$$

où $d_1, \ldots, d_n, c_1 \ldots, c_n, l, r \in L$.

Dans une règle de réécriture conditionnelle, on distingue

— *le* membre gauche l *(aussi appelé lhs),*

— *le* membre droit r *(aussi appelé rhs),*

— *les* conditions $d_1 = c_1 \ \wedge \cdots \wedge \ d_n = c_n$.

Nous supposons que les membres gauches sont des termes algébriques qui ne sont pas des variables et que $\mathsf{FV}(\vec{d}, \vec{c}, r) \subseteq \mathsf{FV}(l)$.

Si $r \in \mathcal{X}$, *alors* $\vec{d} = \vec{c} \supset l \mapsto r$ *est dite* effondrante.

Les règles de réécriture conditionnelle sont souvent appelées simplement « règles de réécriture ». Lorsque $|\vec{d}| = |\vec{c}| = 0$, la règle $\vec{d} = \vec{c} \supset l \mapsto r$ est dite « non conditionnelle ».

Notons que pour les TRS algébriques à gauche, la notion de règle non-conditionnelle effondrante définie en 3.1.16 est un cas particulier de règle conditionnelle effondrante.

Dans toute la suite, le symbole $\mathcal{R}$ désigne un ensemble de règles de réécriture. Les symboles définis (resp. constructeurs) de $\mathcal{R}$ sont les symboles définis (resp. constructeurs) du TRS $\{(l, r) \mid \vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r\}$.

Remarque 4.1.2 (Extension des remarques 2.2.2 et 2.4.2)

— *Les notions de règles de réécriture conditionnelle sur* $\Lambda(\Sigma)$, $\mathcal{T}er(\Sigma, \mathcal{X})$ *et* $\mathcal{T}er(\mathcal{X})$ *sont des conséquences immédiates de la définition 4.1.1.*

— *Si* $\mathcal{R}$ *est un ensemble de règles de réécriture sur* $L \subseteq \Lambda(\Sigma)$, *alors* $(L, \mathcal{R})$ *est un TRS au sens de la définition 2.4.1.*

— *L'extension de la notion de règle de réécriture conditionnelle à une* Σ*-algèbre libre* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *se fait comme pour les relations de réécriture en 2.2.2 et les TRS en 2.4.2 : on dit que* $\vec{d} = \vec{c} \supset l \mapsto r$ *est une règle de réécriture sur* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}})$ *si* $i(\vec{d}) = i(\vec{c}) \supset i(l) \mapsto i(r)$ *en est une sur* $\mathcal{T}er(\Sigma, \mathcal{X})$, *où* i *est l'isomorphisme canonique de* $\mathcal{T}er(\Sigma_{\mathcal{A}}, \mathcal{X}_{\mathcal{A}}) \to \mathcal{T}er(\Sigma, \mathcal{X})$.

Remarque 4.1.3 *Les conditions des règles conditionnelles ne sont pas symétriques : la condition* $d = c$ *n'est pas la même que la condition* $c = d$. *Cette distinction est justifiée par la possibilité de définir des relations de réécriture conditionnelles qui évaluent les conditions de façon asymétrique. C'est le cas par exemple, pour les règles définissant* filter *en 4.1, de la condition* $p\ x = \mathsf{true}$ *qui est évaluée en testant si* $p\ t \to^* \mathsf{true}$.

Les notions de système algébrique et de système applicatif à droite définies en 3.1.8 pour les TRS sont étendues de la manière suivante à la réécriture conditionnelle.

Définition 4.1.4 (Règles conditionnelles applicatives et algébriques) *Une règle conditionnelle* $\vec{d} = \vec{c} \supset l \mapsto r$ *est*

— applicative à droite *si* r *est applicatif,*
— applicative *si elle est applicative à droite et si les termes de* $\vec{d}, \vec{c}$ *sont applicatifs,*
— algébrique à droite *si* r *est algébrique,*
— algébrique *si elle est algébrique à droite et si les termes de* $\vec{d}, \vec{c}$ *sont algébriques.*

Un système de réécriture conditionnel $\mathcal{R}$ *est* applicatif à droite *(resp.* applicatif, algébrique à droite, algébrique*) si toutes ses règles le sont.*

Étant donné un ensemble $\mathcal{R}$ de règles de réécriture conditionnelles, on peut construire de nombreuses relations de réécritures selon la façon dont on évalue des conditions : par conversion, par joignabilité ou par réduction. Dans chaque cas on peut choisir d'autoriser des pas d'une autre relation de réécriture.

Définition 4.1.5 (Relations de réécriture conditionnelle) *Soient* $L \subseteq \Lambda(\Sigma)$, $\mathcal{R}$ *un ensemble de règles de réécriture conditionnelle sur* L *et* $\to_R$ *une relation de réécriture sur* L.

— *La relation de* réécriture R-conditionnelle semi-équationnelle issue de $\mathcal{R}$ *sur* L *est la relation* $\to_{\mathcal{R}(R)^{se}_L}$ *stratifiée par* $(\to_{\mathcal{R}(R)^{se}_{Li}})_{i\in\mathbb{N}}$, *où* $\to_{\mathcal{R}(R)^{se}_{L0}} = \emptyset$ *et pour tout* $i \in \mathbb{N}$, $\to_{\mathcal{R}(R)^{se}_{Li+1}}$ *est la plus petite relation sur* L *qui est close par contexte sur* L *et telle que, pour toute règle* $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, *pour toute substitution* $\sigma : \mathcal{X} \to L$,

$$si \quad \vec{d}\sigma \quad \leftrightarrow^*_{\mathcal{R}(R)^{se}_{Li} \cup R} \quad \vec{c}\sigma \quad alors \quad l\sigma \quad \to_{\mathcal{R}(R)^{se}_{Li+1}} \quad r\sigma \ .$$

— *La relation de* réécriture R-conditionnelle par joignabilité issue de $\mathcal{R}$ *sur* L *est la relation* $\to_{\mathcal{R}(R)^{jo}_L}$ *stratifiée par* $(\to_{\mathcal{R}(R)^{jo}_{Li}})_{i\in\mathbb{N}}$, *où* $\to_{\mathcal{R}(R)^{jo}_{L0}} = \emptyset$ *et pour tout* $i \in \mathbb{N}$, $\to_{\mathcal{R}(R)^{jo}_{Li+1}}$ *est la plus petite relation sur* L *qui est close par contexte sur* L *et telle que, pour toute règle* $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, *pour toute substitution* $\sigma : \mathcal{X} \to L$,

$$si \quad \vec{d}\sigma \quad \downarrow_{\mathcal{R}(R)^{jo}_{Li} \cup R} \quad \vec{c}\sigma \quad alors \quad l\sigma \quad \to_{\mathcal{R}(R)^{jo}_{Li+1}} \quad r\sigma \ .$$

— *La relation de* réécriture R-conditionnelle orientée issue de $\mathcal{R}$ *sur* L *est la relation* $\to_{\mathcal{R}(R)^{o}_L}$ *stratifiée par* $(\to_{\mathcal{R}(R)^{o}_{Li}})_{i\in\mathbb{N}}$, *où* $\to_{\mathcal{R}(R)^{o}_{L0}} = \emptyset$ *et pour tout* $i \in \mathbb{N}$, $\to_{\mathcal{R}(R)^{o}_{Li+1}}$ *est la plus petite relation sur* L *qui est close par contexte sur* L *et telle que, pour toute règle* $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, *pour toute substitution* $\sigma : \mathcal{X} \to L$,

$$si \quad \vec{d}\sigma \quad \to^*_{\mathcal{R}(R)^{o}_{Li} \cup R} \quad \vec{c}\sigma \quad alors \quad l\sigma \quad \to_{\mathcal{R}(R)^{o}_{Li+1}} \quad r\sigma \ .$$

Remarque 4.1.6 *Rappelons que le pas de réécriture conditionnelle, c'est-à-dire le fait de savoir si un terme se réécrit en un autre, n'est en général pas décidable [Kap84] (voir aussi [Ohl02]), même pour des systèmes de réécriture donnant lieu à des relations fortement normalisantes.*

Il est parfois pratique d'écrire « réécriture se-R-conditionnelle », « réécriture jo-R-conditionnelle » et « réécriture o-R-conditionnelle » au lieu de, respectivement, « réécriture

R-conditionnelle semi-équationnelle », « réécriture R-conditionnelle par joignabilité », et « réécriture R-conditionnelle orientée ». Lorsque $\to_R = \emptyset$, on écrit $\to_{\mathcal{R}^X}$ au lieu de $\to_{\mathcal{R}(R)^X}$ ($X \in \{se, jo, o\}$). Dans ce cas, nous parlons de « réécriture X-conditionnelle » au lieu de « réécriture X-R-conditionnelle ».

Nous devons vérifier que nous avons bien défini des relations de réécriture.

Proposition 4.1.7 *Soit $L \subseteq \Lambda(\Sigma)$, $\mathcal{R}$ un ensemble de règles de réécriture conditionnelle sur L et $\to_R$ une relation de réécriture sur L. Alors $\to_{\mathcal{R}(R)^X_{Li}}$ est une relation de réécriture sur L pour tout $X \in \{se, jo, o\}$, et tout $i \in \mathbb{N}$.*

PREUVE. Par induction sur $i \in \mathbb{N}$, en utilisant le lemme 2.2.3. □

Lorsque le contexte le permet, nous écrivons $\to_{\mathcal{R}^X_i}$ pour désigner $\to_{\mathcal{R}^X_{Li}}$. Une forme de réécriture conditionnelle importante en pratique est la réécriture conditionnelle *normale*. C'est un cas particulier de réécriture orientée et de réécriture par joignabilité.

Définition 4.1.8 (Relation de réécriture conditionnelle normale) *Soit $L \subseteq \Lambda(\Sigma)$, $\mathcal{R}$ un ensemble de règles de réécriture conditionnelle sur L, $\to_R$ une relation de réécriture sur L et $X \in \{jo, o\}$. Si pour toute règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ les termes $\vec{c}$ sont clos et en forme $\to_{\mathcal{R}(R)^X}$-normale, alors la relation de réécriture $\to_{\mathcal{R}(R)^X}$ est une relation de réécriture* R-conditionnelle normale.

La propriété suivante, dont la preuve est longue mais évidente, dit qu'il n'y a en fait qu'une seule relation de réécriture normale.

Proposition 4.1.9 *Soit $L \subseteq \Lambda(\Sigma)$, $\mathcal{R}$ un ensemble de règles de réécriture conditionnelle sur L et $\to_R$ une relation de réécriture sur L. Alors $\to_{\mathcal{R}(R)^{jo}}$ est R-normale si et seulement si $\to_{\mathcal{R}(R)^o}$ l'est aussi et dans ce cas $\to_{\mathcal{R}(R)^{jo}} = \to_{\mathcal{R}(R)^o}$.*

PREUVE. La preuve se fait par induction sur les stratifications de $\to_{\mathcal{R}(R)^{jo}}$ et $\to_{\mathcal{R}(R)^o}$. Pour $i \in \mathbb{N}$ et $X \in \{jo, o\}$, nous disons que $\to_{\mathcal{R}(R)^X_{i+1}}$ est normale si pour toute règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ les termes $\vec{c}$ sont clos et en forme $\to_{\mathcal{R}(R)^X_i}$-normale. Par convention, nous disons que $\to_{\mathcal{R}(R)^X_0}$ est toujours normale. Il est évident que si $\to_{\mathcal{R}(R)^X}$ est normale, alors $\to_{\mathcal{R}(R)^X_i}$ est normale pour tout $i \geq 0$.

— Tout d'abord, supposons que $\to_{\mathcal{R}(R)^{jo}}$ soit normale. On montre par induction sur $i \in \mathbb{N}$ que $\to_{\mathcal{R}(R)^o_i}$ est normale et que $\to_{\mathcal{R}(R)^o_i} \subseteq \to_{\mathcal{R}(R)^{jo}}$. Il s'en suit que $\to_{\mathcal{R}(R)^o} \subseteq \to_{\mathcal{R}(R)^{jo}}$, et donc que $\to_{\mathcal{R}(R)^o}$ est normale.
Le cas de base $i = 0$ est trivial parce que $\to_{\mathcal{R}(R)^{jo}_0} = \emptyset = \to_{\mathcal{R}(R)^o_0}$.
Supposons la propriété vraie pour $i \geq 0$, et montrons que $\to_{\mathcal{R}(R)^o_{i+1}}$ est normale.
Pour toute règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, comme les termes $\vec{c}$ sont en forme $\to_{\mathcal{R}(R)^{jo}}$-normale, ils sont en forme en forme $\to_{\mathcal{R}(R)^o_i}$-normale par hypothèse d'induction. Il s'en suit que $\to_{\mathcal{R}(R)^o_{i+1}}$ est normale.
Maintenant, si $t \to_{\mathcal{R}(R)^o_{i+1}} u$, alors il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, une substitution $\sigma : \mathcal{X} \to L$ et un contexte $C[\,] : L \to L$ tels que $t = C[l\sigma]$, $u = C[r\sigma]$ et $\vec{d}\sigma \to^* \vec{c}$. Par hypothèse d'induction, $\vec{d}\sigma \to^*_{\mathcal{R}(R)^{jo}} \vec{c}$, donc $C[l\sigma] \to_{\mathcal{R}(R)^{jo}} C[r\sigma]$.

— La propriété inverse, qui dit que si $\to_{\mathcal{R}(R)^o}$ est normale alors pour tout $i \in \mathbb{N}$, la relation $\to_{\mathcal{R}(R)^{jo}_i}$ est normale et $\to_{\mathcal{R}(R)^{jo}_i} \subseteq \to_{\mathcal{R}(R)^o}$ u, se montre exactement de la même façon. □

Il y a donc au plus une relation de réécriture normale sur un système conditionnel $\mathcal{R}$. Nous la notons $\to_{\mathcal{R}(R)^!}$.

La réécriture normale est la plus facile à manipuler parce que l'application d'une règle ne nécessite que le test de normalisation, qui peut de plus être fait de façon aveugle lorsque la relation de réécriture est confluente. Notons toutefois qu'avec notre définition, le problème de savoir si un système est normal n'est pas décidable parce qu'en général, le pas de réécriture conditionnelle n'est pas décidable. Les définitions courantes de réécriture normale, comme par exemple celle de [Ohl02], sont en fait des restrictions décidables de notre définition.

4.2 Exemples

Nous donnons ici quelques exemples de systèmes de réécriture conditionnelle.

4.2.1 Un système de manipulation de termes

Notre premier exemple est l'adaptation d'un programme CAML de [Hue86]. Il définit des fonctions permettant, dans un terme représenté comme un arbre dont les successeurs sont des listes de nœuds, de faire le remplacement d'un sous-terme à une certaine occurrence par un autre sous-terme.

Ce système doit être lu en pensant à la combinaison de la réécriture conditionnelle et du λ-calcul. Nous avons deux combinaisons en tête :

— la réécriture conditionnelle par joignabilité, que nous notons $\to_{\mathcal{R}}$,

— la réécriture β-conditionnelle par joignabilité, que nous notons $\to_{\mathcal{R}(\beta)}$.

Commençons par une fonction apply telle que apply f n l applique f au n-ème élément de l. Elle utilise la fonction auxiliaire app.

$$
\begin{array}{llllll}
> (\mathsf{length}\ \mathsf{l})\ \mathsf{n} & = & \mathsf{true} & \supset & \mathsf{apply}\ \mathsf{f}\ \mathsf{n}\ \mathsf{l} & \mapsto_{\mathsf{apply}}\ \mathsf{app}\ \mathsf{f}\ \mathsf{n}\ \mathsf{l} \\
> (\mathsf{length}\ \mathsf{l})\ \mathsf{n} & = & \mathsf{false} & \supset & \mathsf{apply}\ \mathsf{f}\ \mathsf{n}\ \mathsf{l} & \mapsto_{\mathsf{apply}}\ \mathsf{err} \\
 & & & & & \\
 & & & & \mathsf{app}\ \mathsf{f}\ 0\ \mathsf{l} & \mapsto_{\mathsf{apply}}\ (\mathsf{f}\ (\mathsf{car}\ \mathsf{l})) :: (\mathsf{cdr}\ \mathsf{l}) \\
 & & & & \mathsf{app}\ \mathsf{f}\ (\mathsf{s}\ \mathsf{n})\ \mathsf{l} & \mapsto_{\mathsf{apply}}\ (\mathsf{car}\ \mathsf{l}) :: (\mathsf{app}\ \mathsf{f}\ \mathsf{n}\ (\mathsf{cdr}\ \mathsf{l}))
\end{array}
$$

On représente les termes du premier ordre comme des arbres avec des noeud node y l où y est l'étiquette et l la liste des successeurs. Les positions sont des listes d'entiers et la fonction occ appliquée aux arguments u et t teste si u est une occurrence de t. Nous la définissons comme suit :

$$
\begin{array}{llllll}
 & & & & \mathsf{occ}\ [\ ]\ \mathsf{t} & \mapsto_{\mathsf{occ}}\ \mathsf{true} \\
> (\mathsf{length}\ \mathsf{l})\ \mathsf{x} & = & \mathsf{false} & \supset & \mathsf{occ}\ (\mathsf{x} :: \mathsf{o})\ (\mathsf{node}\ \mathsf{y}\ \mathsf{l}) & \mapsto_{\mathsf{occ}}\ \mathsf{false} \\
> (\mathsf{length}\ \mathsf{l})\ \mathsf{x} & = & \mathsf{true} & \supset & \mathsf{occ}\ (\mathsf{x} :: \mathsf{o})\ (\mathsf{node}\ \mathsf{y}\ \mathsf{l}) & \mapsto_{\mathsf{occ}}\ \mathsf{occ}\ \mathsf{o}\ (\mathsf{get}\ \mathsf{l}\ \mathsf{x})
\end{array}
$$

La fonction replace t o s remplace par s le sous-terme de t à l'occurrence o :

$$\begin{array}{lllll} \mathsf{occ}\ u\ t & = & \mathsf{true} & \supset & \mathsf{replace}\ t\ o\ s \quad \mapsto_{\mathsf{replace}} \quad \mathsf{rep}\ t\ o\ s \\ \mathsf{occ}\ u\ t & = & \mathsf{false} & \supset & \mathsf{replace}\ t\ o\ s \quad \mapsto_{\mathsf{replace}} \quad \mathsf{err} \end{array}$$

$$\begin{array}{lll} \mathsf{rep}\ t\ [\,]\ s & \mapsto_{\mathsf{replace}} & s \\ \mathsf{rep}\ (\mathsf{node}\ y\ l)\ (x :: o)\ s & \mapsto_{\mathsf{replace}} & \mathsf{node}\ y\ (\mathsf{apply}\ (\lambda z.\mathsf{rep}\ z\ o\ s)\ x\ l) \end{array}$$

Les symboles cdr, car, length et get sont définis par les systèmes $\mapsto_{\mathsf{cdr}}$, $\mapsto_{\mathsf{car}}$, $\mapsto_{\mathsf{length}}$ et $\mapsto_{\mathsf{get}}$ respectivement, présentés en 3.1.12 ; et le symbole $>$ est défini par le système $\mapsto_{>}$ présenté en 3.1.11. Le système

$$\mathsf{Tree} \quad =_{\mathrm{def}} \quad \{\mapsto_{\mathsf{car}}, \mapsto_{\mathsf{cdr}}, \mapsto_{\mathsf{get}}, \mapsto_{\mathsf{length}}, \mapsto_{\mathsf{occ}}\}$$

est algébrique. Les règles de apply et de app sont applicatives à droite et celles de filter contiennent dans leurs conditions une variables p en position active. Cette définition de rep utilise une λ-abstraction dans un membre droit.

4.2.2 Règles conditionnelles de de Vrijer

Dans [Vri89], de Vrijer utilise la réécriture conditionnelle semi-équationnelle pour obtenir un calcul ayant la même conversion que le λ-calcul avec paires surjectives. Les règles sont celles de $\mapsto_{\times}$ plus

$$\pi_2 x = y \supset (\pi_1 x, y) \mapsto_{lr} x \qquad \pi_1 x = y \supset (y, \pi_2 x) \mapsto_{lr} x\ .$$

La relation $\to_{\beta(\times lr)^{se}(\beta)}$ (union de $\to_\beta$ et de la réécriture β-conditionnelle semi-équationnelle pour le système $\mapsto_{\times lr}$) est confluente modulo une certaine relation d'équivalence, ce qui permet à de Vrijer d'en déduire que $=_{\beta\pi SP}$ est une extension conservative de $=_\beta$:

$$\forall t, u \in \Lambda\ .\quad \lambda_{\Rightarrow\times SP} \vdash_{\Rightarrow\times} t =_{\beta\pi SP} u \qquad \text{si et seulement si} \qquad \lambda_{\Rightarrow} \vdash_{\Rightarrow} t =_\beta u\ .$$

4.3 Quotient de la réécriture conditionnelle par alpha-conversion

Dans cette section, nous utilisons les termes de de Bruijn pour montrer que la réécriture *applicative* sur $\Lambda(\Sigma)$ est le quotient par α-équivalence de la réécriture sur $\mathcal{L}(\Sigma)$ modulo α-équivalence.

Comme nous nous focalisons sur le cas de la réécriture applicative, nous n'avons pas besoin d'utiliser le formalisme développé dans [BKR05a] pour traiter la réécriture d'ordre supérieur avec des indices de de Bruijn, ni d'utiliser la traduction étudiée dans [BKR05b] de la réécriture d'ordre supérieur vers la réécriture du premier ordre modulo une théorie équationnelle générée par un calcul de substitution.

Pour tout $X \in \{se, \mathfrak{jo}, o\}$, nous dénotons

— par $\to_{\mathcal{R}^X_\Lambda}$ la relation de réécriture X-conditionnelle issue de $\mathcal{R}$ sur $\Lambda(\Sigma)$,

— par $\to_{\mathcal{R}^X_{\mathcal{L}}}$ la relation de réécriture X-conditionnelle issue de $\mathcal{R}$ sur $\mathcal{L}(\Sigma)$,
— par $\to_{\mathcal{R}^X_{\mathsf{dB}}}$ la relation de réécriture X-conditionnelle issue de $\mathcal{R}$ sur $\mathcal{T}\mathsf{dB}(\Sigma)$.

Du fait que $\Lambda(\Sigma) = (\mathcal{L}(\Sigma)/=_\alpha)$, il naturel de vouloir avoir

$$\forall i \in \mathbb{N}\ .\quad [t] \to_{\mathcal{R}^X_{\Lambda_i}} [u] \qquad \text{si et seulement si} \qquad t \to_{\mathcal{R}^X_{\mathcal{L}_i}} u\ . \tag{4.2}$$

Le sens « si » ne pose pas de problèmes.

Lemme 4.3.1 *Soient* $t, u \in \mathcal{L}(\Sigma)$. *Pour tout* $i \in \mathbb{N}$,

$$t \to_{\mathcal{R}^X_{\mathcal{L}_i}} u \qquad \textit{implique} \qquad [t] \to_{\mathcal{R}^X_{\Lambda_i}} [u]\ .$$

Preuve. On raisonne par induction sur $i \in \mathbb{N}$. On ne détaille que le cas de la réécriture par joignabilité.

Le cas de base $i = 0$ est trivial. Soit $i \in \mathbb{N}$ et supposons la propriété vraie pour $\to_{\mathcal{R}^X_{\mathcal{L}_i}}$. Il est clair que cela implique qu'elle est vraie pour $\to^*_{\mathcal{R}^X_{\mathcal{L}_i}}$. Par induction sur t, on montre que $t \to_{\mathcal{R}^X_{\mathcal{L}_{i+1}}} u$ implique $[t] \to_{\mathcal{R}^X_{\Lambda_{i+1}}} [u]$.

$t = \lambda x.t_1$. Dans ce cas, $u = \lambda x.u_1$ et ne $t_1 \to_{\mathcal{R}^X_{\mathcal{L}_{i+1}}} u_1$, donc par hypothèse d'induction $[t_1] \to_{\mathcal{R}^X_{\Lambda_{i+1}}} [u_1]$ et $\lambda x.[t_1] \to_{\mathcal{R}^X_{\Lambda_{i+1}}} \lambda x.[u_1]$, ce qui conclut la démonstration car $[\lambda x.t_1] = \lambda x.[t_1]$ et $[\lambda x.u_1] = \lambda x.[u_1]$.

$t = t_1 t_2$. Si $u = u_1 u_2$ avec $(t_1, t_2) \to_{\mathcal{R}^X_{\mathcal{L}_{i+1}}} (u_1, u_2)$ alors on conclut directement par hypothèse d'induction et le fait que $[t_1 t_2] = [t_1][t_2]$ et $[u_1 u_2] = [u_1][u_2]$.
Sinon, il y une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ telle que $t = l\sigma$, $u = r\sigma$ et il existe $\vec{v}$ tels que $\vec{d}\sigma \to^*_{\mathcal{R}^X_{\mathcal{L}_i}} \vec{v}$ et $\vec{c}\sigma \to^*_{\mathcal{R}^X_{\mathcal{L}_i}} \vec{v}$. Par hypothèse d'induction, $[\vec{d}][\sigma] \to^*_{\mathcal{R}^X_{\Lambda_i}} \vec{v}$ $[\vec{c}][\sigma] \to^*_{\mathcal{R}^X_{\Lambda_i}} \vec{v}$, et on en déduit que $[l\sigma] \to_{\mathcal{R}^X_{\Lambda_{i+1}}} [r\sigma]$ (rappelons que toutes les règles de $\mathcal{R}$ sont supposées applicatives).

$t = f(t_1, \ldots, t_n)$. Idem. □

Par contre, la propriété inverse :

$$[t] \to^*_{\mathcal{R}^X_{\Lambda}} [u] \qquad \text{implique} \qquad t \to^*_{\mathcal{R}^X_{\mathcal{L}}} u$$

n'est pas vraie en général, même pour la réécriture non conditionnelle. Le problème de base est qu'en présence de règles non-linéaires à gauche, la réécriture sur $\to_{\mathcal{R}^X_{\mathcal{L}}}$ ne commute pas avec $=_\alpha$: on peut avoir $t' =_\alpha t \to_{\mathcal{R}_{\mathcal{L}}} u$ sans qu'il existe u' tel que $t' \to_{\mathcal{R}_{\mathcal{L}}} u' =_\alpha u$.

Exemple 4.3.2 *Avec la règle non-linéaire* $\mathsf{minus}\ x\ x \mapsto 0$, *issue du système* $\mapsto_{\mathsf{minus}}$ *présenté en 3.1.11, on a*

$$\begin{array}{ccc} \mathsf{minus}\ \lambda x.x\ \lambda x.x & =_\alpha & \mathsf{minus}\ \lambda x.x\ \lambda y.y \\ \Big\downarrow \mathcal{L} & & \\ 0 & & \end{array}$$

Comme le terme $\mathsf{minus}\ \lambda x.x\ \lambda y.y$ *est une forme* $\to_{\mathcal{L}}$*-normale, il n'existe pas de terme* u' *tel que* $0 =_\alpha u' \leftarrow_{\mathcal{L}} \mathsf{minus}\ \lambda x.x\ \lambda y.y$.

De ce fait, lorsque

$$[t] \quad \to_{\mathcal{R}^X_\Lambda} \quad [u] \ ,$$

il est possible que le terme t ne soit pas le bon représentant de $[t]$ pour avoir $t \to^*_{\mathcal{R}^X_\mathcal{L}} u$.

Exemple 4.3.3 *Comme* $\lambda x.x =_\alpha \lambda y.y$*, on a*

$$\mathsf{minus}\ [\lambda x.x]\ [\lambda y.y] \quad \to_\Lambda \quad 0 \ ,$$

mais le terme $\mathsf{minus}\ \lambda x.x\ \lambda y.y$ *est* $\to_\mathcal{L}$*-normal.*

Ainsi, lorsque $[t] \to_{\mathcal{R}^X_\Lambda} [u]$, on peut au mieux espérer que

$$\exists t', u' \ . \quad t =_\alpha t' \to_{\mathcal{R}^X_{\mathcal{L}_i}} u' =_\alpha u \ .$$

Ceci correspond à la réécriture *modulo* α*-conversion* sur $\mathcal{L}(\Sigma)$, au sens de [Hue80, JK86] (voir aussi la section 2.5 de [DJ90] et la définition 2.5.1 de [Ohl02]).

Définition 4.3.4 (Réécriture modulo alpha-conversion) *Soit* $X \in \{se, jo, o\}$ *et* $\mathcal{R}$ *un système conditionnel applicatif. Pour tout* $i \in \mathbb{N}$*, la relation* $\to_{\mathcal{R}^X_{\mathcal{L}_i}/=_\alpha} \subseteq \mathcal{L}(\Sigma) \times \mathcal{L}(\Sigma)$ *est définie de la manière suivante :*

$$t \to_{\mathcal{R}^X_{\mathcal{L}_i}/=_\alpha} u \qquad \Longleftrightarrow_{def} \qquad \exists t', u' \ . \quad t =_\alpha t' \to_{\mathcal{R}^X_{\mathcal{L}_i}} u' =_\alpha u \ .$$

Nous allons maintenant montrer que la relation $\to_{\mathcal{R}^X_\mathcal{L}/=_\alpha}$ est la bonne notion de réécriture sur $\mathcal{L}(\Sigma)$ pour avoir une correspondance naturelle avec la réécriture sur $\Lambda(\Sigma)$:

$$\forall i \in \mathbb{N} \ . \quad [t] \to_{\mathcal{R}^X_{\Lambda_i}} [u] \qquad \text{si et seulement si} \qquad t \to_{\mathcal{R}^X_{\mathcal{L}_i}/=_\alpha} u \ . \tag{4.3}$$

Afin de passer de $\to_{\mathcal{R}^X_\Lambda}$ à $\to_{\mathcal{R}^X_\mathcal{L}/=_\alpha}$, il faut pouvoir choisir les bons représentants des classes d'α-équivalence. Pour cela, les termes de de Bruijn semblent intéressants.

Exemple 4.3.5 *Pour tout référentiel* R*, on a*

$$(\mathsf{minus}\ \lambda x.x\ \lambda y.y)_R \quad = \quad \mathsf{minus}\ (\lambda 0)\ (\lambda 0) \quad \to_{dB} \quad S(0) \ .$$

En utilisant les correspondances entre $\Lambda(\Sigma)$, $\mathcal{L}_B(\Sigma)$ et $\mathcal{T}dB(\Sigma)$ établies à la section 1.6, nous montrons les équivalences suivantes : pour tout $t, u \in \mathcal{L}_B(\Sigma)$, et tout référentiel R pour t et u,

$$\forall i \in \mathbb{N} \ . \quad [t] \to_{\mathcal{R}^X_{\Lambda_i}} [u] \quad \Longleftrightarrow \quad t_R \to_{\mathcal{R}^X_{dB_i}} u_R \quad \Longleftrightarrow \quad (t_R)^R \to_{\mathcal{R}^X_{\mathcal{L}_i}} (u_R)^R \ . \tag{4.4}$$

Grâce au théorème 1.6.9, il est alors aisé d'en déduire (4.3).

L'exemple 4.3.5 se généralise à un système conditionnel applicatif quelconque.

Lemme 4.3.6 *Soit* $\mathcal{R}$ *un ensemble de règles conditionnelles applicatives sur* $\mathcal{L}(\Sigma)$ *et* $X \in \{se, jo, o\}$*. Soient* $t, u \in \mathcal{L}_B(\Sigma)$ *et* R *un référentiel pour* t *et* u*. Pour tout* $i \in \mathbb{N}$*,*

$$[t] \to_{\mathcal{R}^X_{\Lambda_i}} [u] \qquad \textit{implique} \qquad t_R \to_{\mathcal{R}^X_{dB_i}} u_R \ .$$

PREUVE. On raisonne par induction sur $i \in \mathbb{N}$.

Le cas de base $i = 0$ est trivial. Soit $i \in \mathbb{N}$ et supposons la propriété vraie pour $\to_{\mathcal{R}^X_{\Lambda i}}$. Il est clair qu'elle est alors vraie pour $\to^*_{\mathcal{R}^X_{\Lambda i}}$.

Montrons que $[t] \to_{\mathcal{R}^X_{\Lambda i+1}} [u]$ implique $t_R \to_{\mathcal{R}^X_{dBi+1}} u_R$. On raisonne par induction sur t et on ne détaille la preuve que pour la réécriture la joignabilité.

$t = \lambda x.t_1$. Dans ce cas, $u = \lambda x.u_1$, et comme $[\lambda x.t_1] = \lambda x.[t_1]$ et $[\lambda x.u_1] = \lambda x.[u_1]$, on a $[t_1] \to_{\mathcal{R}^X_{\Lambda i+1}} [u_1]$. Comme $t_1, u_1 \in \mathcal{L}_B(\Sigma)$, $x \notin BV(t_1, u_1)$, donc $x \cdot R$ est un référentiel pour t_1 et u_1. Par hypothèse d'induction, on a donc $t_{1\,x\cdot R} \to_{\mathcal{R}^X_{dBi+1}} u_{1\,x\cdot R}$, d'où $\lambda t_{1\,x\cdot R} \to_{\mathcal{R}^X_{dBi+1}} \lambda u_{1\,x\cdot R}$, et on en déduit que $(\lambda x.t_1)_R \to_{\mathcal{R}^X_{dBi+1}} (\lambda x.u_1)_R$ car $(\lambda x.t_1)_R = \lambda t_{1\,x\cdot R}$ et $(\lambda x.u_1)_R = \lambda u_{1\,x\cdot R}$.

$t = t_1 t_2$. Si $u = u_1 u_2$ avec $([t_1], [t_2]) \to_{\mathcal{R}^X_{\Lambda i+1}} ([u_1], [u_2])$, alors on conclut grâce à l'hypothèse d'induction car $[t_1 t_2] = [t_1][t_2]$ et $[u_1 u_2] = [u_1][u_2]$.

Sinon, il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et une substitution $[\sigma]$ telles que $[t] = [l][\sigma]$, $[u] = [r][s]$ et il existe $\vec{v}$ tels que $[\vec{d}][\sigma] \to^*_{\mathcal{R}^X_{\Lambda i}} [\vec{v}]$ et $[\vec{c}][\sigma] \to^*_{\mathcal{R}^X_{\Lambda i}} [\vec{v}]$.

Comme $\mathcal{R}$ est applicative, on a $[\vec{d}][\sigma] = [\vec{d}\sigma]$, $[\vec{c}][\sigma] = [\vec{c}\sigma]$, d'où, par hypothèse d'induction sur i, $(\vec{d}\sigma)_R \downarrow^*_{\mathcal{R}^X_{dBi}} (\vec{c}\sigma)_R$, soit $\vec{d}(\sigma_R) \downarrow^*_{\mathcal{R}^X_{dBi}} \vec{c}(\sigma_R)$ par 1.6.10.(ii). On a donc $l(\sigma_R) \to_{\mathcal{R}^X_{dBi+1}} r(\sigma_R)$, c'est-à-dire $(l\sigma)_R \to_{\mathcal{R}^X_{dBi+1}} (r\sigma)_R$.

$t = f(t_1, \ldots, t_n)$. Idem. □

On passe tout aussi facilement de $\to_{\mathcal{R}_{dB}}$ à $\to_{\mathcal{R}_{\mathcal{L}}}$.

Lemme 4.3.7 *Soit $\mathcal{R}$ un ensemble de règles conditionnelles applicatives sur $\mathcal{L}(\Sigma)$ et $X \in \{se, \mathfrak{jo}, o\}$. Soient $t, u \in \mathcal{T}dB(\Sigma)$ et R un référentiel pour t et u. Pour tout $i \in \mathbb{N}$,*

$$t \to_{\mathcal{R}^X_{dBi}} u \qquad \textit{implique} \qquad t^R \to_{\mathcal{R}^X_{\mathcal{L}i}} u^R .$$

PREUVE. On raisonne par induction sur $i \in \mathbb{N}$.

Le cas de base $i = 0$ est trivial. Soit $i \in \mathbb{N}$ et supposons la propriété vraie pour $\to_{\mathcal{R}^X_{dBi}}$. Notons qu'elle est alors vraie pour $\to^*_{\mathcal{R}^X_{dBi}}$, la clôture réflexive et transitive de $\to_{\mathcal{R}^X_{dBi}}$.

On montre que $t \to_{\mathcal{R}^X_{dBi+1}} u$ implique $t^R \to_{\mathcal{R}^X_{\mathcal{L}i+1}} u^R$. On raisonne par induction sur t et on ne détaille la preuve que pour la réécriture par joignabilité.

$t = \lambda t_1$. Dans ce cas on a $u = \lambda u_1$ avec $t_1 \to_{\mathcal{R}^X_{dBi+1}} u_1$. Comme $R' =_{def} x_{|R|+1} \cdot R$ est un référentiel pour t_1 et u_1, par hypothèse d'induction on a ${t_1}^{R'} \to_{\mathcal{R}^X_{\mathcal{L}i+1}} {u_1}^{R'}$, d'où $\lambda x_{|R|+1}.{t_1}^{R'} \to_{\mathcal{R}^X_{\mathcal{L}i+1}} \lambda x_{|R|+1}.{u_1}^{R'}$. On a donc $(\lambda t_1)^R \to_{\mathcal{R}^X_{\mathcal{L}i+1}} (\lambda u_1)^R$ car

$$(\lambda t_1)^R \quad = \quad \lambda x_{|R|+1}.{t_1}^{R'} \qquad \text{et} \qquad (\lambda u_1)^R \quad = \quad \lambda x_{|R|+1}.{u_1}^{R'} .$$

$t = t_1 t_2$. Si $u = u_1 u_2$ avec $(t_1, t_2) \to_{\mathcal{R}^X_{dBi+1}} (u_1, u_2)$ alors on conclut directement par l'hypothèse d'induction car $(t_1 t_2)^R = {t_1}^R {t_2}^R$ et $(u_1 u_2)^R = {u_1}^R {u_2}^R$.

Sinon il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et une substitution σ telles que $t = l\sigma$, $u = r\sigma$ et il existe $\vec{v}$ tels que $\vec{d}\sigma \to^*_{\mathcal{R}^X_{dBi}} \vec{v}$ et $\vec{c}\sigma \to^*_{\mathcal{R}^X_{dBi}} \vec{v}$. Par hypothèse d'induction

sur i on obtient $(\vec{d}\sigma)^R \to^*_{\mathcal{R}^X_{\mathcal{L}i}} (\vec{v})^R$ et $(\vec{c}\sigma)^R \to^*_{\mathcal{R}^X_{\mathcal{L}i}} (\vec{v})^R$. Comme $\mathcal{R}$ est applicative, par 1.6.10.(i) on a $(\vec{d}\sigma)^R = \vec{d}(\sigma^R)$, $(\vec{c}\sigma)^R = \vec{c}(\sigma^R)$, $l(\sigma^R) = (l\sigma)^R$ et $r(\sigma^R) = (r\sigma)^R$, d'où $(l\sigma)^R \to_{\mathcal{R}^X_{\mathcal{L}i+1}} (r\sigma)^R$.

$t = f(t_1, \dots, t_n)$. Idem. □

On en déduit que les équivalences (4.4) sont vérifiés sur les termes de $\mathcal{L}_B(\Sigma)$ (c'est-à-dire satisfaisant la convention de Barendregt locale 1.5.1).

Théorème 4.3.8 *Soit $\mathcal{R}$ un ensemble de règles conditionnelles applicatives sur $\mathcal{L}(\Sigma)$ et $X \in \{se, jo, o\}$. Étant donnés $t, u \in \mathcal{L}_B(\Sigma)$, et R un référentiel pour t et u, pour tout $i \in \mathbb{N}$, on a*

$$[t] \to_{\mathcal{R}^X_{\Lambda i}} [u] \quad \iff \quad t_R \to_{\mathcal{R}^X_{dBi}} u_R \quad \iff \quad (t_R)^R \to_{\mathcal{R}^X_{\mathcal{L}i}} (u_R)^R \ .$$

Preuve. $[t] \to_{\mathcal{R}^X_{\Lambda i}} [u]$ implique $t_R \to_{\mathcal{R}^X_{dBi}} u_R$ par le lemme 4.3.6, qui à son tour implique $(t_R)^R \to_{\mathcal{R}^X_{\mathcal{L}i}} (u_R)^R$ par le lemme 4.3.7. On en déduit $[(t_R)^R] \to_{\mathcal{R}^X_{\Lambda i}} [(u_R)^R]$ par le lemme 4.3.1, soit $[t] \to_{\mathcal{R}^X_{\Lambda i}} [u]$, car $(t_R)^R =_\alpha t$ et $(u_R)^R =_\alpha u$ par le théorème 1.6.9.(ii). □

Grâce aux termes de de Bruijn, on peut donc montrer que $\to_{\mathcal{R}_\Lambda}$ est bien le quotient de $\to_{\mathcal{R}_{\mathcal{L}/=_\alpha}}$ par α-équivalence. Notons que nous prouvons (4.3) sur $\mathcal{L}(\Sigma)$, bien que le théorème 4.3.8 ne concerne que les termes de $\mathcal{L}_B(\Sigma)$.

Corollaire 4.3.9 (Quotient de $\to_{\mathcal{R}_{\mathcal{L}/=_\alpha}}$ par $=_\alpha$) *Soit $X \in \{se, jo, o\}$ et $\mathcal{R}$ un système conditionnel applicatif. Pour tout $t, u \in \mathcal{L}(\Sigma)$, on a*

$$\forall i \in \mathbb{N} \ . \quad [t] \quad \to_{\mathcal{R}^X_{\Lambda i}} \quad [u] \quad \textit{si et seulement si} \quad t \quad \to_{\mathcal{R}^X_{\mathcal{L}i}/=_\alpha} \quad u \ .$$

Preuve. Le sens « si » est le lemme 4.3.1. Montrons le sens inverse. Soient $t, u \in \mathcal{L}(\Sigma)$ tels que $[t] \to_{\mathcal{R}^X_{\Lambda i}} [u]$. Par la proposition 1.5.2.(i), il existe $t', u' \in \mathcal{L}_B(\Sigma)$ tels que $t' =_\alpha t$ et $u' =_\alpha u$. On a donc $[t'] \to_{\mathcal{R}^X_{\Lambda i}} [u']$, d'où $(t'_R)^R \to_{\mathcal{R}^X_{\mathcal{L}i}} (u'_R)^R$ par le théorème 4.3.8. Par le théorème 1.6.9.(ii), on a $(t'_R)^R =_\alpha t'$ et $(u'_R)^R =_\alpha t'$. On en déduit que $t \to_{\mathcal{R}^X_{\mathcal{L}i}/=_\alpha} u$. □

Deuxième partie

Confluence

Chapitre 5

Outils pour la confluence

Dans ce chapitre, nous présentons certains outils de base pour l'étude de la confluence. Nous commençons par quelques rappels sur l'utilisation des relations de réécriture parallèles avec les TRS orthogonaux et avec le λ-calcul.

Ensuite, nous montrons comment ces notions sont étendues à la réécriture conditionnelle, ce qui nous permet d'aborder brièvement la différence entre confluence et confluence par niveaux.

Enfin, nous rassemblons quelques points importants liés à la modularité, en particulier la caractérisation de la confluence par la confluence close et un lien entre les termes infinis et un exemple du fait que la confluence n'est en général pas préservée par combinaison de systèmes non disjoints.

5.1 Systèmes orthogonaux et réécriture parallèle

Nous rappelons ici les formulations des preuves de confluence pour les systèmes orthogonaux utilisant les relations de réécriture parallèles. Nous commençons par les TRS au premier ordre, puis nous abordons le cas du λ-calcul.

5.1.1 Systèmes de réécriture de termes

Nous rappelons ici le résultat connu de la confluence des systèmes des réécriture orthogonaux au premier ordre. À notre connaissance, ce résultat a été prouvé pour la première fois dans [Hue80]. Des propriétés plus fines sur l'espace des dérivations sont étudiées dans [HL91] (nous ne les abordons pas ici). Pour les systèmes d'ordre supérieur, le résultat le plus général est probablement [OR94].

Le seul obstacle possible à la confluence des systèmes de réécriture de termes est la présence de paires critiques : quand deux membres gauches de règles peuvent se superposer à des positions non triviales, on peut choisir de réduire l'un ou l'autre rédex. Ce choix peut être irréversible, et casser la confluence.

Notons, pour les TRS dénombrables, que grâce à la proposition 2.4.4 on peut toujours se ramener à un TRS dont les règles n'ont pas de variables en commun, tout en préservant la relation de réécriture.

D'autre part, rappelons que d'après ce que l'on a vu à la section 1.4, les termes applicatifs sur une signature Σ sont générés par la grammaire :

$$t, u \in \mathcal{T}er(\Sigma_{App}, \mathcal{X}) \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n) \quad | \quad t\ u\ ,$$

où $x \in \mathcal{X}$ et $f \in \Sigma_n$. De plus, d'après la définition 3.1.7, les termes algébriques sur Σ sont générés par la grammaire suivante :

$$t \quad ::= \quad x \quad | \quad f(t_1, \ldots, t_n)t_{n+1} \ldots t_{n+m} \ ,$$

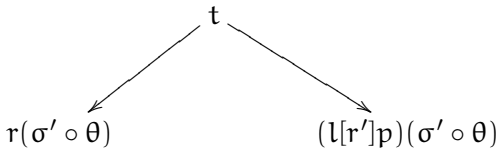
où $x \in \mathcal{X}$, $f \in \Sigma_n$ et $m \geq 0$.

Définition 5.1.1 (Paires critiques) *Soit $\mathcal{R}$ un TRS sur $L \subseteq \Lambda(\Sigma)$ qui est algébrique à gauche, et $l \mapsto_{\mathcal{R}} r$,$l' \mapsto_{\mathcal{R}} r'$ deux règles de réécriture de $\mathcal{R}$ qui n'ont pas de variables en commun. Soit p une occurrence dans l qui n'est pas une occurrence de variable et telle que $p \neq \varepsilon$ s'il existe un renommage ρ tel que $(l, r)\rho = (l', r')$, et soit σ l'unificateur le plus général de $l|_p$ et de l'. Alors*

$$(l[r']_p\sigma, r\sigma)$$

est une paire critique *de $l \mapsto_{\mathcal{R}} r$ et $l' \mapsto_{\mathcal{R}} r'$. Une paire critique de la forme (t, t) est dite* triviale.

Notons que $l|_p$, $l'|_p$ et $l[r']_p$ sont bien définis car l et l' sont algébriques. Si $t = l\sigma$ et si on a une paire critique $(l[r']p\theta, r\theta)$ pour deux règles $l \mapsto_{\mathcal{R}} r$ et $l' \mapsto_{\mathcal{R}} r'$ (dans ce cas $\sigma = \sigma' \circ \theta$), alors on a le choix suivant pour la réduction de t :



Ce choix peut être irréversible car rien n'assure que les termes $r(\sigma' \circ \theta)$ et $l[r']p(\sigma' \circ \theta)$ soient joignables.

Les systèmes de réécriture sans paires critiques et linéaires à gauche sont dits orthogonaux. Rappelons qu'un terme t est linéaire si pour tous sous-termes u, v de t à des positions disjointes, on a $FV(u) \cap FV(v) = \emptyset$.

Définition 5.1.2 (Systèmes de réécriture orthogonaux) *Soit un système de réécriture $\mathcal{R}$ sur $L \subseteq \Lambda(\Sigma)$. Alors, si $\mathcal{R}$ est algébrique à gauche, on dit que $\mathcal{R}$ est*

(i) linéaire à gauche *si l est linéaire pour tout $l \mapsto_{\mathcal{R}} r$;*

(ii) non ambigu *(resp.* faiblement non ambigu*) si*
— *pour toutes règles $l \mapsto_{\mathcal{R}} r$ et $l' \mapsto_{\mathcal{R}} r'$ (pas nécessairement distinctes),*
— *pour tout renommage ρ tel que $FV(l\rho, r\rho) \cap FV(l'\rho, r'\rho) = \emptyset$,*
les règles $(l, r)\rho$ et $(l', r')\rho$ ne forment pas de paire critique (resp. toute paire critique entre $(l, r)\rho$ et $(l', r')\rho$ est triviale) ;

(iii) orthogonal *(resp.* faiblement orthogonal*) s'il est linéaire à gauche et non ambigu (resp. faiblement non ambigu).*

La façon la plus simple de montrer la confluence des systèmes de réécriture orthogonaux est d'utiliser une forme de réécriture parallèle. Cette technique est standard.

Lemme 5.1.3 (Mouvements parallèles) *Soit $\mathcal{R}$ un TRS algébrique à gauche sur $\mathsf{L} \subseteq \Lambda(\Sigma)$ et $\to_{\|\mathcal{R}}$ la clôture parallèle de $\mathcal{R}$, au sens de 2.3.5. Si $\mathcal{R}$ est (faiblement) orthogonal, alors $\to_{\|\mathcal{R}}$ vérifie la propriété du diamant sur L.*

La confluence des systèmes de réécriture orthogonaux est un résultat bien connu. À notre connaissance, il apparaît pour la première fois dans [Hue80] (c'est le lemme 3.3 et le corollaire suivant).

Théorème 5.1.4 *Si $\mathcal{R}$ est un TRS algébrique à gauche sur $\mathsf{L} \subseteq \Lambda(\Sigma)$ qui est (faiblement) orthogonal, alors $\to_{\mathcal{R}}$ est confluent sur $\mathsf{L} \subseteq \Lambda(\Sigma)$.*

Exemple 5.1.5 *Le système $\mapsto_{\mathsf{rec}}$ présenté à l'exemple 3.1.17.(ii) est orthogonal : ses règles*

$$\mathsf{rec}\ x\ y\ 0 \quad \mapsto_{\mathsf{rec}} \quad x \qquad \textit{et} \qquad \mathsf{rec}\ x\ y\ \mathsf{S}(z) \quad \mapsto_{\mathsf{rec}} \quad y\ (\mathsf{rec}\ x\ y\ z)\ z$$

ne forment pas de paire critique. Par le théorème 5.1.4, la relation $\to_{\mathsf{rec}}$ est donc confluente sur $\Lambda(\Sigma)$.

Remarque 5.1.6 *Ce qu'on vient de faire pour les système algébriques à gauche s'applique aussi aux systèmes applicatifs à gauche. Cependant, à cause du symbole d'application, ces systèmes peuvent comporter de nombreuses paires critiques.*

Un exemple est le fait que la fonction eat *de la remarque 3.1.14 n'est pas stricte dans le sens de 3.1.14. En effet, le système composé de*

$$x \cdot \mathsf{err} \quad \mapsto \quad \mathsf{err} \qquad \textit{et} \qquad \mathsf{eat} \cdot x \quad \mapsto \quad \mathsf{eat}$$

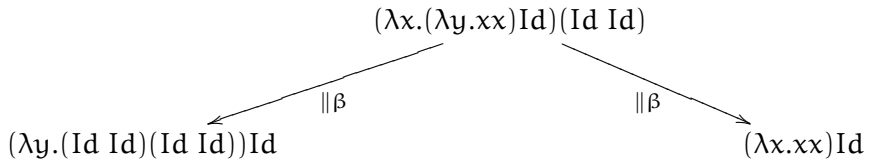
n'est pas confluent :

$$\mathsf{err} \quad \leftarrow \quad \mathsf{eat} \cdot \mathsf{err} \quad \to \quad \mathsf{eat}\ .$$

5.1.2 Lambda-calcul

Dans le cas du λ-calcul, on utilise aussi une relation de réécriture parallèle, mais contrairement aux systèmes algébriques à gauche, cette relation n'est pas la clôture parallèle de $\mapsto_\beta$. En effet, cette relation ne vérifie pas la propriété du diamant car des rédexes parallèles peuvent se retrouver emboîtés après un pas de réduction.

Exemple 5.1.7 *Notons $\to_{\|\beta}$ pour la clôture parallèle de $\mapsto_\beta$ au sens de 2.3.5. On a alors*



On ne peut joindre ce pic avec un pas de $\to_{\|\beta}$ car

$$\begin{aligned}((\lambda y.(\mathrm{Id}\ \mathrm{Id})(\mathrm{Id}\ \mathrm{Id}))\mathrm{Id})_{\|\beta} &= \{(\lambda y.(\mathrm{Id}\ \mathrm{Id})(\mathrm{Id}\ \mathrm{Id}))\mathrm{Id}, (\mathrm{Id}\ \mathrm{Id})(\mathrm{Id}\ \mathrm{Id}),\\ &\quad (\lambda y.\mathrm{Id}\ \mathrm{Id})\mathrm{Id}, (\lambda y.(\mathrm{Id}\ \mathrm{Id})\ \mathrm{Id})\mathrm{Id}, (\lambda y.\mathrm{Id}(\mathrm{Id}\ \mathrm{Id}))\mathrm{Id}\}\\ ((\lambda x.xx)\mathrm{Id})_{\|\beta} &= \{(\lambda x.xx)\mathrm{Id}, \mathrm{Id}\ \mathrm{Id}\}\end{aligned}$$

donc

$$((\lambda y.(\mathrm{Id}\ \mathrm{Id})(\mathrm{Id}\ \mathrm{Id}))\mathrm{Id})_{\|\beta} \quad \cap \quad ((\lambda x.xx)\mathrm{Id})_{\|\beta} \quad = \quad \emptyset\ .$$

La relation de β-réduction parallèle qui vérifie la propriété du diamant est la relation dite de Tait et Martin-Löf.

Définition 5.1.8 (Béta-réduction parallèle) *Soit $\rhd_\beta$ la plus petite relation de réécriture parallèle sur $\Lambda(\Sigma)$ vérifiant la règle suivante :*

$$(\rhd\beta)\ \frac{t_1 \rhd_\beta u_1 \qquad t_2 \rhd_\beta u_2}{(\lambda x.t_1)t_2 \rhd_\beta u_1[u_2/x]}$$

Proposition 5.1.9 *Soit deux termes t, t' et une substitution σ.*

(i) Si $t \rhd_\beta t'$ alors $t\sigma \rhd_\beta t'\sigma$.

(ii) La relation $\rhd_\beta$ est une relation de réécriture parallèle sur $\Lambda(\Sigma)$.

Le point (ii) du lemme suivant est crucial pour la propriété du diamant de $\rhd_\beta$.

Lemme 5.1.10 *Soit un terme t et une substitution σ.*

(i) Si $\sigma \rhd_\beta \sigma'$ alors $t\sigma \rhd_\beta t\sigma'$.

(ii) Si $t \rhd_\beta t'$ et $\sigma \rhd_\beta \sigma'$ alors $t\sigma \rhd_\beta t'\sigma'$.

Nous pouvons maintenant montrer que $\rhd_\beta$ satisfait la propriété du diamant.

Lemme 5.1.11 (Mouvements parallèles — [Bar84, Kri90, Tak95]) *La relation $\rhd_\beta$ vérifie la propriété du diamant.*

On en déduit la confluence de $\rhd_\beta$. Il s'en suit par le lemme 5.1.12 que $\to_\beta$ est confluente (théorème 3.2.4).

Lemme 5.1.12 $\to_\beta \ \subseteq \ \rhd_\beta \ \subseteq \ \to_\beta^*$.

5.2 Confluence de la réécriture conditionnelle

Les stratifications de la réécriture conditionnelle donnent lieu à des raffinements de la notion de confluence, déjà évoqués en 2.1.17.

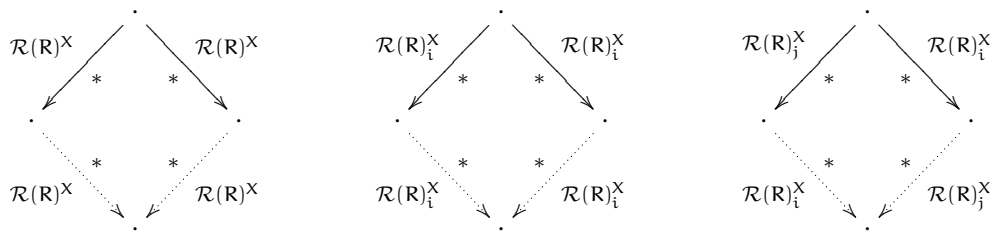
Définition 5.2.1 (Confluences de la réécriture conditionnelle) *Soit $\mathcal{R}$ un système conditionnel et $X \in \{se, jo, o\}$. On dit que $\to_{\mathcal{R}(R)^X}$ est*

— confluent par niveaux *lorsque $\to_{\mathcal{R}(R)^X_i}$ est confluent pour tout $i \in \mathbb{N}$,*

— confluent en profondeur *lorsque pour tout $i, j \in \mathbb{N}$, $\to_{\mathcal{R}(R)^X_i}$ et $\to_{\mathcal{R}(R)^X_j}$ commutent.*

Les notions de confluence, confluence par niveaux et confluence profonde sont représentées figure. 5.1. Il est clair que la confluence profonde implique la confluence par niveaux, qui à son tour implique la confluence. Mais il est important de noter que la confluence n'implique pas nécessairement la confluence par niveaux.

Nous rappelons ici certains résultats connus qui sont utilisés dans la suite, ou bien que nous jugeons intéressants pour la compréhension de la réécriture conditionnelle, en particulier en ce qui concerne la différence entre confluence et confluence par niveaux.



Confluence | Confluence par niveaux | Confluence profonde

FIG. 5.1: Confluences pour $\rightarrow_{\mathcal{R}(R)^X} = \bigcup\{\rightarrow_{\mathcal{R}(R)_i^X} \mid i \in \mathbb{N}\}$

5.2.1 Réécriture conditionnelle orthogonale

Nous présentons ici certains résultats connus sur les confluences de différentes formes de réécriture conditionnelle. Ces résultats concernent les systèmes orthogonaux conditionnels, pour lesquels il nous faut rappeler quelques notions.

La différence avec les systèmes de réécriture de termes est la notion de paire critique conditionnelle, qui est plus riche que dans le cas non conditionnel. Les paires critiques proviennent de la superposition de deux règles de réécritures dont les conditions, selon les cas, peuvent ou ne peuvent pas être satisfaites en même temps. Cela amène à distinguer entre les paires critiques « faisables » et « infaisables ». De plus, la satisfaction des paires critiques dépend de la relation de réécriture conditionnelle considérée.

Définition 5.2.2 (Paires critiques conditionnelles) *Soit* $R : \vec{d} = \vec{c} \supset l \mapsto r$ *et* $R' : \vec{d}' = \vec{c}' \supset l' \mapsto r'$ *deux règles conditionnelles qui n'ont pas de variables en commun. Soit* p *une occurrence dans* l *qui n'est pas une occurrence de variable et telle que* $p \neq \varepsilon$ *s'il existe un renommage* ρ *tel que* $R\rho = R'\rho$ *et soit* σ *l'unificateur le plus général de* $l|_p$ *et de* l'*. Alors*

$$\vec{d}\sigma = \vec{c}\sigma \wedge \vec{d}'\sigma = \vec{c}'\sigma \supset (l[r']_p\sigma, r\sigma)$$

est une paire critique conditionnelle *de* R *et* R'*. Une paire critique est* triviale *si elle est de la forme* $\vec{d} = \vec{c} \supset (t, t)$*.*

Notons que, comme pour le cas non conditionnel 5.1.1, $l|_p$, $l'|_p$ et $l[r']_p$ sont bien définis car l et l' sont algébriques. De plus, la définition 5.2.2 restreinte à des règles non conditionnelles donne la définition 5.1.1.

Supposons que $t = l\sigma$ et que l'on a une paire critique

$$\vec{d}\theta = \vec{c}\theta \wedge \vec{d}'\theta = \vec{c}'\theta \supset (l[r']_p\theta, r\theta)$$

pour deux règles $R : \vec{d} = \vec{c} \supset l \mapsto r$ et $R' : \vec{d}' = \vec{c}' \supset l' \mapsto r'$ (dans ce cas $\sigma = \sigma' \circ \theta$). Alors, si on oublie les conditions, on a le choix suivant pour la réduction de t :

$$r(\sigma' \circ \theta) \quad \leftarrow \quad t \quad \rightarrow \quad (l[r']_p)(\sigma' \circ \theta)$$

L'important avec les règles conditionnelles est que ce choix n'est possible que si les conditions des règles R et R' peuvent être satisfaites par une instance de θ. Bien sûr, il est tout à fait possible que ce ne soit pas le cas.

Cette possibilité que les conditions des paires critiques soient satisfaites ou non s'appelle faisabilité des paire critiques. Il y a autant de notions de faisabilité que de façons de tester les conditions des paires critiques.

Définition 5.2.3 (Faisabilité des paires critiques conditionnelles) *Une paire critique* $\vec{d} = \vec{c} \supset (t, u)$ *est dite*

— se-R-faisable *s'il existe une substitution* σ *telle que* $\vec{d}\sigma \leftrightarrow^*_{\mathcal{R}(\mathsf{R})^{\mathsf{se}} \cup \mathsf{R}} \vec{c}\sigma$.

— jo-R-faisable *s'il existe une substitution* σ *telle que* $\vec{d}\sigma \downarrow_{\mathcal{R}(\mathsf{R})^{\mathsf{jo}} \cup \mathsf{R}} \vec{c}\sigma$.

— o-R-faisable *s'il existe une substitution* σ *telle que* $\vec{d}\sigma \rightarrow^*_{\mathcal{R}(\mathsf{R})^{\mathsf{o}} \cup \mathsf{R}} \vec{c}\sigma$.

Étant donné $\mathsf{X} \in \{\mathsf{se}, \mathsf{jo}, \mathsf{o}\}$*, une paire critique qui n'est pas* X-R-*faisable est dite* X-R-*infaisable.*

Définition 5.2.4 (Réécriture conditionnelle orthogonale) *Étant donnés une relation de réécriture* $\rightarrow_{\mathsf{R}}$ *et* $\mathsf{X} \in \{\mathsf{se}, \mathsf{jo}, \mathsf{o}\}$*, un système conditionnel* $\mathcal{R}$ *linéaire à gauche est*

— X-R-*orthogonal si toutes ses paires critiques sont* X-R-*infaisable,*

— *faiblement* X-R-*orthogonal si toutes ses paires critiques* X-R-*faisables sont triviales.*

Remarque 5.2.5 *Du fait que le pas de réécriture conditionnelle est indécidable (voir remarque 4.1.6), le problème de savoir si un système de réécriture conditionnelle est (faiblement) orthogonal est indécidable.*

On a alors le résultat suivant.

Théorème 5.2.6 ([BK86, Ohl02]) *Soit* $\mathcal{R}$ *un système conditionnel sur* $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$.

(i) Si $\mathcal{R}$ *est faiblement* se*-orthogonal, alors* $\rightarrow_{\mathcal{R}^{\mathsf{se}}}$ *est confluent.*

(ii) Si $\mathcal{R}$ *est normal et faiblement* o*-orthogonal, alors* $\rightarrow_{\mathcal{R}^!}$ *est confluent par niveaux.*

Rappelons que par la proposition 4.1.9, un système normal est o-orthogonal si et seulement s'il est jo-orthogonal.

Remarque 5.2.7 *Dans [Ohl02], il est seulement montré que* $\rightarrow_{\mathcal{R}^!}$ *est confluent par niveaux, mais il semble que la preuve puisse être étendue à la confluence profonde.*

Il est bien connu que les systèmes o- et jo-orthogonaux ne sont pas nécessairement confluents.

Exemple 5.2.8 ([BK86]) *Voici un exemple classique de système orthogonal pour* $\rightarrow_{\mathcal{R}^{\mathsf{se}}}$*,* $\rightarrow_{\mathcal{R}^{\mathsf{jo}}}$ *et* $\rightarrow_{\mathcal{R}^{\mathsf{o}}}$ *mais pour lequel* $\rightarrow_{\mathcal{R}^{\mathsf{jo}}}$ *et* $\rightarrow_{\mathcal{R}^{\mathsf{se}}}$ *ne sont pas confluents :*

$$\begin{array}{ccccccc} & & & & \mathsf{a} & \mapsto & \mathsf{f}(\mathsf{a}) \\ \mathsf{x} & = & \mathsf{f}(\mathsf{x}) & \supset & \mathsf{f}(\mathsf{x}) & \mapsto & \mathsf{b} \end{array}$$

Le pic

$$\mathsf{f(b)} \quad \leftarrow \quad \mathsf{f(f(a))} \quad \rightarrow \quad \mathsf{b}$$

qui se produit pour la réécriture semi-équationnelle, par joignabilité et orientée, n'est pas joignable ni pour $\rightarrow_{\mathcal{R}^{\mathrm{jo}}}$*, ni pour* $\rightarrow_{\mathcal{R}^{\mathrm{o}}}$ *car les termes* $\mathsf{f(b)}$ *et* b *sont normaux pour ces relations. Cependant, avec la réécriture semi-équationnelle, on a* $\mathsf{f(b)} \rightarrow_{\mathcal{R}^{\mathrm{se}}} \mathsf{b}$.

5.2.2 Confluence par niveaux et confluence

On rassemble ici quelques éléments de comparaison entre la confluence et la confluence par niveaux. On montre que tout système semi-équationnellement confluent par niveaux est joignablement confluent.

Lemme 5.2.9 *Si* $\rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}}$ *est confluent par niveaux alors pour tout* $\mathtt{i} \in \mathbb{N}$*, on a* $\rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}}} = \rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}_{\mathtt{i}}}$.

Preuve. On raisonne par induction sur $\mathtt{i} \in \mathbb{N}$. Le cas de base $\mathtt{i} = 0$ étant évident, supposons que $\rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}}} = \rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}_{\mathtt{i}}}$ avec $\mathtt{i} \in \mathbb{N}$. Alors, pour tout règle $\vec{\mathtt{d}} = \vec{\mathtt{c}} \supset \mathtt{l} \rightarrow \mathtt{r} \in \mathcal{R}$, si $\vec{\mathtt{d}}\sigma \downarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}_{\mathtt{i}}} \vec{\mathtt{d}}\sigma$ alors, par hypothèse d'induction on a $\vec{\mathtt{d}}\sigma \downarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}}} \vec{\mathtt{d}}\sigma$, donc $\mathtt{l}\sigma \rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}+1}} \mathtt{r}\sigma$. Inversement, si $\vec{\mathtt{d}}\sigma \leftrightarrow^{*}_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}}} \vec{\mathtt{c}}\sigma$, alors $\vec{\mathtt{d}}\sigma \downarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}}} \vec{\mathtt{c}}\sigma$ car $\rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}}}$ est confluent, donc $\vec{\mathtt{d}}\sigma \downarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}_{\mathtt{i}}} \vec{\mathtt{c}}\sigma$ par hypothèse d'induction, soit $\mathtt{l}\sigma \rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}_{\mathtt{i}+1}} \mathtt{r}\sigma$. Il s'en suit que $\mathtt{l}\sigma \rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}_{\mathtt{i}+1}} \mathtt{r}\sigma$ si et seulement si $\mathtt{l}\sigma \rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}_{\mathtt{i}+1}} \mathtt{r}\sigma$. □

Le lemme 5.2.9 justifie le fait que lorsqu'elle est confluente par niveaux, la réécriture semi-équationnelle puisse être implantée par la réécriture par joignabilité.

Corollaire 5.2.10 *Si* $\rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{se}}}$ *est confluent par niveaux alors* $\rightarrow_{\mathcal{R}(\mathsf{R})^{\mathrm{jo}}}$ *est confluent par niveaux.*

Il s'en suit qu'un système semi-équationnellement confluent peut ne pas être joignablement confluent seulement si il n'est pas semi-équationnellement confluent par niveaux.

Exemple 5.2.11 *Il existe aussi des relations* terminantes *qui sont confluentes mais pas par niveaux. Avec le système*

$$\begin{array}{lllllll} \mathsf{x} & = & \mathsf{h(a)} & \supset & \mathsf{f(x)} & \mapsto & \mathsf{a} \\ \mathsf{x} & = & \mathsf{a} & \supset & \mathsf{h(x)} & \mapsto & \mathsf{x}\,, \end{array}$$

on a

$$\begin{array}{rlllll} & \mathsf{f(h(a))} & \rightarrow_{\mathcal{R}^{\mathrm{se}}_1} \mathsf{a} & \textit{car} & \mathsf{h(a)} & \leftrightarrow^{*}_{\mathcal{R}^{\mathrm{se}}_0} \mathsf{h(a)} \\ \textit{et} & \mathsf{h(a)} & \rightarrow_{\mathcal{R}^{\mathrm{se}}_1} \mathsf{a} & \textit{car} & \mathsf{a} & \leftrightarrow^{*}_{\mathcal{R}^{\mathrm{se}}_0} \mathsf{a}\,. \end{array}$$

Il s'en suit qu'on peut former le pic

$$\mathsf{a} \quad \leftarrow_{\mathcal{R}^{\mathrm{se}}_1} \quad \mathsf{f(h(a))} \quad \rightarrow_{\mathcal{R}^{\mathrm{se}}_1} \quad \mathsf{f(a)}\,,$$

qui ne peut être joint que par la réduction $\mathsf{f(a)} \rightarrow_{\mathcal{R}^{\mathrm{se}}_2} \mathsf{a}$ *qui est déclenchée par* $\mathsf{h(a)} \leftrightarrow^{*}_{\mathcal{R}^{\mathrm{se}}_1} \mathsf{a}$. *La relation* $\rightarrow_{\mathcal{R}^{\mathrm{se}}}$ *est donc confluente sans être confluente par niveaux.*

5.3 Modularité et extension de la signature

Cette section est dévolue à quelques remarques sur la modularité de la confluence. Nous rappelons les résultats connus dans le cas de la réécriture du premier ordre, et les étendons au cas des termes modulo α-conversion.

Ensuite, nous utilisons notre travail de la section 4.3 sur la réécriture avec termes de de Bruijn pour obtenir une caractérisation de la confluence en fonction de la confluence close.

Enfin, nous rappelons un exemple de non préservation de la confluence par combinaison non disjointe. Cet exemple est adapté à la section 6.1 au cas de la combinaison de la réécriture au λ-calcul. Nous en tentons une analyse à l'aide d'arbres de Böhm pour la réécriture [KOV99].

5.3.1 Systèmes du premier ordre

Nous rappelons ici les résultats de modularité pour les combinaisons disjointes de systèmes du premier ordre.

La modularité de la confluence pour les combinaisons disjointes de TRS premier ordre est due à Toyama [Toy87]. La preuve a été simplifiée dans [KMTV94] puis dans [Jou06].

Théorème 5.3.1 ([Toy87]) *Si* $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\mathcal{R}_1)$ *et* $(\mathcal{T}er(\Sigma_2,\mathcal{X}),\mathcal{R}_2)$ *sont deux TRS confluents tels que* $\Sigma_1\cap\Sigma_2=\emptyset$*, alors le TRS* $(\mathcal{T}er(\Sigma_1\cup\Sigma_2,\mathcal{X}),\mathcal{R}_1\cup\mathcal{R}_2)$ *est aussi confluent.*

Une relation $\rightarrow_{\mathcal{R}(R)^X}$ de réécriture conditionnelle sur $\mathcal{T}er(\Sigma,\mathcal{X})$ est un TRS. On pourrait alors être tenté d'utiliser le théorème 5.3.1 pour obtenir la modularité de la confluence pour la réécriture conditionnelle. Soient en effet deux relations de réécriture conditionnelle $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\rightarrow_{\mathcal{R}_1^X})$ et $(\mathcal{T}er(\Sigma_2,\mathcal{X}),\rightarrow_{\mathcal{R}_2^X})$ telles que $\Sigma_1\cap\Sigma_2=\emptyset$. D'après le théorème 5.3.1, le TRS $(\mathcal{T}er(\Sigma_1\cup\Sigma_2,\mathcal{X}),\rightarrow_{\mathcal{R}_1^X}\cup\rightarrow_{\mathcal{R}_2^X})$ est confluent. Mais le problème est que nous voulons la confluence de la relation $\rightarrow_{(\mathcal{R}_1\cup\mathcal{R}_2)^X}$ et il n'est pas clair que ces deux relations coïncident.

Heureusement, il existe un résultat de modularité de la confluence pour la réécriture conditionnelle, dû à Middeldorp [Mid91]. Malheureusement, le cas de la réécriture orientée n'a pas été traité.

Théorème 5.3.2 ([Mid91]) *Soient* $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\mathcal{R}_1)$ *et* $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\mathcal{R}_2)$ *deux systèmes de réécriture conditionnels tels que* $\Sigma_1\cap\Sigma_2=\emptyset$*. Pour tout* $X\in\{se,jo\}$*,*

si $\rightarrow_{\mathcal{R}_1{}^X_{\mathcal{T}er(\Sigma_1,\mathcal{X})}}$ *est confluent sur* $\mathcal{T}er(\Sigma_1,\mathcal{X})$

et si $\rightarrow_{\mathcal{R}_2{}^X_{\mathcal{T}er(\Sigma_2,\mathcal{X})}}$ *est confluent sur* $\mathcal{T}er(\Sigma_2,\mathcal{X})$*,*

alors $\rightarrow_{(\mathcal{R}_1\cup\mathcal{R}_2)^X_{\mathcal{T}er(\Sigma_1\cup\Sigma_2,\mathcal{X})}}$ *est confluent sur* $\mathcal{T}er(\Sigma_1\cup\Sigma_2,\mathcal{X})$*.*

Un corollaire simple mais important est la préservation de la confluence par extension de la signature.

Corollaire 5.3.3 (Extension de la signature) *Soit* $(\mathcal{T}er(\Sigma,\mathcal{X}),\mathcal{R})$ *un système de réécriture conditionnel,* $\Sigma' \supseteq \Sigma$ *et* $X \in \{se, jo\}$. *Si* $\to_{\mathcal{R}^X_{\mathcal{T}er(\Sigma,\mathcal{X})}}$ *est confluent sur* $\mathcal{T}er(\Sigma,\mathcal{X})$, *alors* $\to_{\mathcal{R}^X_{\mathcal{T}er(\Sigma',\mathcal{X})}}$ *est confluent sur* $\mathcal{T}er(\Sigma',\mathcal{X})$.

5.3.2 Lambda termes

Voyons maintenant comment étendre ces résultats à la réécriture conditionnelle sur les λ-termes, c'est-à-dire comment prendre en compte l'α-conversion.

Soit $(\_)^{App}$ l'inverse à gauche de l'opération $(\_)_{App}$ définie en 1.4.1 : $\Sigma^{App} = \Sigma\backslash\{\_\cdot\_\}$, de telle sorte que $(\Sigma_{App})^{App} = \Sigma$. Cette opération sert à simplifier les énoncés : quand on passe de $\mathcal{T}er(\Sigma,\mathcal{X})$ à $\mathcal{L}(\Sigma^{App})$ ou à $\Lambda(\Sigma^{App})$, on ne se préoccupe pas de savoir si Σ contient ou pas un symbole d'application.

En combinant le corollaire 5.3.3 au corollaire 4.3.9, on obtient la préservation de la confluence lors du passage des termes du premier ordre aux λ-termes. Étant donnée une signature Σ, on étend les notations de la section 4.3 en posant $\to_{\mathcal{R}^X_{\mathcal{T}er}} =_{def} \to_{\mathcal{R}^X_{\mathcal{T}er(\Sigma,\mathcal{X})}}$. Rappelons que pour $t \in \mathcal{L}(\Sigma)$, $[t]$ désigne la classe d'α-équivalence de t.

Théorème 5.3.4 (Confluence sur les lambda-termes) *Soit* $X \in \{jo, se\}$ *et* $\mathcal{R}$ *un système conditionnel sur* $\mathcal{T}er(\Sigma,\mathcal{X})$. *Si* $\to_{\mathcal{R}^X_{\mathcal{T}er}}$ *est confluent sur* $\mathcal{T}er(\Sigma,\mathcal{X})$, *alors* $\to_{\mathcal{R}^X_\Lambda}$ *est confluent sur* $\Lambda(\Sigma^{App},\mathcal{X})$.

Preuve. Si $\to_{\mathcal{R}^X_{\mathcal{T}er}}$ est confluent sur $\mathcal{T}er(\Sigma,\mathcal{X})$, alors, par le corollaire 5.3.3, $\to_{\mathcal{R}^X_{dB}}$ est confluent sur $\mathcal{T}dB(\Sigma^{App})$.

On en déduit, en utilisant le théorème 4.3.8, que $\to_{\mathcal{R}^X_\Lambda}$ est confluent sur $\Lambda(\Sigma^{App})$: Soient $[t],[u],[v] \in \Lambda(\Sigma)$ et R un référentiel pour t, u et v. Si $[u] \leftarrow^*_{\mathcal{R}^X_\Lambda} [t] \to^*_{\mathcal{R}^X_\Lambda} [v]$, alors $u_R \leftarrow^*_{\mathcal{R}^X_{dB}} t_R \to^*_{\mathcal{R}^X_{dB}} v_R$. Comme $\to_{\mathcal{R}^X_{dB}}$ est confluent sur $\mathcal{T}dB(\Sigma^{App})$, il existe $w \in \mathcal{T}dB(\Sigma^{App})$ tel que $u_R \to^*_{\mathcal{R}^X_{dB}} w \leftarrow^*_{\mathcal{R}^X_{dB}} v_R$. Comme les indices libres de w sont libres dans u_R et dans v_R, R est aussi un référentiel pour w. Par le théorème 1.6.9.(i), on a donc $(w^R)_R = w$, d'où $[u] \to^*_{\mathcal{R}^X_\Lambda} [w^R] \leftarrow^*_{\mathcal{R}^X_\Lambda} [v]$. □

Le théorème 5.3.4 permet d'étendre le théorème 5.3.2 à la modularité de la confluence pour la réécriture conditionnelle du premier ordre modulo α-conversion. L'opération $(\_)^{App}$ permet qu'un des systèmes (mais pas les deux) utilise un symbole d'application.

Théorème 5.3.5 (Modularité modulo alpha-conversion) *Soient* $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\mathcal{R}_1)$ *et* $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\mathcal{R}_2)$ *deux systèmes de réécriture conditionnels tels que* $\Sigma_1 \cap \Sigma_2 = \emptyset$. *Pour tout* $X \in \{se, jo\}$,

si $\to_{\mathcal{R}_1{}^X_{\Lambda(\Sigma_1^{App})}}$ *est confluent sur* $\Lambda(\Sigma_1^{App})$ *et* $\to_{\mathcal{R}_2{}^X_{\Lambda(\Sigma_2^{App})}}$ *est confluent sur* $\Lambda(\Sigma_2^{App})$,

alors $\to_{(\mathcal{R}_1\mathcal{R}_2)^X_{\Lambda(\Sigma_1\cup\Sigma_2)^{App}}}$ *est confluent sur* $\Lambda((\Sigma_1 \cup \Sigma_2)^{App})$.

Preuve. Si les relations $\to_{\mathcal{R}_1{}^X_{\Lambda(\Sigma_1^{App})}}$ et $\to_{\mathcal{R}_2{}^X_{\Lambda(\Sigma_2^{App})}}$ sont confluentes alors $\to_{\mathcal{R}_1{}^X_{\mathcal{T}er(\Sigma_1,\mathcal{X})}}$ et $\to_{\mathcal{R}_2{}^X_{\mathcal{T}er(\Sigma_2,\mathcal{X})}}$ le sont aussi. Il s'en suit par le théorème 5.3.2 que $\to_{(\mathcal{R}_1\mathcal{R}_2)^X_{\mathcal{T}er(\Sigma_1\cup\Sigma_2,\mathcal{X})}}$

est confluent donc que $\to_{(\mathcal{R}_1\mathcal{R}_2)^X_{\Lambda((\Sigma_1\cup\Sigma_2)^{App})}}$ est confluent sur $\Lambda((\Sigma_1 \cup \Sigma_2)^{App})$ par le théorème 5.3.4. □

Remarque 5.3.6 *Il n'est pas clair que le théorème 5.3.5 puisse être obtenu en utilisant des résultats sur la modularité de la confluence pour la réécriture modulo.*

Par exemple, les résultat de [Jou06] ne s'appliquent pas car l'α-conversion est partagée par les deux systèmes de réécriture.

On en déduit l'extension du corollaire 5.3.3 aux λ-termes.

Corollaire 5.3.7 (Extension de la signature modulo α-conversion) *Soit $\mathcal{R}$ un système conditionnel sur $\mathcal{T}er(\Sigma, \mathcal{X})$, $\Sigma' \supseteq \Sigma$ et $X \in \{se, \mathfrak{so}\}$. Si $\to_{\mathcal{R}^X_{\Lambda(\Sigma^{App})}}$ est confluent sur $\Lambda(\Sigma^{App})$, alors $\to_{\mathcal{R}^X_{\Lambda(\Sigma'^{App})}}$ est confluent sur $\Lambda(\Sigma'^{App})$.*

5.3.3 Une caractérisation de la confluence

Les résultats de modularité présentés ci-dessus, ainsi que les liens entre la réécriture sur les λ-termes et la réécriture sur les termes de de Bruijn permettent de caractériser la confluence en fonction de la confluence close.

Théorème 5.3.8 *Soit $\mathcal{R}$ un système applicatif sur $\Lambda(\Sigma)$ et $X \in \{se, \mathfrak{so}\}$. Alors $\to_{\mathcal{R}^X_\Lambda}$ est confluent sur $\Lambda(\Sigma)$ si et seulement si $\to_{\mathcal{R}^X_{\mathcal{T}er(\Sigma')}}$ est confluent sur $\mathcal{T}er(\Sigma')$ pour tout $\Sigma' \supseteq \Sigma_{App}$.*

Preuve. Si $\to_{\mathcal{R}^X_\Lambda}$ est confluente sur $\Lambda(\Sigma)$, alors elle est confluente sur $\mathcal{T}er(\Sigma_{App}, \mathcal{X})$, donc $\to'_{\mathcal{R}^X_{\mathcal{T}er(\Sigma')}}$ est confluente sur $\mathcal{T}er(\Sigma')$ pour tout $\Sigma' \supseteq \Sigma_{App}$ par le corollaire 5.3.3.

Inversement, si $\to_{\mathcal{R}^X_{\mathcal{T}er(\Sigma')}}$ est confluent sur $\mathcal{T}er(\Sigma')$ pour tout $\Sigma' \supseteq \Sigma_{App}$, alors $\to_{\mathcal{R}^X_{dB}}$ est confluent sur $\mathcal{T}dB(\Sigma)$. On en déduit la confluence de $\to_{\mathcal{R}^X_\Lambda}$ sur $\Lambda(\Sigma)$ en utilisant le théorème 4.3.8 : Soient $[t], [u], [v] \in \Lambda(\Sigma)$ et R un référentiel pour t, u et v. Si $[u] \leftarrow^*_{\mathcal{R}^X_\Lambda} [t] \to^*_{\mathcal{R}^X_\Lambda} [v]$, alors $u_R \leftarrow^*_{\mathcal{R}^X_{dB}} t_R \to^*_{\mathcal{R}^X_{dB}} v_R$. Comme $\to_{\mathcal{R}^X_{dB}}$ est confluent sur $\mathcal{T}dB(\Sigma)$, il existe $w \in \mathcal{T}dB(\Sigma)$ tel que $u_R \to^*_{\mathcal{R}^X_{dB}} w \leftarrow^*_{\mathcal{R}^X_{dB}} v_R$. Comme les indices libres de w sont libres dans u_R et dans v_R, R est aussi un référentiel pour w. Par le théorème 1.6.9.(i), on a donc $(w^R)_R = w$, d'où $[u] \to^*_{\mathcal{R}^X_\Lambda} [w^R] \leftarrow^*_{\mathcal{R}^X_\Lambda} [v]$. □

5.3.4 Combinaisons non disjointes et termes infinis

Il est bien connu que la confluence n'est en général pas préservée lorsque les systèmes peuvent partager des symboles. Dans cette section, nous tentons une explication intuitive d'un exemple important pour comprendre les problèmes liés à la combinaison du λ-calcul non typé et de la réécriture.

Le lemme de Hindley-Rosen dit que la confluence de la combinaison de relations de réécriture confluentes suit de la commutation des deux relations. Dans le cas de deux TRS algébriques à gauche $\mathcal{R}_1$ et $\mathcal{R}_2$, les relations $\to_{\mathcal{R}_1}$ et $\to_{\mathcal{R}_2}$ commutent lorsque $\mathcal{R}_1$

et $\mathcal{R}_2$ sont linéaires à gauche et qu'il n'y a pas de paire critique entre une règle de $\mathcal{R}_1$ et une règle de $\mathcal{R}_2$.

Théorème 5.3.9 ([Oos94] — Théorème 5.10.5 dans [Ter03]) *Soient deux TRS algébriques à gauche confluents* $(\mathcal{T}er(\Sigma_1, \mathcal{X}), \mathcal{R}_1)$ *et* $(\mathcal{T}er(\Sigma_2, \mathcal{X}), \mathcal{R}_2)$. *Si* $\mathcal{R}_1$ *et* $\mathcal{R}_2$ *sont linéaires à gauche et s'il n'y pas de paire critique entre une règle de* $\mathcal{R}_1$ *et une règle de* $\mathcal{R}_2$ *alors* $(\mathcal{T}er(\Sigma_1 \cup \Sigma_2, \mathcal{X}), \mathcal{R}_1\mathcal{R}_2)$ *est confluent.*

L'hypothèse de linéarité à gauche ne peut être éliminée.

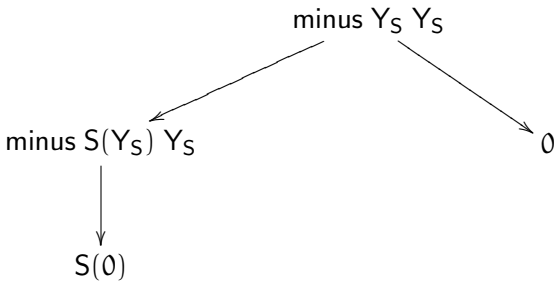
Exemple 5.3.10 (Exemple 8.2.1 dans [Ohl02]) *Les deux règles non-linéaires suivantes, issues du système* $\mapsto_{\mathsf{minus}}$ *présenté en 3.1.11, induisent une relation confluente sur* $\mathcal{T}er(\Sigma_{\mathsf{minus}}, \mathcal{X})$ *:*

$$\mathsf{minus}\ x\ x \quad \mapsto \quad 0 \qquad\qquad \mathsf{minus}\ \mathsf{S}(x)\ x \quad \mapsto \quad \mathsf{S}(0)\ .$$

Par contre, si on ajoute un symbole $\mathsf{Y_S}$ *défini par la règle*

$$\mathsf{Y_S} \quad \mapsto_{\mathsf{Y_S}} \quad \mathsf{S}(\mathsf{Y_S})\ ,$$

alors on a



Ce système est un exemple de *combinaison avec constructeurs partagés* : les symboles partagés entre les deux systèmes ne sont que des constructeurs. Les combinaisons avec constructeurs partagés sont un cas particulier de *combinaisons hiérarchiques*, dans lesquelles les symboles partagés peuvent être définis dans un des deux systèmes dès lors que toutes les définitions de symboles partagés se font dans le même système. Voir la section 8 de [Ohl02] pour plus de détails à ce sujet.

Une manière simple de comprendre l'exemple 5.3.10[1], est de voir $\mathsf{Y_S}$ comme représentant l'« entier infini » ∞, le terme $\mathsf{minus}\ \mathsf{Y_S}\ \mathsf{Y_S}$ représentant l'opération $\infty - \infty$ qui n'est pas définie. Essayons de voir comment $\mathsf{Y_S}$ représente ∞. Nous tentons une explication intuitive utilisant la notion d'arbre de Böhm pour la réécriture développée en [KOV99]. On se base sur la présentation de [Ter03].

Les arbres de Böhm de la définition 12.9.10 de [Ter03] sont paramétrés par une notion de termes « indéfinis », qui ont tous pour arbre de Böhm l'arbre minimal $\perp$. Dans notre cas, l'identification des termes n'ayant pas de forme normale de tête convient.

[1] Nous la devons à Gilles Dowek.

Définition 5.3.11 *Une* forme normale de tête *pour un TRS $\mathcal{R}$ algébrique à gauche est un terme t tel qu'il n'existe pas de règle $\mathsf{l} \mapsto_{\mathcal{R}} \mathsf{r}$ ni de substitution σ telles que $\mathsf{t} = \mathsf{l}\sigma$.*

On applique la définition 12.9.10 de [Ter03] au système $\mapsto_{\mathsf{Y_S}}$, en utilisant comme ensemble de termes indéfinis l'ensemble des termes n'ayant pas de forme normale de tête pour $\mapsto_{\mathsf{Y_S}}$. Cet ensemble est vide car tous les termes ont une forme $\mapsto_{\mathsf{Y_S}}$-normale de tête. De ce fait, il suit de la définition 12.9.10 de [Ter03] que l'arbre de Böhm de $\mathsf{Y_S}$ est sa forme normale par réduction transfinie fortement continue, qui est unique par le corollaire 12.8.15 de [Ter03]. À l'aide du théorème 12.2.1 de [Ter03], on peut représenter cette forme normale par l'arbre infini suivant :

Notre intuition est que $\mathsf{Y_S}$ représente l'entier infini ∞, non pas seulement en étendant la signature Σ_{minus} par une constante ∞ (ce qui aurait préservé la confluence par le corollaire 5.3.3), mais, grâce à la règle $\mapsto_{\mathsf{Y_S}}$, en étendant « sémantiquement » l'algèbre de termes $\mathcal{T}er(\Sigma_{\mathsf{minus}}, \mathcal{X})$ avec des termes infinis.

C'est une manifestation du fait que bien que la confluence soit préservée par extension de la signature (corollaire 5.3.7), elle n'est en général pas préservée par extension de l'algèbre (par exemple par des éléments infinis).

Chapitre 6

Réécriture combinée au lambda-calcul

Nous présentons ici quelques résultats d'études systématiques sur la combinaison du λ-calcul et de la réécriture. Nous nous concentrons sur des propriétés liées à la confluence.

Après avoir vu comment l'exemple 5.3.10 s'instancie avec les points fixes définissables dans le λ-calcul, nous passons en revue les travaux de [Mül92, BTM87, BT88, BTG89, Oka89, Dou92] sur la confluence de la combinaison du λ-calcul et de la réécriture.

Ensuite, nous nous penchons brièvement sur l'extensionnalité. Nous commençons par un exemple dans lequel l'extensionnalité fait perdre la confluence d'un système de réécriture, puis nous rappelons quelques travaux [DCK94, DCK96, Xi96]. De plus, nous donnons une nouvelle preuve du fait que sur les termes algébriques, la conversion extensionnelle est une extension conservative de la conversion de la réécriture algébriquement typée dans le système F, résultat dû à [BTG89, BTG94] pour le système F *à la Church*. Notre preuve a l'avantage de ne pas utiliser de termes en forme normale βη-longue et de fonctionner avec le système F *à la Curry*.

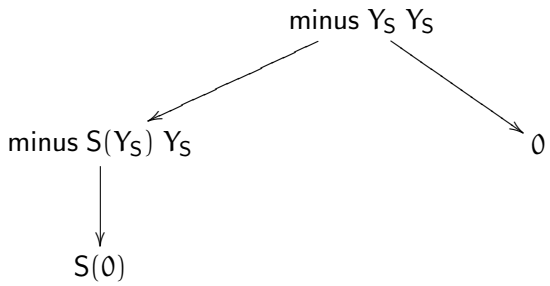
En particulier cela montre, que bien que possiblement non confluente, la combinaison de la réécriture confluente et algébriquement typé au système F extensionnel engendre une conversion consistante.

6.1 Confluence de la réécriture et du lambda-calcul

Comme montré par J.W. Klop dans sa thèse [Klo80], l'ajout d'un système de réécriture confluent mais non-linéaire à gauche au λ-calcul non typé peut résulter en un système non confluent. Un exemple très simple est présenté dans [BTM87]. Il peut être vu comme une variation sur l'exemple 5.3.10, dans laquelle le système $\mapsto_{\mathsf{Y_S}}$ est implanté par le combinateur de point fixe $\mathsf{Y_S}$ du λ-calcul, présenté en 3.2.10.

Exemple 6.1.1 (Suite de l'exemple 5.3.10 — [BTM87]) *La relation* $\rightarrow_{\beta\mathsf{minus}}$ *n'est*

pas confluente sur $\Lambda(\Sigma)$ *:*



Nous renvoyons à la section 5.3.4 pour une tentative d'interprétation de cet exemple à l'aide des arbres de Böhm de [KOV99]. Nous considérons deux moyens de s'en sortir :
— restreindre les règles de réécriture,
— restreindre la β-réduction.

6.1.1 Combinaisons non typées

Si on se restreint aux règles de réécriture dont les membres gauches sont linéaires, on prive la réécriture de moyen de tester l'égalité des éléments arbitraires de l'algèbre. C'est suffisant pour obtenir la préservation de la confluence.

La linéarité à gauche permet d'obtenir la propriété suivante, simple mais fondamentale. Rappelons que $\sigma \rhd_\beta \sigma'$ si et seulement si $\mathcal{D}om(\sigma) = \mathcal{D}om(\sigma')$ et $\sigma(x) \rhd_\beta \sigma'(x)$ pour tout $x \in \mathcal{D}om(\sigma)$ (c'est une instance de la définition 1.3.12). Rappelons aussi que $\rhd_\beta$ est une relation réflexive.

Proposition 6.1.2 *Soit* t *un terme algébrique et linéaire et* σ *une substitution telle que* $t\sigma \rhd_\beta u$*. Alors* $u = t\sigma'$ *avec* $\sigma \rhd_\beta \sigma'$ *et* $\sigma(x) = \sigma'(x)$ *pour tout* $x \notin FV(t)$*.*

Preuve. Par induction sur t.

$t = x \in \mathcal{X}$. Dans ce cas, $t\sigma = \sigma(x)$. Donc si σ' est telle que $\sigma'(x) = u$ et $\sigma'(y) = \sigma(y)$ pour tout $y \neq x$, alors on a bien $u = t\sigma'$ avec $\sigma \rhd_\beta \sigma'$ et $\sigma(y) = \sigma'(y)$ pour tout $y \notin FV(t)$.

$t = t_1 t_2$. Comme t est algébrique, $t_1\sigma\, t_2\sigma$ n'est pas un β-rédex, donc $u = u_1 u_2$ avec $t_1\sigma \rhd_\beta u_1$ et $t_2\sigma \rhd_\beta u_2$.
Par hypothèse d'induction, on a donc deux substitutions σ'_1 et σ'_2 telles que pour tout $i \in \{1,2\}$ on ait $\sigma \rhd_\beta \sigma'_i$, $u_i = t_i\sigma'_i$, et $\sigma'_i(x) = \sigma(x)$ pour tout $x \notin FV(t_i)$.
Comme t est linéaire, l'intersection de $FV(t_1)$ et de $FV(t_2)$ est vide, donc avec $\sigma' =_{\text{def}} \sigma'_1 \uplus \sigma'_2$, on a $u = u_1 u_2 = t_1\sigma'\, t_2\sigma' = t\sigma'$, $\sigma \rhd_\beta \sigma'$ et $\sigma(y) = \sigma'(y)$ pour tout $y \notin FV(t)$.

$t = f(t_1, \ldots, t_n)$. Similaire. □

Lemme 6.1.3 ([Mül92]) *Soit* $\mathcal{R}$ *un système de réécriture linéaire à gauche et algébrique à gauche. Pour tout* $t, u, v \in \Lambda(\Sigma)$*, si* $u \lhd_\beta t \to_{\mathcal{R}} v$ *alors il existe un terme* w *tel que* $u \to^*_{\mathcal{R}} w \lhd_\beta v$*.*

PREUVE. On raisonne par induction sur t. Si t les deux réductions $t \rhd_\beta u$ et $t \to_\mathcal{R} v$ sont dans des sous-termes stricts de t, alors on conclut par hypothèse d'induction. Sinon, par le lemme 3.2.6 il y a deux possibilités :

(i) t est un $\mathcal{R}$-rédex $l\sigma$, avec $v = r\sigma$ et $l \mapsto_\mathcal{R} r$. Alors, comme l est linéaire et algébrique, par la proposition 6.1.2, il existe σ' telle que $u = l\sigma'$ et $\sigma \rhd_\beta \sigma'$, et il s'en suit que $r\sigma \rhd_\beta r\sigma'$ par le lemme 2.3.6. On a $u \to_\mathcal{R} r\sigma' \lhd_\beta r\sigma$, donc $w =_{\text{def}} r\sigma'$ convient.

(ii) t est un β-rédex $(\lambda x.t_1)t_2$, avec $v = (\lambda x.v_1)v_2$ et $u = u_1[u_2/x]$, où $(t_1, t_2) \to_\mathcal{R} (v_1, v_2)$ et $t_i \rhd_\beta u_i$ pour tout $i \in \{1, 2\}$.
Par hypothèse d'induction, il existe w_1 et w_2 tels que $u_i \to_\mathcal{R}^* w_i \lhd_\beta v_i$ pour tout $i \in \{1, 2\}$. On en conclut que $u_1[u_2/x] \to_\mathcal{R}^* w_1[w_2/x]$ par le lemme 2.3.1, et que $(\lambda x.v_1)v_2 \rhd_\beta w_1[w_2/x]$ en utilisant la règle $(\rhd\beta)$; donc $w =_{\text{def}} w_1[w_2/x]$ convient. □

On en déduit facilement que la confluence est préservée.

Théorème 6.1.4 (Confluence de $\to_{\beta\mathcal{R}}$ — [Mül92]) *Soit un TRS $\mathcal{R}$ algébrique à gauche et linéaire à gauche sur une signature Σ. Si $\to_\mathcal{R}$ est confluent sur $\Lambda(\Sigma)$, alors $\to_{\beta\mathcal{R}}$ est confluent sur $\Lambda(\Sigma)$.*

PREUVE. Du lemme 6.1.3 suit la commutation de $\to_\mathcal{R}$ et $\rhd_\beta$ par le lemme 2.1.15. On obtient ainsi la commutation de $\to_\mathcal{R}$ et $\to_\beta$ grâce au lemme 5.1.12, et on en déduit la confluence de $\to_{\mathcal{R}\beta}$ par lemme de Hindley-Rosen (théorème 2.1.14). □

Remarque 6.1.5 *En combinant le théorème 6.1.4 avec le théorème 5.1.4, on obtient que si $\mathcal{R}$ est un TRS sur $\Lambda(\Sigma)$ algébrique à gauche et orthogonal, alors $\to_{\beta\mathcal{R}}$ est confluent sur $\Lambda(\Sigma)$.*

Nous avons vu à l'exemple 5.1.5 que le système $\mapsto_{\mathsf{rec}}$ (présenté à l'exemple 3.1.17.(ii)) est orthogonal et algébrique à gauche. Il s'en suit que $\to_{\beta\mathsf{rec}}$ est confluente sur $\Lambda(\Sigma)$.

Remarque 6.1.6

— *En réalité, dans [Mül92], Müller ne prouve pas exactement la propriété exprimée dans le lemme 6.1.3. Il n'utilise pas la relation $\rhd_\beta$ de Tait et Martin-Löf, mais la clôture parallèle de $\to_\beta$, que nous avons notée $\to_{\|\beta}$ à l'exemple 5.1.7. Contrairement à $\rhd_\beta$, cette relation ne satisfait pas la propriété du diamant.*
Nous présentons le lemme 6.1.3 avec $\rhd_\beta$ car nous utilisons la propriété du diamant de $\rhd_\beta$ en 7.2.1 pour l'extension du théorème 6.1.4 à la réécriture β-conditionnelle (théorème 7.2.9).

— *Le théorème 6.1.4 est subsumé par les résultats de [OR94]. Ces résultats concernent une forme très générale de réécriture, et leur adaptation au raisonnement « stratifié » de la réécriture conditionnelle n'est pas claire.*

6.1.2 Combinaisons typées

L'autre façon de faire est de limiter le λ-calcul. Un des moyens les plus naturels est de se restreindre à une discipline de typage.

Pour les systèmes présentés jusqu'à maintenant, la combinaison se passe plutôt bien. En fait, l'interaction entre les deux systèmes est plutôt faible, comme le suggère le résultat suivant.

Rappelons que d'après la remarque 3.6.6, $\mathcal{T}er(\Sigma_\tau, \mathcal{X})$ est isomorphe en tant que Σ-algèbre libre sur $\mathcal{X}$ aux termes algébriquement typés de $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0, \tau)$, et que d'après la définition 3.6.10, $\mathcal{T}er(\Sigma_\tau, \mathcal{X}) \vdash \mathtt{t} \equiv \mathtt{u}$ signifie que $\mathtt{t} \equiv \mathtt{u}$ sur les termes algébriquement typés de $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0, \tau)$.

Théorème 6.1.7 (Extenstion conservative — [BTM87]) *Soit $\mathcal{R}$ un TRS algébrique algébriquement typé dans $\boldsymbol{\lambda}_2(\mathcal{B}_0, \Sigma_0)$. Pour tout $\mathtt{t}, \mathtt{u}$ algébriquement typés on a*

$$\boldsymbol{\lambda}_2(\mathcal{B}_0, \Sigma_0) \vdash \mathtt{t} =_{\beta\mathcal{R}} \mathtt{u} \qquad \textit{si et seulement si} \qquad \mathcal{T}er(\Sigma_\tau, \mathcal{X}) \vdash \mathtt{t} =_{\mathcal{R}} \mathtt{u}\ .$$

Avec des systèmes algébriquement typés, la réécriture peut être projetée sur les formes β-normales bien typées. On peut alors se passer de la condition de linéarité gauche. Les premiers résultats dans cette direction ont été obtenus pour le λ-calcul simplement typé par [BT88]. Ils ont été étendus au système F explicite dans [BTG89].

Lemme 6.1.8 ([BT88, BTG89]) *Soit $\mathsf{ty} \in \{\Rightarrow, 2, 2\Lambda\}$, $\mathcal{R}$ un TRS algébriquement typé dans $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0)$ et $\mathtt{t}$ un terme bien typé de $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0)$. Si $\mathtt{t} \to_{\mathcal{R}} \mathtt{u}$ alors $\beta\mathtt{nf}(\mathtt{t}) \to^*_{\mathcal{R}} \beta\mathtt{nf}(\mathtt{u})$.*

On en déduit facilement que $\mathtt{t} \to^*_{\beta\mathcal{R}} \mathtt{u}$ implique $\beta\mathtt{nf}(\mathtt{t}) \to^*_{\mathcal{R}} \beta\mathtt{nf}(\mathtt{u})$.

Théorème 6.1.9 ([BT88, BTG89]) *Soit $\mathsf{ty} \in \{\Rightarrow, 2, 2\Lambda\}$ et $\mathcal{R}$ un TRS algébriquement typé dans $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0)$. Si $\to_{\mathcal{R}}$ est confluent sur $\Lambda(\Sigma)$, alors $\to_{\mathcal{R}\beta}$ est confluent sur les terme typés de $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}_0, \Sigma_0)$.*

Remarque 6.1.10 (Curryfication) *Notons que les résultats 6.1.7, 6.1.8 et 6.1.9 s'appliquent à des systèmes* curryfiés*, voir l'exemple 3.1.13 et le chapitre 8.*

Comme les termes curryfiés algébriquement typés sur (Σ_0, τ) sont des Σ_τ-algèbres libres (voir remarque 3.6.6), les travaux [BT88, BTG89] contiennent implicitement la préservation de la confluence par curryfication typée.

Quelques mots sur la normalisation forte. Les résultats présentés jusqu'ici ne permettent pas de prendre en compte les systèmes qui sont à la fois non-linéaires à gauche et non-algébriques à droite. Mais lorsqu'on sait prouver la normalisation forte de $\to_{\beta\mathcal{R}}$, on peut utiliser le lemme de Newman (lemme 2.1.9), et obtenir la confluence de $\to_{\beta\mathcal{R}}$ à partir de sa confluence locale.

Le « schéma général » de Jouannaud et Okada [JO91] est une condition suffisante pour la normalisation forte (typée) de $\to_{\beta\mathcal{R}}$ pour les systèmes d'ordre supérieur. Étendu au λ-cube dans [BFG97] (via un passage par les types intersection [BF96]), puis dans [Bla05b], il donne, pour les systèmes applicatifs à gauche vérifiant les conditions du théorème 8.5 de [BFG97], une condition suffisante pour la confluence de $\to_{\beta\mathcal{R}}$ dans le λ-cube.

D'autre part, notons que pour les systèmes algébriquement typés, le théorème 6.1.9 s'étend à la normalisation forte. Nous revenons sur ce résultat à la section 9.4.

6.1.3 Importance de l'arité

Daniel Dougherty a remarqué dans [Dou92], pour les systèmes de réécriture algébriquement typés dans le système F (voir définition 3.6.7), que le rôle du typage dans le théorème 6.1.9 est essentiellement d'assurer la normalisation de $\to_\beta$ et la projection des dérivations de réécriture sur les termes β-normaux. Ces deux propriétés peuvent être obtenues par des conditions plus faibles que le typage. Pour ce faire, Dougherty introduit une restriction d'arité aux applications des symboles de fonction.

Rappelons que d'après la définition 3.1.7, les termes algébriques sur Σ sont donnés par la grammaire suivante :

$$t \quad ::= \quad x \quad | \quad f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m}$$

où $x \in \mathcal{X}$, $f \in \Sigma_n$ et $m \geq 0$.

Définition 6.1.11 (TRS avec arité applicative) *Un terme algébrique* $t \in \Lambda(\Sigma)$ respecte $a : \Sigma \to \mathbb{N}$ *si, récursivement,*

— $t = x \in \mathcal{X}$, *ou*

— $t = f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m}$ *avec* $m \leq a(f)$ *et* t_i *respecte* a *pour tout* $i \in \{1, \dots, n+m\}$.

Un TRS $\mathcal{R}$ *algébrique à gauche sur* $\Lambda(\Sigma)$ respecte $a : \Sigma \to \mathbb{N}$ *si pour toute règle* $l \mapsto_\mathcal{R} r$, l *et* r *respectent* a *et* $l = f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m}$ *avec* $a(f) = m$.

Proposition 6.1.12 *Soit* t *un terme algébrique et* σ *une substitution. Si* t *respecte* a *et* $\sigma(x)$ *respecte* a *pour tout* $x \in \mathcal{D}om(\sigma)$, *alors* $t\sigma$ *respecte* a.

On dit aussi que les TRS respectant une arité $a : \Sigma \to \mathbb{N}$ sont « compatibles » avec a. Les TRS avec arité empêchent les règles effondrantes (définies en 3.1.16) de créer des β-rédexes lorsqu'elles sont appliquées à des termes algébriques respectant cette arité.

Dougherty utilise des ensembles de termes fortement β-normalisants qui respectent une arité a compatible avec un système de réécriture $\mathcal{R}$. Ces ensembles de termes sont dits « $(\mathcal{R}, a)$-stables ».

Définition 6.1.13 ($(\mathcal{R}, a)$-stabilité) *Étant donné un TRS* $\mathcal{R}$, *un ensemble de termes* $R \subseteq \mathcal{SN}_\beta$ *est* $(\mathcal{R}, a)$-stable *si*

— *pour tout* $t \in R$, *si* $t \to_{\beta\mathcal{R}} u$ *alors* $u \in R$,

— *pour tout* $t \in R$, *si* u *est un sous-terme de* t *alors* $u \in R$,

— *pour tout terme* $f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m} \in R$, *on a* $m \leq a(f)$.

Un terme est $(\mathcal{R}, a)$*-stable s'il appartient à un ensemble* $(\mathcal{R}, a)$*-stable.*

Lorsque le contexte le permet, nous parlons de « $\mathcal{R}$-stabilité », ou même « stabilité » au lieu de « $(\mathcal{R}, a)$-stabilité ».

Exemple 6.1.14 *La règle* $\mathsf{id}\ x \mapsto x$ *définissant la fonction* id *implique que* $a(\mathsf{id}) = 1$. *De ce fait, le terme* $\mathsf{id}\ \mathsf{Id}\ y$ *n'est pas stable. C'est un terme β-normal qui se réduit vers le β-rédex* $\mathsf{Id}\ y$. *Il est alors facile de construire un terme non stable qui met en défaut la préservation de la confluence. Ainsi, le terme* $\mathsf{minus}\ (\mathsf{id}\ \omega_S\ \omega_S)\ (\mathsf{id}\ \omega_S\ \omega_S)$ *est β-normal mais se* $\mapsto_{\mathsf{id}}$*-réduit vers* $\mathsf{minus}\ Y_S\ Y_S$. *Comme vu à l'exemple 6.1.1, la relation* $\to_{\beta\mathsf{minus}}$ *n'est pas confluente à partir de ce terme.*

Remarque 6.1.15 *Le terme* $(\mathsf{id}\ \mathsf{id})\ \mathsf{id}$ *n'est pas stable. Ainsi, comme le remarque Dougherty dans [Dou92], la condition de stabilité prend bien en compte la réécriture algébrique, mais une adaptation est nécessaire pour prendre en compte la réécriture polymorphe.*

Dans [Dou92], Dougherty prouve la préservation de la confluence sur les termes *curryfiés*, c'est-à-dire dans lesquels tous les symboles ont une arité algébrique nulle : $\Sigma_i = \emptyset$ pour tout $i > 0$, donc $\Sigma = \Sigma_0$.

En fait, le plus important avec les termes stables est d'obtenir un analogue au lemme 6.1.8, c'est-à-dire d'obtenir la projection des dérivations de réécriture sur les formes β-normales. C'est le lemme 6.1.16, qui est implicite dans la preuve du théorème 4.7. de [Dou92]. On le montre par induction sur $\to_\beta$ (c'est une relation bien fondée dans les ensembles stables). Nous étendons ce résultat au cas de la réécriture conditionnelle et aux termes *faiblement* β-normalisants en 7.1.2 et 7.2.2.

Lemme 6.1.16 *Soit* $\mathcal{R}$ *est un TRS algébrique sur* $\Lambda(\Sigma)$ *qui respecte* $a : \Sigma \to \mathbb{N}$. *Si* t *est* $(\mathcal{R}, a)$*-stable et* $t \to_\mathcal{R} u$ *alors* $\beta nf(t) \to^*_\mathcal{R} \beta nf(u)$.

On en déduit aisément la préservation de la confluence sur les termes stables.

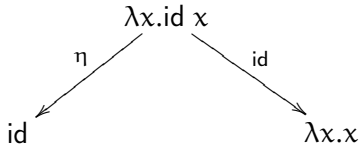
Théorème 6.1.17 ([Dou92]) *Si* $\mathcal{R}$ *est un TRS algébrique confluent sur* $\Lambda(\Sigma)$ *qui respecte* a, *alors* $\to_{\beta\mathcal{R}}$ *est confluent sur tout ensemble* $(\mathcal{R}, a)$*-stable.*

Dougherty montre aussi que la confluence est préservée par curryfication vers les ensembles stables. De plus, le théorème 6.1.17 est étendu au cas de la normalisation forte.

6.2 Consistance de la réécriture algébrique extensionnelle

Étant donné un système de réécriture $\mathcal{R}$ algébrique à gauche et confluent sur $\Lambda(\Sigma_0)$, il est bien connu que la relation $\to_{\eta\mathcal{R}}$ peut ne pas être confluente.

Exemple 6.2.1 *Avec la fonction* id *définie en 3.1.5 on a la paire critique injoignable suivante :*



La relation $\to_{\eta\mathcal{R}}$ *n'est donc pas confluente.*

Cependant, on peut remarquer que les termes $\lambda x.x$ et id de l'exemple 6.2.1 sont *extensionnellement* égaux par $\beta\mathcal{R}$-conversion : pour tout t, on a $(\lambda x.x)t =_{\beta\mathcal{R}} \mathsf{id}\ t$ (donc $\lambda x.x =_{\mathcal{R}_{EXT}} \mathsf{id}$ au sens de 3.5.3). Si $\to_{\beta\mathcal{R}}$ est confluente, cela veut dire que la paire critique de l'exemple 6.2.1, bien qu'injoignable, ne fait peut-être pas pour autant s'effondrer la relation $=_{\beta\eta\mathcal{R}}$.

On peut alors se demander ce que l'on rajoute vraiment lorsqu'on considère $=_{\beta\eta\mathcal{R}}$, et en particulier si la consistance de $=_{\beta\mathcal{R}}$ implique la consistance de $=_{\beta\eta\mathcal{R}}$. Ce résultat est

montré dans [BT88] (resp. [BTG89, BTG94]) pour les termes typés de $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B},\Sigma_0)$ (resp. de $\boldsymbol{\lambda}_2(\mathcal{B},\Sigma_0)$), en passant par des termes en forme normale $\beta\eta$-longue. Nous donnons une preuve simple de ce résultat à la section 6.2.2.

6.2.1 Confluence de la réécriture avec l'éta-expansion typée

On peut aborder la question de manière algorithmique, et essayer d'engendrer la congruence $=_{\beta\eta\mathcal{R}}$ en ajoutant à $\to_{\beta\mathcal{R}}$ une règle de réécriture, tout en conservant une relation de réécriture confluente. C'est l'approche de [DCK94, Xi96], où est définie une règle d'η-expansion, dont l'ajout à $\to_{\beta\mathcal{R}}$ préserve la confluence et la normalisation forte, sur les termes typés, lorsque les règles de réécriture sont typables algébriquement et linéaires à gauche.

La règle utilisée pour l'η-expansion est conditionnée par le type des termes, qui doit donc être unique. On se place donc dans des systèmes *à la Church* pour lesquels on utilise le lemme 3.3.20. La règle d'η-expansion conditionnelle est la suivante :

$$t \quad \mapsto_{\eta^+} \quad \lambda x.tx \qquad \text{si} \qquad t : U \Rightarrow T \quad \text{et} \quad x \in \mathcal{X}_U \setminus \mathsf{FV}(t) \quad \text{et} \quad t \neq \lambda y.u\ .$$

Remarque 6.2.2 *La règle η^+ n'est pas confluente sur les termes non quotientés par α-conversion : si $x, y \in \mathcal{X}_U \setminus \mathsf{FV}(t)$, alors*

$$\lambda y.ty \quad \leftarrow_{\eta^+} \quad t \quad \to_{\eta^+} \quad \lambda x.tx\ .$$

Mais comme pour tout $y, x \in \mathcal{X}_U \setminus \mathsf{FV}(t)$ on a $\lambda y.ty =_\alpha \lambda x.tx$, elle est confluente sur $\Lambda^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$.

Le résultat suivant a été montré dans [DCK94, DCK96] pour $\boldsymbol{\lambda}^{\mathcal{C}h}_{\times}(\mathcal{B},\Sigma_0)$ puis étendu à $\boldsymbol{\lambda}^{\mathcal{C}h}_{2}(\mathcal{B},\Sigma_0)$ dans [Xi96].

Théorème 6.2.3 ([DCK94, Xi96]) *Soit $ty \in \{\times, 2\}$ et $\mathcal{R}$ est un TRS algébriquement typé dans $\boldsymbol{\lambda}^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$ et linéaire à gauche. Si $\to_{\mathcal{R}}$ est confluente sur $\Lambda^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$, alors $\to_{\beta\eta^+\mathcal{R}}$ est confluente sur $\Lambda^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$.*

Comme $=_{\eta^+}$ et $=_\eta$ coïncident (pour $\boldsymbol{\lambda}^{\mathcal{C}h}_{\times}(\mathcal{B},\Sigma_0)$, c'est le théorème 2.9 dans [DCK96]), on déduit facilement la consistance de $=_{\beta\eta\mathcal{R}}$ à partir de la confluence de $\mathcal{R}$.

Corollaire 6.2.4 *Soit $ty \in \{\times, 2\}$ et $\mathcal{R}$ est un TRS algébriquement typé dans $\boldsymbol{\lambda}^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$ et linéaire à gauche. Si $\to_{\mathcal{R}}$ est confluente sur $\Lambda^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$, alors $=_{\beta\eta\mathcal{R}}$ est consistante sur $\Lambda^{\mathcal{C}h}_{ty}(\mathcal{B},\Sigma_0)$.*

Notons que le corollaire 6.2.4 ne s'applique (directement) qu'aux systèmes *à la Church*.

6.2.2 Consistance de la réécriture avec éta-conversion

Dans cette section, nous donnons une nouvelle preuve du fait que la relation $=_{\beta\eta\mathcal{R}}$ est consistante sur les termes typé dans le système F, dès lors que $\mathcal{R}$ est un système de réécriture algébriquement typé tel que $=_{\mathcal{R}}$ est consistante.

En comparaison de [BTG89, BTG94] notre approche a l'avantage de ne pas nécessiter de connaître le type des variables sous les abstractions : on peut se placer dans le cadre du λ-calcul typé *à la Curry*. Il nous semble de plus qu'elle est plus simple, dans la mesure où nous n'avons pas besoin de la notion de forme normale βη-longue.

Nous ne définissons pas de relation confluente simulant l'η-réduction. La propriété principale est le lemme 6.2.5, qui repose sur un raisonnement équationnel suggéré par l'exemple 6.2.1. Une conséquence importante de ce lemme est une preuve syntaxique de l'extension du théorème 6.1.7 à la βη-conversion.

Nous ne prouvons le lemme 6.2.5 que pour $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma_0)$. La preuve semble s'étendre sans difficulté au cas de $\boldsymbol{\lambda}_{\mathsf{ty}}(\mathcal{B}, \Sigma_0)$ pour $\mathsf{ty} \in \{\times, \times 2, \times 2\Lambda, 2\Lambda\}$.

Lemme 6.2.5 *Soit* (Σ_0, τ) *une signature algébrique sur* $\mathcal{T}_2(\mathcal{B})$ *et* $\mathcal{R}$ *un TRS algébriquement typé dans* $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma_0, \tau)$. *Soient* $t, u \in \Lambda_2(\Sigma_0)$ *deux termes* β*-normaux tels qu'il existe* Γ *à valeurs dans* $\mathcal{B}$ *et* $B \in \mathcal{B}$ *tels que* $\Gamma \vdash t : B$ *et* $\Gamma \vdash u : B$. *Si* $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma_0) \vdash t =_{\beta\eta\mathcal{R}} u$ *est vrai dans* $[\Gamma \vdash B]$ *alors* $\beta\mathrm{nf}(t) =_{\mathcal{R}} \beta\mathrm{nf}(u)$.

Preuve. On raisonne par induction sur la longueur de $t \leftrightarrow^*_{\beta\eta\mathcal{R}} u$. Le cas de base $t = u$ est trivial. Sinon, il existe $v \in [\Gamma \vdash B]$ tel que $t \leftrightarrow_{\beta\eta\mathcal{R}} v \leftrightarrow^*_{\beta\eta\mathcal{R}} u$. Par hypothèse d'induction on a $\beta\mathrm{nf}(v) \leftrightarrow^*_{\mathcal{R}} \beta\mathrm{nf}(u)$.

— Si $t \leftrightarrow_\beta v$ alors $\beta\mathrm{nf}(t) = \beta\mathrm{nf}(v)$, donc $\beta\mathrm{nf}(t) \leftrightarrow^*_{\mathcal{R}} \beta\mathrm{nf}(u)$.

— Si $t \leftrightarrow_\eta v$, alors on a $\beta\mathrm{nf}(t) \downarrow_{\beta\eta} \beta\mathrm{nf}(u)$ par le théorème 3.5.4, d'où $\beta\mathrm{nf}(t) \downarrow_\eta \beta\mathrm{nf}(u)$ par le lemme 3.5.5.
D'autre part, $\beta\mathrm{nf}(t), \beta\mathrm{nf}(u) \in [\Gamma \vdash B]$ par le lemme 3.6.9. Or, la proposition 3.6.4 dit qu'un terme β-normal dans $[\Gamma \vdash B]$ est algébrique, ce qui veut dire que $\beta\mathrm{nf}(t)$ et $\beta\mathrm{nf}(u)$ sont η-normaux, donc $\beta\mathrm{nf}(t) = \beta\mathrm{nf}(u)$.
On en déduit que $\beta\mathrm{nf}(t) = \beta\mathrm{nf}(v) \leftrightarrow^*_{\mathcal{R}} \beta\mathrm{nf}(u)$.

— Si $t \leftrightarrow_{\mathcal{R}} v$, alors par le lemme 6.1.8, on a $\beta\mathrm{nf}(t) \leftrightarrow^*_{\mathcal{R}} \beta\mathrm{nf}(v) \leftrightarrow^*_{\mathcal{R}} \beta\mathrm{nf}(u)$. □

Théorème 6.2.6 (Extension conservative) *Soit* (Σ_0, τ) *une signature algébrique sur* $\mathcal{T}_2(\mathcal{B})$ *et* $\mathcal{R}$ *un TRS algébriquement typé dans* $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma_0, \tau)$. *Pour tout* t, u *algébriquement typés on a*

$$\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma_0) \vdash t =_{\beta\eta\mathcal{R}} u \qquad \textit{si et seulement si} \qquad \mathcal{T}\mathrm{er}(\Sigma_0, \mathcal{X}) \vdash t =_{\mathcal{R}} u\ .$$

Rappelons que par la définition 3.6.10, $\mathcal{T}\mathrm{er}(\Sigma_0, \mathcal{X}) \vdash t \equiv u$ implique que $t \equiv u$ soit vrai dans $[\Gamma \vdash B]$ pour un $B \in \mathcal{B}$ et un contexte Γ à valeurs dans $\mathcal{B}$.

Remarque 6.2.7 *En particulier, la consistance de* $=_{\mathcal{R}}$ *sur* $\mathcal{T}\mathrm{er}(\Sigma_0, \mathcal{X})$ *implique la consistance de* $=_{\beta\eta\mathcal{R}}$ *sur les termes bien typés de* $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma_0)$.

Chapitre 7

Réécriture conditionnelle combinée au lambda-calcul

Dans ce chapitre, nous étendons à la réécriture conditionnelle des résultats de préservation de la confluence présentés à la section 6.1.

Étant donné un système conditionnel $\mathcal{R}$, nous considérons deux façons de combiner la réécriture à la β-réduction :

— La plus simple est la relation $\to_{\beta\mathcal{R}^{jo}}$. C'est l'union la réécriture conditionnelle par joignabilité et de la β-réduction ; on n'utilise pas $\to_\beta$ dans l'évaluation des conditions.

— La plus intéressante est probablement $\to_{\beta\mathcal{R}(\beta)^{jo}}$. C'est l'union de la réécriture conditionnelle par β-joignabilité et de la β-réduction ; les conditions des règles peuvent être évaluées en utilisant $\to_\beta$.

Notre but est d'obtenir des conditions suffisantes permettant d'assurer la confluence de $\to_{\beta\mathcal{R}(\beta)^{jo}}$ à partir de la confluence de $\to_{\mathcal{R}^{jo}}$. Cependant, mis à part pour les systèmes *orthonormaux* présentés et étudiés en 7.2.3, nous ne savons pas comment déduire la confluence de $\to_{\beta\mathcal{R}(\beta)^{jo}}$ de celle de $\to_{\mathcal{R}^{jo}}$ sans utiliser la confluence de $\to_{\beta\mathcal{R}^{jo}}$. Nos résultats sont donc répartis en deux catégories :

— ceux concernant la confluence de $\to_{\beta\mathcal{R}^{jo}}$, présentés en 7.1 ;

— ceux concernant la confluence de $\to_{\beta\mathcal{R}(\beta)^{jo}}$, présentés en 7.2.

Les résultats présentés dans ce chapitre ont été publiés dans [BKR06].

7.1 Confluence de la béta-réduction avec la réécriture conditionnelle

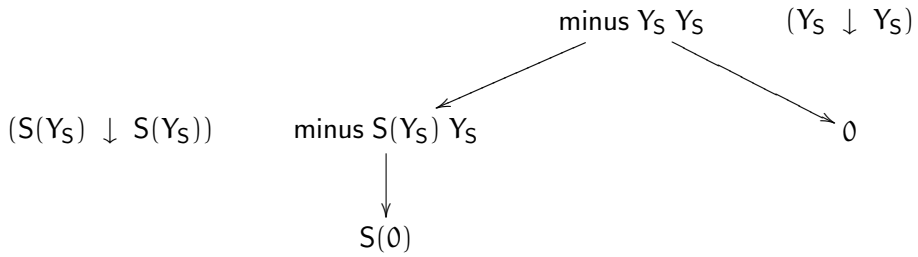
Le contre exemple 6.1.1 de Breazu-Tannen et Meyer [BTM87] peut être très facilement adapté aux systèmes conditionnels linéaires à gauche.

Exemple 7.1.1 *Considérons le système conditionnel*

$$x = y \ \supset\ \mathsf{minus}\ x\ y \ \mapsto\ 0 \qquad\qquad x = \mathsf{S}(y) \ \supset\ \mathsf{minus}\ x\ y \ \mapsto\ \mathsf{S}(0)\ .$$

Ce système est linéaire à gauche, mais les conditions de ses règles testent l'égalité de termes ouverts. La relation de réécriture conditionnelle par joignabilité qui en est issue

forme, avec $\to_\beta$*, le pic injoignable suivant :*



Comme à la section 6.1, on considère deux moyens de s'en sortir : restreindre les règles de réécriture ou restreindre la β-réduction.

7.1.1 Systèmes linéaires à gauche et semi-clos

L'important avec la condition de linéarité gauche est qu'elle interdit à la réécriture de comparer des termes arbitraires, ce qui empêche en particulier les comparaisons de termes « infinis », comme Ys. Nous introduisons une condition, appelée « semi-clôture » qui interdit aux systèmes conditionnels de tester l'égalité de termes arbitraires *dans leurs conditions.*

Définition 7.1.2 (Systèmes conditionnels semi-clos) *Un système conditionnel* $\mathcal{R}$ *est* semi-clos *si pour toute règle*

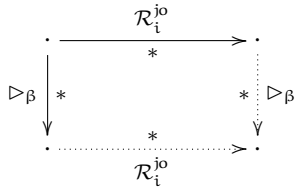
$$d_1 = c_1 \wedge \ldots \wedge d_n = c_n \supset l \mapsto_{\mathcal{R}} r\,,$$

le terme c_i *est applicatif et clos pour tout* $i \in \{1, \ldots, n\}$.

Dans une règle semi-close $\vec{d} = \vec{c} \supset l \mapsto r$, il serait sûrement intéressant, en pratique, de normaliser $\vec{c}$ a priori, et ainsi d'obtenir un système normal, au sens de 4.1.8.

Étant donné un système semi-clos et linéaire à gauche, on montre que la confluence de $\to_{\beta\mathcal{R}}$ suit de la confluence de $\to_{\mathcal{R}}$. Le lemme principal est l'extension à la réécriture conditionnelle de la commutation de $\lhd_\beta$ et de $\to_{\mathcal{R}}$, c'est à dire du lemme 6.1.3. Comme d'habitude avec la réécriture conditionnelle, on raisonne par induction sur la stratification.

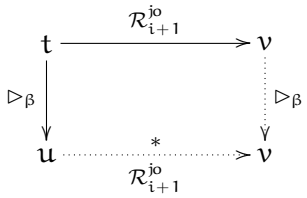
Lemme 7.1.3 *Si* $\mathcal{R}$ *est un système conditionnel semi-clos, linéaire à gauche et applicatif à droite, alors* $\to_{\mathcal{R}_i^{jo}}$ *commute avec* $\rhd_\beta$ *pour tout* $i \in \mathbb{N}$*. En image :*



Remarquons que $\to_{\mathcal{R}}$ est issue du système algébrique à gauche et applicatif à droite $\{(l, r) \mid \vec{d} = \vec{c} \supset l \mapsto r \in \mathcal{R}\}$ au sens de la définition 3.1.9.

PREUVE. On raisonne par induction sur $i \in \mathbb{N}$. Le cas de base $i = 0$ est trivial. Soit $i \geq 0$ et supposons que $\to_{\mathcal{R}_i^{jo}}$ commute avec $\to_\beta$. On montre que $\to_{\mathcal{R}_{i+1}^{jo}}$ commute avec $\to_\beta$.

On commence par montrer que pour tout $t, u, v \in \Lambda(\Sigma)$, si $u \lhd_\beta t \to_{\mathcal{R}_{i+1}^{jo}} v$ alors il existe un terme w tel que $u \to^*_{\mathcal{R}_{i+1}^{jo}} w \lhd_\beta v$. En image :



(7.1)

On raisonne exactement comme au lemme 6.1.3. La seule différence est lorsque t est le $\to_{\mathcal{R}_{i+1}^{jo}}$-rédex contracté dans le pas $t \to_{\mathcal{R}_{i+1}^{jo}} v$ (cela correspond au cas (i) de la preuve du lemme 6.1.3).

Dans ce cas, il existe une règle $\vec{d} = \vec{c} \supset l \mapsto r \in \mathcal{R}$ et une substitution σ telle que $t = l\sigma$, $v = r\sigma$, ainsi que des termes $\vec{v}$ tels que $\vec{d}\sigma \to^*_{\mathcal{R}_i^{jo}} \vec{v} \leftarrow^*_{\mathcal{R}_i^{jo}} \vec{c}\sigma$. Comme $\mathcal{R}$ est semi-clos, les termes $\vec{c}$ sont clos et applicatifs, donc $\vec{c}\sigma = \vec{c}$ et selon la proposition 3.1.10, les termes $\vec{v}$ sont applicatifs car $\mathcal{R}$ est applicatif à droite.

De même qu'en 6.1.3.(i), comme l est linéaire et algébrique, on déduit de la proposition 6.1.2 qu'il existe une substitution σ' telle que $\sigma \rhd_\beta \sigma'$ et $u = l\sigma'$. Il s'en suit par le lemme 2.3.6 que $r\sigma \rhd_\beta r\sigma'$ et $\vec{d}\sigma \rhd_\beta \vec{d}\sigma'$, soit $\vec{d}\sigma \to^*_\beta \vec{d}\sigma'$ par le lemme 5.1.12.

Pour conclure que le terme $w =_{\text{def}} r\sigma'$ convient, il reste à montrer que $l\sigma' \to_{\mathcal{R}_{i+1}^{jo}} r\sigma'$, c'est-à-dire que $\vec{d}\sigma' \downarrow_{\mathcal{R}_i^{jo}} \vec{c}$. Or, on a $\vec{d}\sigma' \leftarrow^*_\beta \vec{d}\sigma \to^*_{\mathcal{R}_i^{jo}} \vec{v}$, donc par hypothèse il existe $\vec{w}$ tels que $\vec{d}\sigma' \to^*_{\mathcal{R}_i^{jo}} \vec{w} \leftarrow^*_\beta \vec{v}$. Il s'en suit que $\vec{d}\sigma' \to^*_{\mathcal{R}_i^{jo}} \vec{v}$, les termes $\vec{v}$ étant applicatifs donc en forme β-normale. On a donc $\vec{d}\sigma' \downarrow_{\mathcal{R}_i^{jo}} \vec{c}$, et on en déduit que $l\sigma' \to_{\mathcal{R}_{i+1}^{jo}} r\sigma'$.

On en déduit que $\to_{\mathcal{R}_{i+1}^{jo}}$ commute avec $\to_\beta$ en appliquant le lemme 2.1.15 à (7.1). □

On peut alors appliquer le lemme de Hindley-Rosen à $\to_{\mathcal{R}^{jo}}$ et $\to_\beta$, et ainsi obtenir la confluence de $\to_{\beta\mathcal{R}^{jo}}$.

Théorème 7.1.4 (Confluence de $\to_{\beta\mathcal{R}^{jo}}$) *Soit $\mathcal{R}$ un système conditionnel semi-clos, linéaire à gauche et applicatif à droite. Si $\to_{\mathcal{R}^{jo}}$ est confluent sur $\Lambda(\Sigma)$, alors $\to_{\beta\mathcal{R}^{jo}}$ est confluent sur $\Lambda(\Sigma)$.*

PREUVE. Le lemme 7.1.3 implique la commutation de $\to_{\mathcal{R}^{jo}}$ avec $\to_\beta$ et comme $\to_{\mathcal{R}^{jo}}$ et $\to_\beta$ sont confluentes, on en déduit la confluence de $\to_{\mathcal{R}\beta}$ par lemme de Hindley-Rosen (théorème 2.1.14). □

Comparaison avec le théorème 6.1.4 — [Mül92]. Le résultat original de Müller n'est énoncé que pour les systèmes algébriques, mais la preuve s'étend sans problèmes aux systèmes avec membres droits non-algébriques. C'est d'ailleurs sous cette forme que nous présentons le théorème 6.1.4.

Cependant, pour la réécriture conditionnelle, nous nous limitons aux systèmes applicatifs à droite, ce qui permet par la proposition 3.1.10 d'avoir la préservation des terme applicatifs par réécriture. C'est nécessaire avec la réécriture conditionnelle, car sinon le lemme 7.1.3 peut être mis en défaut.

Exemple 7.1.5 *Considérons le système :*

$$\mathsf{h} \;\mapsto\; \lambda x.x \qquad\qquad x \;=\; \mathsf{h}\,\mathsf{a} \;\supset\; \mathsf{g}\,x \;\mapsto\; \mathsf{a}\,.$$

On a les réductions

$$\mathsf{g}\,\mathsf{a} \quad \leftarrow_{\beta} \quad \mathsf{g}\,((\lambda x.x)\mathsf{a}) \quad \rightarrow_{\mathcal{R}^{jo}} \quad \mathsf{a}\,,$$

mais $\mathsf{g}\,\mathsf{a}$ *est une forme* $\rightarrow_{\mathcal{R}^{jo}}$*-normale car* $(\lambda x.x)\mathsf{a}$ *est une forme* $\rightarrow_{\mathcal{R}^{jo}}$*-normale.*

Pour pouvoir utiliser des membres droits arbitraires, il faut donc restreindre les conditions. Ainsi, on se place dans le cas de la réécriture normale.

Théorème 7.1.6 (Extension de [Mül92]) *Soit* $\mathcal{R}$ *un système conditionnel semi-clos et linéaire à gauche tel que* $\rightarrow_{\mathcal{R}^{jo}}$ *soit une relation de réécriture conditionnelle normale. Si* $\rightarrow_{\mathcal{R}^{jo}}$ *est confluent sur* $\Lambda(\Sigma)$*, alors* $\rightarrow_{\beta\mathcal{R}^{jo}}$ *est confluent sur* $\Lambda(\Sigma)$*.*

Preuve. Comme pour le théorème 7.1.4. L'équivalent du lemme 7.1.3 se prouve sans difficulté. La seule différence est la suivante. Supposons que $\rightarrow_{\mathcal{R}_i^{jo}}$ commute avec $\rightarrow_{\beta}$ et que, pour une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et une substitution σ on ait $l\sigma' \lhd_{\beta} l\sigma \rightarrow_{\mathcal{R}_{i+1}^{jo}} r\sigma$. Comme $\rightarrow_{\mathcal{R}^{jo}}$ est normale, on a $\vec{d}\sigma \rightarrow^*_{\mathcal{R}_i^{jo}} \vec{c}$, et par hypothèse d'induction il existe $\vec{c}'$ tels que $\vec{d}\sigma' \rightarrow^*_{\mathcal{R}_i^{jo}} \vec{c}' \leftarrow^*_{\beta} \vec{c}$. Or, comme $\mathcal{R}$ est semi-clos, les termes de $\vec{c}$ sont algébriques donc β-normaux. On a donc $\vec{d}\sigma' \rightarrow^*_{\mathcal{R}_i^{jo}} \vec{c}$, d'où $l\sigma' \rightarrow_{\mathcal{R}_{i+1}^{jo}} r\sigma'$. □

7.1.2 Confluence sur les termes faiblement normalisants

Nous nous intéressons maintenant aux systèmes qui ne satisfont pas les conditions de linéarité gauche et de semi-clôture.

Le résultat principal de cette section, le théorème 7.1.16, s'inscrit dans la lignée des travaux de Dougherty [Dou92], et plus précisément du théorème 6.1.17. On veut obtenir une propriété similaire à celle énoncée au lemme 6.1.16, c'est-à-dire :

$$\begin{array}{ccc} t & \xrightarrow[*]{\beta\cup\mathcal{R}^{jo}} & u \\ {\scriptstyle\beta}\,\vdots\,{\scriptstyle *} & & {\scriptstyle *}\,\vdots\,{\scriptstyle\beta} \\ \downarrow & & \downarrow \\ \beta\mathrm{nf}(t) & \xrightarrow[\mathcal{R}^{jo}]{*} & \beta\mathrm{nf}(u) \end{array} \tag{7.2}$$

Outre l'adaptation à la réécriture conditionnelle, notre apport principal est un affaiblissement notable des hypothèses du théorème : au lieu de manipuler uniquement des termes

fortement β-normalisants qui respectent une arité compatible avec le système de réécriture, nous travaillons avec des termes faiblement β-normalisants, dont seules les formes β-normales sont supposées respecter l'arité du système de réécriture.

Comme remarqué à l'exemple 7.1.1, les combinateurs de point fixe du λ-calcul mettent en défaut la commutation de $\to_\beta$ avec $\to_{\mathcal{R}^{jo}}$ lorsque que les règles utilisent des tests d'égalité entre des termes ouverts. Lorsque l'on se restreint aux termes faiblement β-normalisants, on peut projeter la réécriture sur les formes β-normales, et ainsi éliminer les points fixes dès lors qu'ils ne sont pas utiles à la réduction.

Pour cela, on utilise les propriétés suivantes.

— Tout d'abord, les formes β-normales doivent être stables par réécriture. On suppose donc que les membres droits sont algébriques et on ne considère que des termes et des règles qui respectent une fonction d'arité applicative $\mathfrak{a} : \Sigma \to \mathbb{N}$ (voir définition 6.1.11).

  On suppose de plus que les conditions sont algébriques. Ainsi, étant données une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et deux substitutions σ, σ' telles que $\sigma' = \beta nf(\sigma)$, on a $\beta nf(l\sigma) = l\sigma'$, $\beta nf(r\sigma) = r\sigma'$, $\beta nf(\vec{d}\sigma) = \vec{d}\sigma'$ et $\beta nf(\vec{c}\sigma) = \vec{c}\sigma'$.

— Ensuite, on a besoin de β-réductions normalisantes qui commutent avec la réécriture. Pour cela on utilise la β-réduction *normale*, qui donne la forme β-normale d'un terme lorsqu'elle existe (théorème 3.2.9).

Commençons par étendre la notion de TRS avec arité applicative à la réécriture conditionnelle.

Définition 7.1.7 (Système conditionnel avec arité applicative) *Soit* $\mathfrak{a} : \Sigma \to \mathbb{N}$. *On dit d'une règle conditionnelle* $\vec{d} = \vec{c} \supset l \mapsto r$,

— *qu'elle* respecte faiblement $\mathfrak{a}$ *si* l *et* r *respectent* $\mathfrak{a}$ *et si*

$$l = f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m} \qquad \textit{avec} \qquad m = \mathfrak{a}(f) \ ,$$

— *qu'elle* respecte $\mathfrak{a}$ *si elle respecte faiblement* $\mathfrak{a}$ *et si les termes de* $\vec{d}, \vec{c}$ *respectent* $\mathfrak{a}$.

Un système conditionnel $\mathcal{R}$ *respecte (faiblement)* $\mathfrak{a}$ *si toutes ses règles respectent (faiblement)* $\mathfrak{a}$.

Comme vu en 6.1.14, en présence de règles effondrantes, les termes ne respectant pas d'arité applicative compatible avec les règles de réécriture peuvent faire perdre la confluence.

Cependant, nous ne nous limitons pas aux termes stables. Si un terme a une forme β-normale sans être fortement β-normalisant, alors la β-réduction normale n'évalue jamais ses sous-termes non normalisants. De ce fait, ils peuvent ne pas être stables sans pour autant empêcher la projection de la réécriture sur les formes β-normales.

Définition 7.1.8 *Soit* $\mathfrak{a} : \Sigma \to \mathbb{N}$. *On désigne par* $\mathcal{AN}_\mathfrak{a}$ *l'ensemble des termes de* $\mathcal{WN}_\beta$ *dont la forme* β*-normale respecte* $\mathfrak{a}$.

Par le théorème 3.2.9, la réduction normale $\to_n$ est bien fondée sur $\mathcal{AN}_\mathfrak{a}$. Dans notre preuve nous utilisons cette propriété de manière indirecte. En effet, nous avons besoin d'une induction sur $\to_n$, mais aussi de pouvoir accéder aux sous-termes. Ceci est formalisé

dans la relation $\succ_h$, définie en 7.1.9, qui conjugue β-réduction de tête et accessibilité des sous-termes. La bonne fondaison de $\succ_h$ sur $\mathcal{AN}_a$, prouvée au lemme 7.1.11, repose sur la bonne fondaison de $\to_n$ sur $\mathcal{AN}_a$.

L'autre point clef de la preuve est le fait que les β-réductions de tête commutent bien avec la réécriture algébrique à gauche (lemme 7.1.14), voir par exemple [BFG97].

Rappelons que par le lemme 3.2.6, tout λ-terme est, récursivement, sous une des trois formes suivantes :

$$\lambda x_1 \dots x_p.(\lambda x.t)u t_1 \dots t_n \quad (a)$$
$$\lambda x_1 \dots x_p.x\ t_1 \dots t_n \quad (b)$$
$$\lambda x_1 \dots x_p.f(t_1, \dots, t_n)\ t_{n+1} \dots t_{n+m} \quad (c)$$

où $n, m, p \geq 0$, $x \in \mathcal{X}$ et $f \in \Sigma_n$. Nous définissons une relation sur $\Lambda(\Sigma)$ qui utilise cette notation des λ-termes ainsi que la relation sous-terme sur les λ-termes définie en 1.2.20. Comme noté en 1.2.21, cette relation ne préserve pas les classes d'α-équivalence.

Rappelons que si $t \in \mathcal{L}(\Sigma)$ alors $[t]$ est la classe d'α-équivalence de t et que par convention, on désigne $[x]$ par x. Tout $t \in \Lambda(\Sigma)$ peut donc être écrit sous la forme $t = [t']$ avec $t' \in \mathcal{L}(\Sigma)$. De plus, sur $\Lambda(\Sigma)$, les abstractions sont représentées par les fonctions $\lambda x.\_ : \Lambda(\Sigma) \to \Lambda(\Sigma)$ définies en 1.2.12 comme suit :

$$\lambda x.[t_1] = \{\lambda y.u_1 \mid u_1[y \mapsto z] =_\alpha t_1[x \mapsto z] \text{ pour tout } z \in \mathcal{X} \text{ sauf un nombre fini}\}\ .$$

Définition 7.1.9 *On dénote par $\succ_h$ la plus petite relation sur $\Lambda(\Sigma)$ telle que*

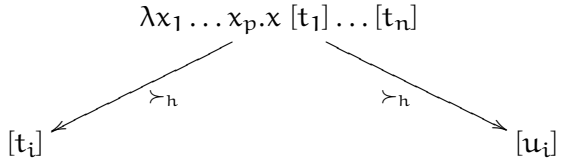
$$\lambda x_1 \dots x_p.(\lambda x.[t])[u][t_1] \dots [t_n] \succ_h \lambda x_1 \dots x_p.[t][[u]/x][t_1] \dots [t_n] \quad (a)$$
$$\lambda x_1 \dots x_p.x\ [t_1] \dots [t_n] \succ_h [t_i] \quad (b)$$
$$\lambda x_1 \dots x_p.f([t_1], \dots, [t_n])\ [t_{n+1}] \dots [t_{n+m}] \succ_h [t_j] \quad (c)$$

où $n, m, p \geq 0$, $x \in \mathcal{X}$, $f \in \Sigma_n$, $i \in \{1, \dots, n\}$, et $j \in \{1, \dots, n+m\}$.

Remarque 7.1.10 *Soient $t_1, \dots, t_n, u_1, \dots, u_n \in \mathcal{L}(\Sigma)$ et $i \in \{1, \dots, n\}$. On peut avoir*

$$\lambda x_1 \dots x_p.x\ t_1 \dots t_n \quad =_\alpha \quad \lambda y_1 \dots y_p.y\ u_1 \dots u_n \qquad \text{avec} \qquad u_i \quad \neq_\alpha \quad t_i\ .$$

Comme



il s'en suit que $\succ_h$ peut ne pas respecter les classes d'α-équivalence. Le problème est le même que celui évoqué en 1.2.15. Par exemple, on a

$$x \quad \prec_h \quad [\lambda x.x] \quad \succ_h \quad y\ .$$

L'important est que la relation $\succ_h$ soit bien fondée sur $\mathcal{AN}_a$.

Lemme 7.1.11 *Si* $t \succ_h u$ *et* $t \in \mathcal{WN}_\beta$ *alors* $u \in \mathcal{WN}_\beta$. *De plus,* $\succ_h$ *est bien fondée sur* $\mathcal{WN}_\beta$.

PREUVE. Pour le premier point, on raisonne par cas sur t, en utilisant le lemme 3.2.6.

$t = \lambda\vec{x}.(\lambda x.[v])[w][t_1]\dots[t_n]$. Dans ce cas, on a $t \to_n u$, d'où $u \in \mathcal{WN}_\beta$ par le théorème 3.2.9.

$t = \lambda\vec{x}.x\ [t_1]\dots[t_n]$. Dans ce cas, il existe $\lambda\vec{x}.y\ u_1 \in u_n \in \mathcal{L}(\Sigma)$ et $i \in \{1,\dots,n\}$ tels que $u = [u_i]$ avec

$$\lambda\vec{y}.y\ u_1 \dots u_n \quad =_\alpha \quad \lambda\vec{x}.x\ t_1 \dots t_n\ .$$

Donc $t = \lambda\vec{y}.y\ [u_1]\dots[u_n]$ et $[u_i] \in \mathcal{WN}_\beta$ car $t \in \mathcal{WN}_\beta$.

$t = f([t_1],\dots,[t_n])\ [t_{n+1}]\dots[t_{n+m}]$. Idem.

Montrons maintenant la bonne fondaison de $\succ_h$ sur $\mathcal{WN}_\beta$. Étant donné $t \in \mathcal{WN}_\beta$, on écrit $n(t)$ pour désigner le nombre de pas dans la réduction normale issue de t. On montre, en utilisant le lemme 3.2.6, que $t \succ_h u$ implique $(n(t), |t|) >_{lex} (n(u), |u|)$ pour tout $t \in \mathcal{WN}_\beta$.

$t = \lambda\vec{x}.(\lambda v)w\ t_1 \dots t_n$. Dans ce cas $t \to_n u$ donc $n(t) > n(u)$.

$t = \lambda\vec{x}.x\ t_1 \dots t_n$. Dans ce cas on a $n(t) \geq n(u)$ et $|t| > |u|$.

$t = \lambda\vec{x}.f(t_1,\dots,t_n)\ t_{n+1}\dots t_{n+m}$. Idem. □

Notre preuve préservation de la confluence repose sur une notion de réécriture parallèle emboîtée pour la réécriture conditionnelle, inspirée de la relation utilisée dans [Dou92]. C'est une réduction *à la* Tait et Martin-Löf : de la même manière que $\rhd_\beta$, elle permet de réduire en un pas des rédexes emboîtés les uns dans les autres. Notons que ce n'est pas la réduction parallèle utilisée dans le lemme 5.1.3.

Nous définissons les parallélisations emboîtées de manière générale pour la réécriture X-R-conditionnelle. Cela facilite grandement l'extension des résultats de cette section à la réécriture β-conditionnelle, que nous traitons en 7.2.2. Les propriétés énoncées en 7.1.13 et en 7.1.14 sont valables pour toutes les parallélisations emboîtées de relations de réécriture X-R-conditionnelle sur $\Lambda(\Sigma)$.

Définition 7.1.12 (Parallélisation emboitée) *Soit* $\mathcal{R}$ *un système conditionnel sur* $\Lambda(\Sigma)$, $\to_R$ *une relation de réécriture sur* $\Lambda(\Sigma)$ *et* $X \in \{se, jo, o\}$. *Pour tout* $i \in \{1,\dots,n\}$, *on dénote par* $\rhd_{\mathcal{R}(R)^X_i}$ *la clôture parallèle de la plus petite relation sur* $\Lambda(\Sigma)$ *vérifiant la règle suivante :*

$$(\rhd\mathcal{R})\ \frac{\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r \qquad l\sigma \to_{\mathcal{R}(R)^X_i} r\sigma \qquad \sigma \rhd_{\mathcal{R}(R)^X_i} \theta}{l\sigma \rhd_{\mathcal{R}(R)^X_i} r\theta}$$

La relation $\rhd_{\mathcal{R}(R)^X_i}$ *est la* parallélisation emboîtée *de la relation de réécriture* R*-conditionnelle* $\to_{\mathcal{R}(R)^X_i}$.

Rappelons que, dans le cas de la réécriture par joignabilité, $l\sigma \to_{\mathcal{R}(R)^{jo}_{i+1}} r\sigma$ provient de $\vec{d}\sigma \downarrow_{\mathcal{R}(R)^{jo}_i} \vec{c}\sigma$. De plus, $\sigma \rhd_{\mathcal{R}(R)^X_i} \theta$ dit que σ et θ on même domaine et que $\sigma(x) \rhd_{\mathcal{R}(R)^X_i} \theta(x)$ pour tout $x \in \mathcal{D}om(\sigma)$. Les parallélisations emboîtées ont des propriétés intéressantes :

Proposition 7.1.13 *Pour tout* $i \geq 0$*, on a*

(i) $\to_{\mathcal{R}(R)_i^X} \subseteq \rhd_{\mathcal{R}(R)_i^X} \subseteq \to^*_{\mathcal{R}(R)_i^X}$,

(ii) $[t \rhd_{\mathcal{R}(R)_i^X} u \quad \wedge \quad v \rhd_{\mathcal{R}(R)_i^X} w] \quad \Longrightarrow \quad v[t/x] \rhd_{\mathcal{R}(R)_i^X} w[u/x]$.

PREUVE.

(i) Par induction sur la définition de $\rhd_{\mathcal{R}(R)_i^X}$.

(ii) On raisonne par induction sur $v \rhd_{\mathcal{R}(R)_i^X} w$.

($\rhd$**Var**). Dans ce cas, $v = w = y \in \mathcal{X}$. Si $y = x$ alors $v[t/x] = t \rhd_{\mathcal{R}(R)_i^X} u = w[u/x]$, et sinon $v[t/x] = y \rhd_{\mathcal{R}(R)_i^X} y = w[u/x]$.

($\rhd$**Abs**), ($\rhd$**App**), ($\rhd$**Symb**). Par hypothèse d'induction.

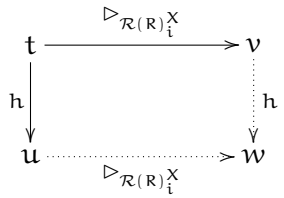
($\rhd\mathcal{R}$). Dans ce cas, il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et deux substitutions σ et θ telles que $v = l\sigma$, $w = r\theta$, $l\sigma \to_{\mathcal{R}(R)_i^X} r\sigma$ et $\sigma \rhd_{\mathcal{R}(R)_i^X} \theta$. Il s'en suit que $l\sigma[t/x] \to_{\mathcal{R}(R)_i^X} r\sigma[t/x]$, et, par hypothèse d'induction, on a $\sigma[t/x] \rhd_{\mathcal{R}(R)_i^X} \theta[u/x]$, d'où $l\sigma[t/x] \rhd_{\mathcal{R}(R)_i^X} r\theta[u/x]$. □

La propriété 7.1.13.(ii) est caractéristique des parallélisations emboîtées. Elle ne serait pas satisfaite par les clôtures parallèles directes $\to_{\|\mathcal{R}(R)_i^X}$ au sens de 2.3.5. Notons que comme $\rhd_{\mathcal{R}(R)_i^X}$ est réflexive,

$$t \quad \rhd_{\mathcal{R}(R)_i^X} \quad u \qquad \Longrightarrow \qquad v[t/x] \quad \rhd_{\mathcal{R}(R)_i^X} \quad v[u/x] \ .$$

La β-réduction de tête commute *en un pas* avec la réécriture algébrique à gauche. Cela a déjà été remarqué dans [BFG97] pour la réécriture non conditionnelle. Rappelons que par la définition 4.1.1, les membres gauches des règles de réécriture conditionnelle sont algébriques.

Lemme 7.1.14 *Pour tout* $i \geq 0$ *et tout* $t, u, v \in \Lambda(\Sigma)$*, si* $u \leftarrow_h t \rhd_{\mathcal{R}(R)_i^X} v$*, alors il existe* $w \in \Lambda(\Sigma)$ *tel que* $u \rhd_{\mathcal{R}(R)_i^X} w \leftarrow_h v$*. En image,*



PREUVE. Supposons que $u \leftarrow_h \lambda\vec{x}.(\lambda y.t_0)t_1 \dots t_n \rhd_{\mathcal{R}(R)_i^X} v$. Comme les règles ont des membres droits algébriques qui ne sont pas des variables, on a $v = \lambda\vec{x}.(\lambda y.v_0)v_1 \dots v_n$ avec $t_i \rhd_{\mathcal{R}(R)_i^X} v_i$ pour tout $i \in \{1, \dots, n\}$. D'autre part, $u = \lambda\vec{x}.t_0[t_1/y]t_2 \dots t_n$, et par la proposition 7.1.13.(ii) on a

$$\lambda\vec{x}.t_0[t_1/y]t_2 \dots t_n \quad \rhd_{\mathcal{R}(R)_i^X} \quad \lambda\vec{x}.v_0[v_1/y]v_2 \dots v_n$$

en un pas. D'où $u \rhd_{\mathcal{R}(R)_i^X} \lambda\vec{x}.v_0[v_1/x]v_2 \dots v_n \leftarrow_h v$. □

Nous prouvons maintenant (7.2) sur les termes de $\mathcal{AN}_a$ et pour la réécriture conditionnelle algébrique et compatible avec une fonction d'arité applicative a.

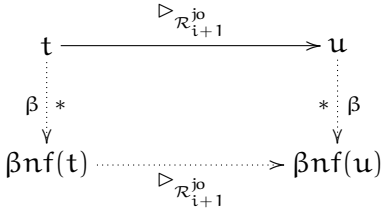
Pour tout $i \in \{1, \dots, n\}$, on utilise la parallélisation emboîtée de $\to_{\mathcal{R}_i^{jo}}$ notée $\rhd_{\mathcal{R}_i^{jo}}$.

Lemme 7.1.15 *Soit $\mathcal{R}$ un système conditionnel algébrique respectant $a : \Sigma \to \mathbb{N}$. Pour tout $i \in \mathbb{N}$,*

$$\Big(t \in \mathcal{AN}_a \quad \wedge \quad t \to^*_{\beta \cup \mathcal{R}_i^{jo}} u\Big) \quad \Longrightarrow \quad \Big(u \in \mathcal{AN}_a \quad \wedge \quad \beta nf(t) \to^*_{\mathcal{R}_i^{jo}} \beta nf(u)\Big) \tag{7.3}$$

Preuve. On raisonne par induction sur $i \in \mathbb{N}$. Le cas de base $i = 0$ est trivial. Supposons que la propriété est vraie pour $i \geq 0$ et montrons-la pour $i + 1$.

On commence par montrer que pour tout $t \in \mathcal{AN}_a$, le diagramme suivant commute :



(7.4)

On raisonne par induction sur $\succ_h$ en utilisant le lemme 3.2.6.

$t = \lambda\vec{x}.x\ t_1 \dots t_n$. Dans ce cas, $\beta nf(t) = \lambda\vec{x}.x\ \beta nf(t_1) \dots \beta nf(t_n)$ et $u = \lambda\vec{x}.x\ u_1 \dots u_n$ avec $t_k \rhd_{\mathcal{R}_{i+1}^{jo}} u_k$ pour tout $k \in \{1, \dots, n\}$. Comme $t \succ_h t_k$, par hypothèse d'induction sur $\succ_h$, on a $u_k \in \mathcal{AN}_a$ et $\beta nf(t_k) \rhd_{\mathcal{R}_{i+1}^{jo}} \beta nf(u_k)$. Il s'en suit que u a pour forme β-normale $\lambda\vec{x}.x\ \beta nf(u_1) \dots \beta nf(u_n)$, d'où $u \in \mathcal{AN}_a$ et $\beta nf(t) \rhd_{\mathcal{R}_{i+1}^{jo}} \beta nf(u)$.

$t = f(t_1, \dots, t_n)\ t_{n+1} \dots t_{n+m}$. Si $u = f(u_1, \dots, u_n)\ u_{n+1} \dots u_{n+m}$ et $t_k \rhd_{\mathcal{R}_{i+1}^{jo}} u_k$ pour tout $k \in \{1, \dots, n+m\}$, alors on raisonne comme dans le cas précédent.

Sinon, il y a une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et deux substitutions σ et θ telles que

$$t = \lambda\vec{x}.l\sigma \cdot t_{n+k} \dots t_{n+m} \qquad \text{et} \qquad u = \lambda\vec{x}.r\theta \cdot u_{n+k} \dots u_{n+m}$$

avec $l\sigma \to_{\mathcal{R}_{i+1}^{jo}} r\sigma$, $\sigma \rhd_{\mathcal{R}_{i+1}^{jo}} \theta$ et $t_p \rhd_{\mathcal{R}_{i+1}^{jo}} u_p$ pour tout $p \in \{n+k, \dots, n+m\}$. On peut supposer que $\mathcal{D}om(\sigma) = FV(l)$.

Comme l est un terme algébrique qui n'est pas une variable, on a

$$\beta nf(t) \quad = \quad \lambda\vec{x}.l\sigma'\beta nf(t_{n+k}) \dots \beta nf(t_{n+m})$$

avec $\sigma' = \beta nf(\sigma)$. Comme $\beta nf(t)$ et $\mathcal{R}$ respectent a, on a $k = m$ et $\beta nf(t) = \lambda\vec{x}.l\sigma'$. Il s'en suit que $t = \lambda\vec{x}.l\sigma$ et que $u = \lambda\vec{x}.r\theta$.

Il reste à montrer que u a une forme β-normale, que cette forme β-normale respecte a et que $\beta nf(t) = \lambda\vec{x}.l\sigma' \rhd_{\mathcal{R}_{i+1}^{jo}} \beta nf(u)$. Comme l est algébrique, on a $l\sigma \succ_h \sigma(x)$ pour tout $x \in \mathcal{D}om(\sigma)$. Or, pour tout $x \in \mathcal{D}om(\sigma)$, $\sigma(x)$ a pour forme β-normale $\sigma'(x)$ qui respecte a. On peut donc appliquer l'hypothèse d'induction sur $\succ_h$ à

$\sigma \rhd_{\mathcal{R}^{jo}_{i+1}} \theta$, et il s'en suit que pour tout $x \in \mathcal{D}om(\sigma)$, $\theta(x)$ a une forme β-normale qui respecte a. On pose $\theta' =_{def} \beta nf(\theta)$. Par la proposition 6.1.12, $r\theta'$ respecte a, donc u a pour forme β-normale $\lambda\vec{x}.r\theta'$ et cette forme β-normale respecte a.
Pour conclure, il suffit donc de montrer que $l\sigma' \rightarrow_{\mathcal{R}^{jo}_{i+1}} r\sigma'$. c'est à dire de montrer que $\vec{d}\sigma' \downarrow_{\mathcal{R}^{jo}_{i}} \vec{c}\sigma'$. Comme $l\sigma \rightarrow_{\mathcal{R}^{jo}_{i+1}} r\sigma$, il existe des termes $\vec{v}$ tels que

$$\vec{d}\sigma \quad \rightarrow^*_{\mathcal{R}^{jo}_{i}} \quad \vec{v} \quad \leftarrow^*_{\mathcal{R}^{jo}_{i}} \quad \vec{c}\sigma\ .$$

Par hypothèse d'induction sur i, on a donc

$$\beta nf(\vec{d}\sigma) \quad \rightarrow^*_{\mathcal{R}^{jo}_{i}} \quad \beta nf(\vec{v}) \quad \leftarrow^*_{\mathcal{R}^{jo}_{i}} \quad \beta nf(\vec{c}\sigma)$$

c'est-à-dire $\vec{d}\sigma' \downarrow_{\mathcal{R}^{jo}_{i}} \vec{c}\sigma'$, car $\vec{d}$ et $\vec{c}$ sont faits de termes algébriques.

$t = \lambda\vec{x}.(\lambda x.\nu)wt_1 \dots t_n$. Dans ce cas, on normalise t en tête et on obtient un terme $t' \in \mathcal{AN}_a$ de la forme (b) ou (c). En utilisant la commutation en un pas de $\rightarrow_h$ et de $\rhd_{\mathcal{R}^{jo}_{i+1}}$ (lemme 7.1.14), on obtient un terme u' tel que $t' \rhd_{\mathcal{R}^{jo}_{i+1}} u'$. Comme $t \succ^+_h t'$ on peut raisonner à partir de t' comme dans les deux cas précédents.

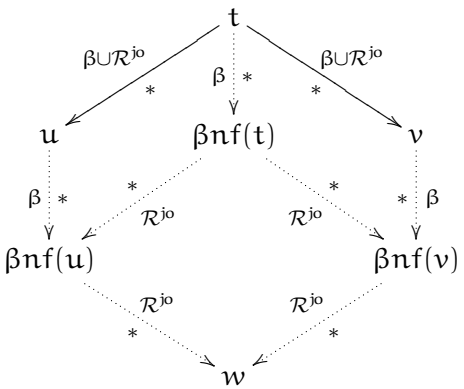
Pour en déduire (7.3), on raisonne par induction sur $t \rightarrow^*_{\beta\cup\mathcal{R}^{jo}_{i+1}} u$, en utilisant la proposition 7.1.13.(i). Le cas de base est trivial car $t = u$. Dans le cas d'induction, il existe ν tel que $t \rightarrow_{\beta\cup\mathcal{R}^{jo}_{i+1}} \nu \rightarrow^*_{\beta\cup\mathcal{R}^{jo}_{i+1}} u$. Par hypothèse d'induction, on a $\beta nf(\nu) \rightarrow^*_{\mathcal{R}^{jo}_{i+1}} \beta nf(u)$, et il y a deux cas :

— si $t \rightarrow_{\mathcal{R}^{jo}_{i+1}} \nu$, alors on a $t \rhd_{\mathcal{R}^{jo}_{i+1}} \nu$, d'où $\beta nf(t) \rightarrow^*_{\mathcal{R}^{jo}_{i+1}} \beta nf(\nu)$ par (7.4),
— sinon $t \rightarrow_\beta \nu$ donc $\beta nf(t) = \beta nf(\nu)$. □

La préservation de la confluence est une conséquence directe de la projection de la réécriture sur les formes β-normales.

Théorème 7.1.16 *Soit $\mathcal{R}$ un système conditionnel algébrique qui respecte $a : \Sigma \to \mathbb{N}$. Si $\rightarrow_{\mathcal{R}^{jo}}$ est confluent sur $\mathcal{AN}_a$ alors $\rightarrow_{\beta\cup\mathcal{R}^{jo}}$ est confluent sur $\mathcal{AN}_a$.*

Preuve. Soient t, u, ν tels que $t \in \mathcal{AN}_a$ et $u \leftarrow^*_{\beta\cup\mathcal{R}^{jo}} t \rightarrow^*_{\beta\cup\mathcal{R}^{jo}} \nu$. En appliquant deux fois le lemme 7.1.15, on obtient $\beta nf(u) \leftarrow^*_{\mathcal{R}^{jo}} \beta nf(t) \rightarrow^*_{\mathcal{R}^{jo}} \beta nf(\nu)$ avec $\beta nf(t), \beta nf(u)$ et $\beta nf(\nu) \in \mathcal{AN}_a$. On conclut par la confluence de $\rightarrow_{\mathcal{R}^{jo}}$ sur $\mathcal{AN}_a$. En image :

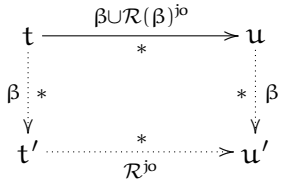


□

7.2 Confluence avec béta-réduction dans les conditions

Dans cette section, nous nous concentrons sur la relation de réécriture β-conditionnelle par joignabilité $\to_{\mathcal{R}(\beta)^{\mathrm{jo}}}$. Cette relation diffère de $\to_{\mathcal{R}^{\mathrm{jo}}}$ par le fait qu'elle autorise des pas de β-réduction dans l'évaluation des conditions.

Une première possibilité est de partir des résultats de la section 7.1 et d'étudier des conditions suffisantes pour que la confluence de $\to_{\mathcal{R}^{\mathrm{jo}}}$ entraîne la confluence de $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$. Notre stratégie est de projeter, par β-réduction, les pas de $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$ sur des pas de $\to_{\mathcal{R}^{\mathrm{jo}}}$, c'est-à-dire



(7.5)

La propriété (7.5) est en général fausse lorsque les conditions ou les membres droits des règles contiennent des variables actives ou bien des termes ne respectant pas l'arité des symboles. De plus, la relation $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$ peut ne pas être confluente alors que la relation $\to_{\beta\cup\mathcal{R}^{\mathrm{jo}}}$ l'est.

Exemple 7.2.1 *Pour chacun des systèmes conditionnels non-algébriques (7.6), (7.7), (7.8) et (7.9) présentés ci-dessous,*

(i) la relation $\to_{\beta\cup\mathcal{R}^{\mathrm{jo}}}$ est confluente,

(ii) la propriété (7.5) n'est pas vérifiée et la relation $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$ n'est pas confluente.

$$\mathsf{g\,x\,y} \mapsto \mathsf{x\,y} \qquad \mathsf{g\,x\,c} = \mathsf{d} \supset \mathsf{f\,x} \mapsto \mathsf{a\,x} \qquad \mathsf{f\,x} \mapsto \mathsf{b\,x} \tag{7.6}$$

$$\mathsf{x\,c} = \mathsf{d} \supset \mathsf{f\,x} \mapsto \mathsf{a\,x} \qquad \mathsf{f\,x} \mapsto \mathsf{b\,x} \tag{7.7}$$

$$\mathsf{id\,x\,c} = \mathsf{d} \supset \mathsf{f\,x} \mapsto \mathsf{a\,x} \qquad \mathsf{f\,x} \mapsto \mathsf{b\,x} \tag{7.8}$$

$$\mathsf{h\,x\,y} \mapsto \mathsf{id\,x\,y} \qquad \mathsf{h\,x\,c} = \mathsf{d} \supset \mathsf{f\,x} \mapsto \mathsf{a\,x} \qquad \mathsf{f\,x} \mapsto \mathsf{b\,x} \tag{7.9}$$

où le symbole id *est défini par* $\mathsf{id\,x} \mapsto \mathsf{x}$.

Preuve.

(i) Dans aucun de ces systèmes le symbole d n'est défini. Ce sont donc des systèmes normaux auxquels on peut appliquer le théorème 7.1.6. Ainsi la confluence de $\to_{\beta\cup\mathcal{R}^{\mathrm{jo}}}$ sur $\Lambda(\Sigma)$ suit de la confluence de $\to_{\mathcal{R}^{\mathrm{jo}}}$ sur $\Lambda(\Sigma)$.

Comme ce sont des systèmes linéaires à gauche, par le théorème 5.2.6.(ii) on a la confluence de $\to_{\mathcal{R}^{\mathrm{jo}}}$ sur $\mathcal{T}er(\Sigma_{\mathrm{App}},\mathcal{X})$ si leurs paires critiques sont infaisables. Chaque système contient une unique règle conditionnelle et une unique paire critique, qui est la superposition de cette règle avec la règle $\mathsf{fx} \mapsto \mathsf{bx}$.

Dans chaque cas, le nombre d'occurrence du symbole c dans un terme est préservé par $\to_{\mathcal{R}^{\mathrm{jo}}}$-réduction. Or, pour chaque instantiation de la règle conditionnelle, le

membre gauche instantié de la condition contient au moins une occurrence du symbole c. Il ne peut donc se $\to_{\mathcal{R}^{\mathrm{jo}}}$-réduire vers le terme d. La paire critique est donc infaisable et le système est confluent sur $\mathcal{T}er(\Sigma_{\mathcal{A}pp}, \mathcal{X})$. Par le théorème 5.3.4, on a la confluence de $\to_{\mathcal{R}^{\mathrm{jo}}}$ sur $\Lambda(\Sigma)$, d'où la confluence de $\to_{\beta\cup\mathcal{R}^{\mathrm{jo}}}$ sur $\Lambda(\Sigma)$.

(ii) Dans chaque cas, le pas $\mathsf{f}\ \lambda\mathsf{x.d} \to_{\mathcal{R}(\beta)^{\mathrm{jo}}} \mathsf{a}\ \lambda\mathsf{x.d}$ n'est pas dans $\to_\beta^* \to_{\mathcal{R}^{\mathrm{jo}}}^* \leftarrow_\beta^*$ et le pic suivant est injoignable :

$$\mathsf{a}\ \lambda\mathsf{x.d} \quad \leftarrow_{\mathcal{R}(\beta)^{\mathrm{jo}}} \quad \mathsf{f}\ \lambda\mathsf{x.d} \quad \to_{\mathcal{R}(\beta)^{\mathrm{jo}}} \quad \mathsf{b}\ \lambda\mathsf{x.d}\ .$$

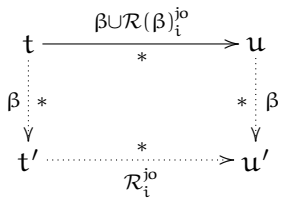
□

Notons que les systèmes (7.6) et (7.7) contiennent respectivement un membre droit et une condition non-algébrique et que les systèmes (7.8) et (7.9) contiennent respectivement un membre droit et une condition qui ne respecte pas l'arité du symbole id (voir l'exemple 6.1.14).

Nous commençons à la section 7.2.1 par l'extension du théorème 7.1.4 à la relation $\to_{\mathcal{R}(\beta)^{\mathrm{jo}}}$. Comme montré par l'exemple précédant, il faut travailler avec des termes vérifiant des conditions d'arité. Pour cela, nous utilisons une version faible de la $(\mathcal{R}, \mathfrak{a})$-stabilité qui ne se restreint pas aux termes fortement β-normalisants. Ensuite, à la section 7.2.2 nous passons au cas du théorème 7.1.16, qui s'étend directement à $\to_{\mathcal{R}(\beta)^{\mathrm{jo}}}$.

Enfin, à la section 7.2.3, nous donnons une condition permettant d'assurer la confluence de $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$ lorsque les conditions et les membres droits des règles peuvent contenir des abstractions et des variables actives. Comme nous l'avons vu avec l'exemple 7.2.1, on ne peut, dans ce cas, déduire la confluence de $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$ de la confluence de $\to_{\mathcal{R}^{\mathrm{jo}}}$ ni même de $\to_{\beta\cup\mathcal{R}^{\mathrm{jo}}}$. Nous introduisons une classe de systèmes, dits « orthonormaux » pour laquelle on peut prouver la confluence de $\to_{\beta\cup\mathcal{R}(\beta)^{\mathrm{jo}}}$ de façon directe.

Dans [BKR06], nous avons montré (7.5) en utilisant une stratification de $\to_{\mathcal{R}(\beta)^{\mathrm{jo}}}$ dans laquelle, au lieu d'avoir $\to_{\mathcal{R}(\beta)_0^{\mathrm{jo}}} = \emptyset$ comme prescrit dans la définition 4.1.5, on avait $\to_{\mathcal{R}(\beta)_0^{\mathrm{jo}}} = \to_{\mathcal{R}^{\mathrm{jo}}}$ (il est aisé de voir que ces deux cas de base induisent la même relation $\to_{\mathcal{R}(\beta)^{\mathrm{jo}}}$). Ici, on procède de manière légèrement différente. On prouve (7.5) en utilisant $\to_{\mathcal{R}(\beta)_0^{\mathrm{jo}}} = \emptyset$ et en passant par la propriété intermédiaire suivante : pour tout $\mathfrak{i} \in \mathbb{N}$,



(7.10)

7.2.1 Systèmes linéaires à gauche et semi-clos

Cette section est consacrée à la preuve de (7.10) pour les systèmes linéaires à gauche et semi-clos. On commence par quelques propriétés techniques. La première est une généralisation de la propriété du diamant de $\rhd_\beta$. C'est une conséquence directe de la propriété (5) de [Tak95].

Proposition 7.2.2 *Soit* $n \geq 0$ *et supposons que* $t, t_1, \ldots, t_n$ *soient des termes tels que* $t \rhd_\beta t_i$ *pour tout* $i \in \{1, \ldots, n\}$. *Il existe un terme* t' *tel que* $t \rhd_\beta t'$ *et* $t_i \rhd_\beta t'$ *pour tout* $i \in \{1, \ldots, n\}$.

On en déduit la proposition 7.2.3, qui est importante pour les termes algébriques. La preuve du cas (ii) utilise la propriété du diamant de $\rhd_\beta$.

Proposition 7.2.3 *Soient* $t_1, \ldots, t_n$ *des termes algébriques et* σ *une substitution.*

(i) Si $t_i\sigma \rhd_\beta u_i$ *pour tout* $i \in \{1, \ldots, n\}$, *alors il existe une substitution* σ' *telle que* $\sigma \rhd_\beta \sigma'$ *et* $u_i \rhd_\beta t_i\sigma'$ *pour tout* $i \in \{1, \ldots, n\}$.

(ii) Si $t_i\sigma \to_\beta^* u_i$ *pour tout* $i \in \{1, \ldots, n\}$, *alors il existe une substitution* σ' *telle que* $\sigma \to_\beta^* \sigma'$ *et* $u_i \to_\beta^* t_i\sigma'$ *pour tout* $i \in \{1, \ldots, n\}$.

Notons que les termes $t_1, \ldots, t_n$ peuvent ne pas être linéaires.

Preuve.

(i) Comme t_i est algébrique, toute occurrence d'un β-rédex dans $t_i\sigma$ est de la forme $p.d$ où p est l'occurrence d'une variable x dans t_i. Comme $\rhd_\beta$ est réflexive, il existe donc des termes

$$(s_{i,x,p})_{i\in\{1,\ldots,n\},\ x\in FV(t_i),\ p\in\mathcal{O}cc(x,t_i)}$$

tels que pour tout $i \in \{1, \ldots, n\}$, $x \in FV(t_i)$ et $p \in \mathcal{O}cc(x, t_i)$ on ait $t_i\sigma|_p = \sigma(x) \rhd_\beta s_{i,x,p}$ et pour tout $i \in \{1, \ldots, n\}$

$$u_i \quad = \quad t_i[p \leftarrow s_{i,x,p} \mid x \in FV(t_i) \quad \wedge \quad p \in \mathcal{O}cc(x, t_i)] \ .$$

Par la proposition 7.2.2, pour tout $x \in FV(t_1, \ldots, t_n)$, il existe un terme ν_x tel que $\sigma(x) \rhd_\beta \nu_x$ et $s_{i,x,p} \rhd_\beta \nu_x$ pour tout $i \in \{1, \ldots, n\}$ et tout $p \in \mathcal{O}cc(x, t_i)$. On a donc pour tout $i \in \{1, \ldots, n\}$,

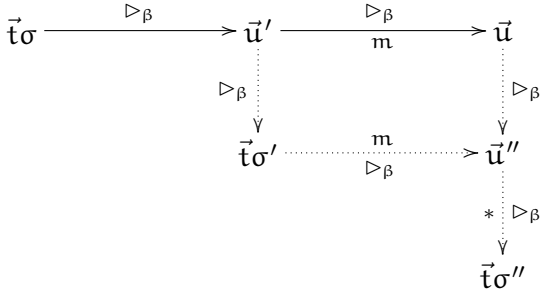
$$u_i \quad \rhd_\beta \quad t_i[p \leftarrow \nu_x \mid x \in FV(t_i) \quad \wedge \quad p \in \mathcal{O}cc(x, t_i)] \ .$$

Si σ' est la substitution de même domaine que σ telle que $\sigma'(x) = \nu_x$ pour tout $x \in FV(t_1, \ldots, t_n)$ et $\sigma'(x) = \sigma(x)$ pour tout $x \notin FV(t_1, \ldots, t_n)$, alors on a $\sigma \rhd_\beta \sigma'$ et $u_i \rhd_\beta t_i\sigma'$ pour tout $i \in \{1, \ldots, n\}$.

(ii) Par induction sur $m \in \mathbb{N}$, on montre que si $t_i\sigma \rhd^m u_i$ pour tout $i \in \{1, \ldots, n\}$, alors il existe σ' telle que $\sigma \rhd_\beta^* \sigma'$ et $u_i \rhd^* t_i\sigma'$ pour tout $i \in \{1, \ldots, n\}$.

Le cas de base $t_i\sigma \rhd^0 u_i$ pour tout $i \in \{1, \ldots, n\}$ est trivial. Dans le cas d'induction, il existe $u'_1, \ldots, u'_n$ tels que $t_i\sigma \rhd_\beta^m u'_i \rhd_\beta u_i$ pour tout $i \in \{1, \ldots, n\}$. Alors, par (i), il existe σ' telle que $\sigma \rhd_\beta \sigma'$ et $u'_i \rhd_\beta t_i\sigma'$ pour tout $i \in \{1, \ldots, n\}$. Comme $\rhd_\beta$ satisfait la propriété du diamant (lemme 5.1.11), pour tout $i \in \{1, \ldots, n\}$ il existe u''_i tel que $t_i\sigma' \rhd_\beta^m u''_i \lhd_\beta u_i$, et par hypothèse d'induction sur m, il existe

σ'' telle que $\sigma' \rhd_\beta^* \sigma''$ et $u_i'' \rhd_\beta^* t_i\sigma''$ pour tout $i \in \{1,\dots,n\}$. On en déduit que $u_i \rhd_\beta^* t_i\sigma''$ pour tout $i \in \{1,\dots,n\}$. En image :



□

On raisonne sur des termes vérifiant une arité compatible avec leur système de réécriture. On a besoin que cette propriété soit préservée par $\to_{\beta\mathcal{R}}$-réduction, mais aussi par les conditions des règles : étant donnée une règle semi-close $\vec{d} = \vec{c} \supset l \mapsto_\mathcal{R} r$ qui respecte $a : \Sigma \to \mathbb{N}$ et une substitution σ telle que $l\sigma$ respecte a, alors les termes de $r\sigma, \vec{d}\sigma$ doivent aussi respecter a. C'est bien entendu le cas s'il sont algébriques et qu'ils respectent a (proposition 6.1.12).

Proposition 7.2.4 *Soit $\mathcal{R}$ un système conditionnel algébrique et $t \in \Lambda(\Sigma)$ qui respectent tout deux $a : \Sigma \to \mathbb{N}$. Si $t \to_{\mathcal{R}^{jo}} u$ alors u respecte a.*

Preuve. Par induction sur t, en utilisant le lemme 3.2.6.

Le seul cas qui ne suit pas directement de l'hypothèse d'induction est celui où $t = \lambda\vec{x}.f(t_1,\dots,t_n)t_{n+1}\dots t_{n+m}$ et il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_\mathcal{R} r$, une substitution σ et $k \in \{1,\dots,m\}$ tels que $t = \lambda\vec{x}.l\sigma t_{n+k}\dots t_{n+m}$. Comme t et $\mathcal{R}$ respectent a, on a $k = m$. Donc $u = \lambda\vec{x}.r\sigma$, et u respecte a par la proposition 6.1.12 car r est un terme algébrique qui respecte a. □

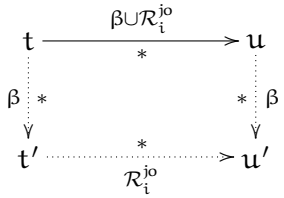
La seule façon de passer d'un terme respectant a à un terme qui ne respecte pas a est donc la β-réduction. Par exemple, avec $a(\mathsf{id}) = 1$, $(\lambda x.x\ y\ y)\mathsf{id}$ respecte a alors que $\mathsf{id}\ y\ y$ ne respecte pas a.

Définition 7.2.5 *Soit $\mathcal{R}$ un système conditionnel et considérons le TRS*

$$\to_{Cond_\mathcal{R}} \quad =_{def} \quad \{(l, d_i) \mid \vec{d} = \vec{c} \supset l \mapsto_\mathcal{R} r \quad \wedge \quad i \in \{1,\dots,|\vec{d}|\}\}\ .$$

On dit que t respecte $\mathcal{R}$-héréditairement a si u respecte a pour tout $u \in (t)^*_{\to_{\beta\cup\mathcal{R}^{jo}\cup Cond_\mathcal{R}}}$.

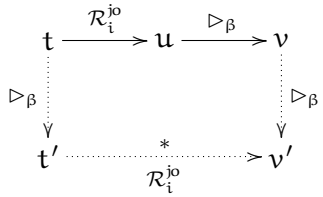
Nous nous embarquons maintenant dans la preuve de (7.10). Notre stratégie est d'utiliser une propriété similaire de $\to_{\mathcal{R}^{jo}}$: pour tout $i \in \mathbb{N}$



(7.11)

Cette propriété occupe la proposition 7.2.6 et le lemme 7.2.7. Notons que nous la prouvons pour des systèmes dont les conditions ne sont pas nécessairement algébriques. Par contre, lorsque les membres droits des règles ne sont pas algébriques ou ne respectent pas l'arité des membres gauches, alors la réécriture peut créer des β-rédexes, et dans ce cas, la propriété (7.11) peut ne pas être satisfaite. D'autre part, nous supposons que tous les termes considérés respectent l'arité du système de réécriture.

Proposition 7.2.6 *Soit $\mathcal{R}$ un système semi-clos, linéaire à gauche et algébrique à droite respectant $a : \Sigma \to \mathbb{N}$. Pour tout $i \in \mathbb{N}$ et tout $t, u, v \in \Lambda(\Sigma)$, si t respecte $\mathcal{R}$-héréditairement a et $t \to_{\mathcal{R}_i^{jo}} u \rhd_\beta v$, alors il existe t' et v' tels que $t \rhd_\beta t' \to^*_{\mathcal{R}_i^{jo}} v' \lhd_\beta v$. En image :*



Preuve. Le cas $i = 0$ est trivial, on peut donc supposer $i > 0$. On raisonne par induction sur t, en utilisant le lemme 3.2.6.

$t = \lambda\vec{x}.x\ t_1 \dots t_n$. Dans ce cas, $u = \lambda x.\vec{x}.x u_1 \dots u_n$ avec $(t_1, \dots, t_n) \to_{\mathcal{R}_i^{jo}} (u_1, \dots, u_n)$. De plus, $v = \lambda\vec{x}.x\ v_1 \dots v_n$ avec $(u_1, \dots, u_n) \rhd_\beta (v_1, \dots, v_n)$. Par hypothèse d'induction, il existe $(t'_1, \dots, t'_n)$ et $(v'_1, \dots, v'_n)$ tels que

$$(t_1, \dots, t_n) \quad \rhd_\beta \quad (t'_1, \dots, t'_n) \quad \to^*_{\mathcal{R}_i^{jo}} \quad (v'_1, \dots, v'_n) \quad \lhd_\beta \quad (v_1, \dots, v_n)\ .$$

On a donc

$$\lambda\vec{x}.x\ t_1 \dots t_n \quad \rhd_\beta \quad \lambda\vec{x}.x\ t'_1 \dots t'_n \quad \to^*_{\mathcal{R}_i^{jo}} \quad \lambda\vec{x}.x\ v'_1 \dots v'_n \quad \lhd_\beta \quad \lambda\vec{x}.x\ v_1 \dots v_n\ .$$

$t = \lambda\vec{x}.f(t_1, \dots, t_n)t_{n+1} \dots t_{n+m}$. Si on a $u = \lambda\vec{x}.f(u_1, \dots, u_n)u_{n+1} \dots u_{n+m}$ avec

$$(t_1, \dots, t_{n+m}) \quad \to_{\mathcal{R}_i^{jo}} \quad (u_1, \dots, u_{n+m})\ ,$$

alors $v = \lambda\vec{x}.f(v_1, \dots, v_n)v_{n+1} \dots v_{n+m}$ avec $(u_1, \dots, u_{n+m}) \rhd_\beta (v_1, \dots, v_{n+m})$ et on raisonne comme dans le cas précédent.

Sinon, il y une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, une substitution σ et $k \in \{1, \dots, m\}$ tels que $t = \lambda\vec{x}.l\sigma t_{n+k} \dots t_{n+m}$ et $u = \lambda\vec{x}.r\sigma t_{n+k} \dots t_{n+m}$. Comme t et $\mathcal{R}$ respectent a, on a $m = k$, d'où $t = \lambda\vec{x}.l\sigma$, $u = \lambda\vec{x}.r\sigma$ et $v = \lambda\vec{x}.w$ avec $r\sigma \rhd_\beta w$.

Le terme r étant algébrique, la proposition 7.2.3.(i) implique qu'il existe σ' telle que $\sigma \rhd_\beta \sigma'$ et $w \rhd_\beta r\sigma'$. Comme on a $l\sigma \rhd_\beta l\sigma'$, il reste à montrer que $l\sigma' \to_{\mathcal{R}_i^{jo}} r\sigma'$.

Pour cela, il suffit de raisonner exactement comme dans la preuve du lemme 7.1.3.

$t = \lambda\vec{x}.(\lambda x.t_0)t_1 \dots t_n$ $(n \geq 1)$. Alors le terme u est de la forme $u = \lambda\vec{x}.(\lambda x.u_0)u_1 \dots u_n$ et $(t_0, \dots, t_n) \to_{\mathcal{R}_i^{jo}} (u_0, \dots, u_n)$. Si $v = \lambda\vec{x}.(\lambda x.v_0)v_1 \dots v_n$ avec

$$(u_0, \dots, u_n) \quad \rhd_\beta \quad (v_0, \dots, v_n)\ ,$$

alors on conclut par hypothèse d'induction, comme dans le premier cas.
Sinon, $\nu = \lambda\vec{x}.\nu_0[\nu_1/x]\nu_2\dots\nu_n$ avec $(u_0,\dots,u_n) \rhd_\beta (\nu_0,\dots,\nu_n)$. Par hypothèse d'induction, on a

$$(t_0,\dots,t_n) \quad \rhd_\beta \quad (t'_0,\dots,t'_n) \quad \to^*_{\mathcal{R}_i^{jo}} \quad (\nu'_0,\dots,\nu'_n) \quad \lhd_\beta \quad (\nu_0,\dots,\nu_n)\ .$$

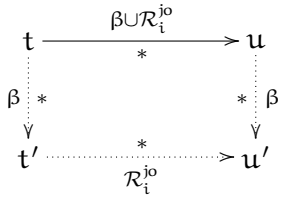
Il s'en suit par $(\rhd\beta)$, $(\rhd\textsc{App})$ et par le lemme 2.3.1 que

$$\begin{array}{ccc} \lambda\vec{x}.(\lambda x.t_0)t_1\dots t_n & & \lambda\vec{x}.\nu_0[\nu_1/x]\nu_2\dots\nu_n \\ \triangledown_\beta & & \triangledown_\beta \\ \lambda\vec{x}.t'_0[t'_1/x]t'_2\dots t'_n & \to^*_{\mathcal{R}_i^{jo}} & \lambda\vec{x}.\nu'_0[\nu'_1/x]\nu'_2\dots\nu'_n\ . \end{array}$$

□

On en déduit (7.11).

Lemme 7.2.7 *Soit $\mathcal{R}$ un système semi-clos, linéaire à gauche et algébrique à droite respectant $a : \Sigma \to \mathbb{N}$. Pour tout $i \in \mathbb{N}$ et tout $t, u \in \Lambda(\Sigma)$, si t respecte $\mathcal{R}$-héréditairement a et si $t \to^*_{\beta\cup\mathcal{R}_i^{jo}} u$, alors il existe t' et u' tels que $t \to^*_\beta t' \to^*_{\mathcal{R}_i^{jo}} u' \leftarrow^*_\beta u$. En image :*



Preuve. La preuve se déroule en trois étapes.

(i) $\to^*_{\mathcal{R}_i^{jo}} \rhd_\beta \subseteq \rhd_\beta \to^*_{\mathcal{R}_i^{jo}} \lhd^*_\beta$. On raisonne par induction sur $\to^*_{\mathcal{R}_i^{jo}}$. Si

$$t \quad \to^*_{\mathcal{R}_i^{jo}} \quad u' \quad \to_{\mathcal{R}_i^{jo}} \quad u \quad \rhd_\beta \quad \nu\ ,$$

par la proposition 7.2.6, il existe w et ν' tels que $u' \rhd_\beta w \to^*_{\mathcal{R}_i^{jo}} \nu' \lhd_\beta \nu$. Par hypothèse d'induction, on en déduit qu'il existe t' et t'' tels que $t \rhd_\beta t' \to^*_{\mathcal{R}_i^{jo}} t'' \lhd^*_\beta w$. Le lemme 7.1.3 implique qu'il existe ν'' tel que $t'' \to^*_{\mathcal{R}_i^{jo}} \nu'' \lhd^*_\beta \nu'$, d'où $t \rhd_\beta \to^*_{\mathcal{R}_i^{jo}} \lhd^*_\beta \nu$.

(ii) $\to^*_{\mathcal{R}_i^{jo}} \rhd^*_\beta \subseteq \rhd^*_\beta \to^*_{\mathcal{R}_i^{jo}} \lhd^*_\beta$. On raisonne par induction sur le nombre de pas de $\rhd_\beta$. Si $t \to^*_{\mathcal{R}_i^{jo}} u' \rhd_\beta u \rhd^n_\beta \nu$, alors par (i) il existe t' et u'' tels que

$$t \quad \rhd_\beta \quad t' \quad \to^*_{\mathcal{R}_i^{jo}} \quad u'' \quad \rhd^*_\beta \quad u\ .$$

Comme $\rhd_\beta$ vérifie la propriété du diamant (lemme 5.1.11), il existe ν' tel que $u'' \rhd^n_\beta \nu' \lhd^*_\beta \nu$. Par hypothèse d'induction sur n, il existe t'' et ν'' tels que

$$t' \quad \rhd^*_\beta \quad t'' \quad \to^*_{\mathcal{R}_i^{jo}} \quad \nu'' \quad \lhd^*_\beta \quad \nu'\ .$$

On a donc $t \rhd^*_\beta \to^*_{\mathcal{R}_i^{jo}} \lhd^*_\beta \nu$.

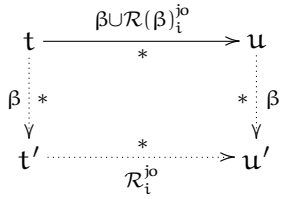
(iii) $(\to_{\mathcal{R}_i^{jo}} \cup \rhd_\beta)^* \subseteq \rhd_\beta^* \to_{\mathcal{R}_i^{jo}}^* \lhd_\beta^*$. On raisonne par induction sur $(\to_{\mathcal{R}_i^{jo}} \cup \rhd_\beta)^*$. Si

$$t \quad (\to_{\mathcal{R}_i^{jo}} \cup \rhd_\beta) \quad u \quad (\to_{\mathcal{R}_i^{jo}} \cup \rhd_\beta)^* \quad v\ ,$$

alors par hypothèse d'induction il existe u' et v' tels que $u \rhd_\beta^* u' \to_{\mathcal{R}_i^{jo}}^* v' \lhd_\beta^* v$. Il y a alors deux cas. Si $t \rhd_\beta u$ alors on a $t \rhd_\beta^* \to_{\mathcal{R}_i^{jo}}^* \lhd_\beta^* v$. Sinon, par (ii), il existe t' et u'' tels que $t \rhd_\beta^* t' \to_{\mathcal{R}_i^{jo}}^* u'' \lhd_\beta^* u'$. Par le lemme 7.1.3, on en déduit qu'il existe v'' tel que $u'' \to_{\mathcal{R}_i^{jo}}^* v'' \lhd_\beta^* v'$. On a donc $t \rhd_\beta^* \to_{\mathcal{R}_i^{jo}}^* v'' \lhd_\beta^* v$. □

Voici la preuve de (7.10).

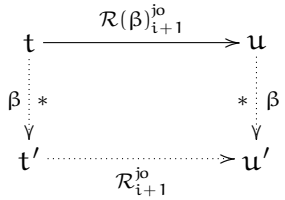
Lemme 7.2.8 *Soit $\mathcal{R}$ un système semi-clos, linéaire à gauche et algébrique respectant $a : \Sigma \to \mathbb{N}$. Pour tout $i \in \mathbb{N}$ et tout $t, u \in \Lambda(\Sigma)$, si t respecte $\mathcal{R}$-héréditairement a et si $t \to_{\beta \cup \mathcal{R}(\beta)_i^{jo}}^* u$, alors il existe t' et u' tels que $t \to_\beta^* t' \to_{\mathcal{R}_i^{jo}}^* u' \leftarrow_\beta^* u$. En image :*



(7.12)

Preuve. On montre (7.12) par induction sur $i \in \mathbb{N}$. Le cas de base $i = 0$ est trivial. Supposons que la propriété soit satisfaite pour $i \geq 0$ et montrons la pour $i + 1$.

On commence par montrer que le diagramme (7.13) commute :



(7.13)

On raisonne par induction sur t, en utilisant le lemme 3.2.6. Le seul cas qui ne suit pas directement de l'hypothèse d'induction est celui où $t = \lambda\vec{x}.f(t_1, \ldots, t_n)t_{n+1} \ldots t_{n+m}$ et il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, une substitution σ et $k \in \{1, \ldots, m\}$ tels que $t = \lambda\vec{x}.l\sigma t_{n+k} \ldots t_{n+m}$ et $u = \lambda\vec{x}.r\sigma u_{n+k} \ldots u_{n+m}$ avec $l\sigma \to_{\mathcal{R}(\beta)_{i+1}^{jo}} r\sigma$. Comme t et $\mathcal{R}$ respectent a, on a $k = m$, donc $u = \lambda\vec{x}.r\sigma$.

On en déduit (7.13) s'il existe une substitution σ' telle que

$$l\sigma \quad \to_\beta^* \quad l\sigma' \quad \to_{\mathcal{R}_{i+1}^{jo}} \quad r\sigma' \quad \leftarrow_\beta^* \quad r\sigma\ .$$

Comme $l\sigma \to_{\mathcal{R}(\beta)_{i+1}^{jo}} r\sigma$, il existe des termes $\vec{v}$ tels que $\vec{d}\sigma \to_{\beta \cup \mathcal{R}(\beta)_i^{jo}}^* \vec{v} \leftarrow_{\beta \cup \mathcal{R}(\beta)_i^{jo}}^* \vec{c}$. Par hypothèse d'induction sur i, il existe des termes $\vec{w}$ et $\vec{v}'$ tels que

$$\vec{d}\sigma \quad \to_\beta^* \quad \vec{w} \quad \to_{\mathcal{R}_i^{jo}}^* \quad \vec{v}' \quad \leftarrow^* \beta \quad \vec{v}\ .$$

Comme les termes $\vec{d}$ sont algébriques, par la proposition 7.2.3.(ii) il existe une substitution σ' telle que $\sigma \to_\beta^* \sigma'$ et $\vec{w} \to_\beta^* \vec{d}\sigma'$. En appliquant la commutation de $\to_{\mathcal{R}_i^{jo}}$ avec $\to_\beta$ (lemme 7.1.3), on obtient des termes $\vec{v}'$ tels que $\vec{d}\sigma' \to_{\mathcal{R}_i^{jo}}^* \vec{v}' \leftarrow_\beta^* \vec{v}$. On a donc

$$\vec{d}\sigma \quad \to_\beta^* \quad \vec{d}\sigma' \quad \to_{\mathcal{R}_i^{jo}}^* \quad \vec{v}' \quad \leftarrow_\beta^* \quad \vec{v} \quad \leftarrow_{\beta\cup\mathcal{R}(\beta)_i^{jo}}^* \quad \vec{c}\,.$$

Maintenant, comme les termes $\vec{c}$ sont algébriques et comme la relation $\to_{\mathcal{R}(\beta)_i^{jo}}$ est issue du système applicatif $\{(l, r) \mid \vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r\}$, par la proposition 3.1.10 tous les réduits des termes $\vec{c}$ par $\to_{\mathcal{R}(\beta)^{jo}}$ sont β-normaux. On a donc $\vec{v}' = \vec{v}$ et par hypothèse d'induction sur i, $\vec{c} \to_{\mathcal{R}_i^{jo}}^* \vec{v}$. Il s'en suit que $\vec{d}\sigma' \downarrow_{\mathcal{R}_i^{jo}} \vec{c}$, donc que $l\sigma' \to_{\mathcal{R}_{i+1}^{jo}} r\sigma'$. Comme $\sigma \to_\beta^* \sigma'$, on a $l\sigma \to_\beta^* l\sigma'$ et $r\sigma \to_\beta^* r\sigma'$, d'où

$$t \quad \to_\beta^* \quad \lambda\vec{x}.l\sigma' \quad \to_{\mathcal{R}_{i+1}^{jo}} \quad \lambda\vec{x}.r\sigma \quad \leftarrow_\beta^* \quad u\,.$$

Ceci conclut la preuve de (7.13).

On montre maintenant (7.12) en raisonnant par induction sur $t \to_{\beta\cup\mathcal{R}(\beta)_{i+1}^{jo}}^* u$. Le case de base $t = u$ est trivial. Dans le cas d'induction, il existe v tel que $t \to_{\beta\cup\mathcal{R}(\beta)_{i+1}^{jo}} v \to_{\beta\cup\mathcal{R}(\beta)_{i+1}^{jo}}^*$ et par hypothèse d'induction, il existe v' et u' tels que

$$v \quad \to_\beta^* \quad v' \quad \to_{\mathcal{R}_{i+1}^{jo}}^* \quad u' \quad \leftarrow_\beta^* \quad u\,.$$

Si $t \to_\beta v$ alors on a

$$t \quad \to_\beta^* \quad v' \quad \to_{\mathcal{R}_{i+1}^{jo}}^* \quad u' \quad \leftarrow_\beta^* \quad u\,.$$

Sinon, par (7.13), il existe t' et v'' tels que

$$t \quad \to_\beta^* \quad t' \quad \to_{\mathcal{R}_{i+1}^{jo}} \quad v'' \quad \leftarrow_\beta^* \quad v\,.$$

La confluence de $\to_\beta$ implique qu'il existe v''' tel que $v'' \to_\beta^* v''' \leftarrow^* v$, et par la commutation de $\to_{\mathcal{R}_{i+1}^{jo}}$ avec $\to_\beta$ (lemme 7.1.3), il existe u'' tel que

$$v''' \quad \to_{\mathcal{R}_{i+1}^{jo}}^* \quad u'' \quad \leftarrow_\beta^* \quad u'\,.$$

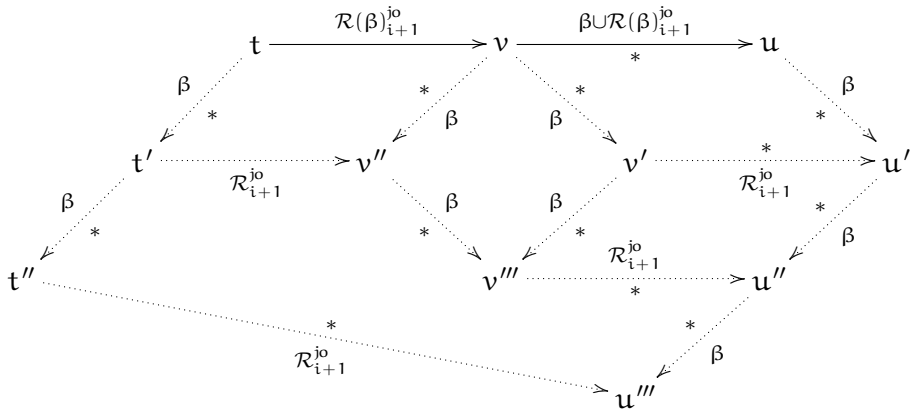
On a donc

$$t \quad \to_\beta^* \quad t' \quad \to_{\beta\cup\mathcal{R}_{i+1}^{jo}}^* \quad u'' \quad \leftarrow_\beta^* \quad u\,,$$

et par le lemme 7.2.7, on en déduit qu'il existe t'' et u''' tels que

$$t' \quad \to_\beta^* \quad t'' \quad \to_{\mathcal{R}_{i+1}^{jo}}^* \quad u''' \quad \leftarrow_\beta^* \quad u''\,,$$

d'où (7.12). En image :



□

On en déduit facilement (7.5), donc, par le théorème 7.1.4 que $\to_{\beta\cup\mathcal{R}(\beta)^{jo}}$ est confluent si $\to_{\mathcal{R}^{jo}}$ est confluent.

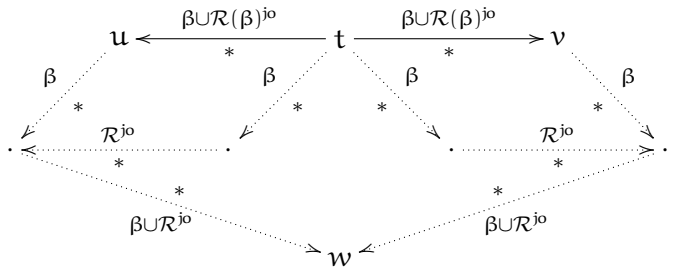
Théorème 7.2.9 *Soit $\mathcal{R}$ est un système conditionnel sur $\Lambda(\Sigma)$ algébrique, linéaire à gauche et semi-clos qui respecte $\mathfrak{a}$. Si $\to_{\mathcal{R}^{jo}}$ est confluent sur $\Lambda(\Sigma)$ alors $\to_{\beta\mathcal{R}(\beta)^{jo}}$ est confluent sur les termes qui respectent $\mathcal{R}$-héréditairement $\mathfrak{a}$.*

Preuve. Tout d'abord, comme $\mathcal{R}$ est semi-clos, linéaire à gauche et applicatif à droite, on a la confluence de $\to_{\beta\cup\mathcal{R}^{jo}}$ par le théorème 7.1.4.

Soient $t, u, v \in \Lambda(\Sigma)$ tels que t respecte $\mathcal{R}$-héréditairement $\mathfrak{a}$ et

$$u \quad \leftarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}} \quad t \quad \to^*_{\beta\cup\mathcal{R}(\beta)^{jo}} \quad v\,.$$

Par le lemme 7.2.8 appliqué deux fois et la confluence de $\to_{\beta\cup\mathcal{R}^{jo}}$, il existe w tel que $u \to^*_{\beta\cup\mathcal{R}^{jo}} w \leftarrow^*_{\beta\cup\mathcal{R}^{jo}} v$. On conclut par le fait que $\to_{\mathcal{R}^{jo}} \subseteq \to_{\mathcal{R}(\beta)^{jo}}$. En image :



□

7.2.2 Confluence sur les termes faiblement normalisants

Dans cette section, nous étendons les résultats de la section 7.1.2 à la relation $\to_{\mathcal{R}(\beta)^{jo}_i}$. Le point principal est d'obtenir un analogue au lemme 7.1.15. Comme en 7.2.1, pour

tout $i \in \mathbb{N}$ on projette des pas de $\to_{\mathcal{R}(\beta)^{jo}_i}$ sur des pas de $\to_{\mathcal{R}^{jo}_i}$. On veut donc obtenir la propriété suivante, qui implique (7.10) :

$$\begin{array}{ccc} t & \xrightarrow[*]{\beta\cup\mathcal{R}(\beta)^{jo}_i} & u \\ {\scriptstyle\beta}\,\vdots\,{\scriptstyle *} & & {\scriptstyle *}\,\vdots\,{\scriptstyle\beta} \\ \downarrow & & \downarrow \\ \beta nf(t) & \xrightarrow[\mathcal{R}^{jo}_i]{*} & \beta nf(u) \end{array} \tag{7.14}$$

Pour cela, on utilise exactement le même outillage qu'en 7.1.2. On se place sur des termes ayant une forme β-normale vérifiant une arité compatible avec le système de réécriture et on raisonne par induction sur $\succ_h$. On suppose aussi que les règles sont algébriques.

On écrit $\rhd_{\mathcal{R}(\beta)^{jo}_i}$ pour désigner la parallélisation emboîtée de la réécriture β-conditionnelle par joignabilité, définie en 7.1.12. Cette relation satisfait donc à la propriété 7.1.13 et au lemme 7.1.14.

Nous pouvons maintenant prouver (7.14) en utilisant exactement la même méthode que pour prouver (7.2) au lemme 7.1.15.

Lemme 7.2.10 *Soit $\mathcal{R}$ un système conditionnel algébrique et respectant $a : \Sigma \to \mathbb{N}$. Pour tout $i \in \{1, \dots, n\}$,*

$$\Big(t \in \mathcal{AN}_a \quad \wedge \quad t \to^*_{\beta\cup\mathcal{R}(\beta)^{jo}_i} u\Big) \implies \Big(u \in \mathcal{AN}_a \quad \wedge \quad \beta nf(t) \to^*_{\mathcal{R}^{jo}_i} \beta nf(u)\Big) \tag{7.15}$$

Preuve. On raisonne exactement comme dans la preuve du lemme 7.1.15. On prouve la propriété par induction sur $i \in \mathbb{N}$, et dans le cas d'induction, on montre que pour tout $t \in \mathcal{AN}_a$,

$$\begin{array}{ccc} t & \xrightarrow{\rhd_{\mathcal{R}(\beta)^{jo}_{i+1}}} & u \\ {\scriptstyle\beta}\,\vdots\,{\scriptstyle *} & & {\scriptstyle *}\,\vdots\,{\scriptstyle\beta} \\ \downarrow & & \downarrow \\ \beta nf(t) & \xrightarrow[\rhd_{\mathcal{R}^{jo}_{i+1}}]{} & \beta nf(u) \end{array} \tag{7.16}$$

On raisonne par induction sur $\succ_h$ en utilisant le lemme 3.2.6.

La seule différence avec le lemme 7.1.15 est le cas où $t = f(t_1, \dots, t_n)\ t_{n+1} \dots t_{n+m}$ et il y a une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et deux substitutions σ et θ telles que

$$t = \lambda\vec{x}.l\sigma \cdot t_{n+k} \dots t_{n+m} \qquad \text{et} \qquad u = \lambda\vec{x}.r\theta \cdot u_{n+k} \dots u_{n+m}$$

avec $l\sigma \to_{\mathcal{R}(\beta)^{jo}_{i+1}} r\sigma$, $\sigma \rhd_{\mathcal{R}(\beta)^{jo}_{i+1}} \theta$ et $t_p \rhd_{\mathcal{R}(\beta)^{jo}_{i+1}} u_p$ pour tout $p \in \{n+k, \dots, n+m\}$. On peut supposer que $\mathcal{D}om(\sigma) = FV(l)$.

Exactement pour les mêmes raisons qu'au lemme 7.1.15, $m = k$, et $\beta nf(t) = \lambda\vec{x}.l\sigma'$, où $\sigma' = \beta nf(\sigma)$ et u a une forme β-normale qui respecte a et s'écrit $\lambda\vec{x}.r\theta'$ où $\theta' = \beta nf(\theta)$.

On a de plus $\sigma' \rhd_{\mathcal{R}^{jo}_{i+1}} \theta'$ et pour obtenir $\beta nf(t) \rhd_{\mathcal{R}^{jo}_{i+1}} \beta nf(u)$, il reste à montrer que $l\sigma' \to_{\mathcal{R}^{jo}_{i+1}} r\sigma'$, c'est-à-dire $\vec{d}\sigma' \downarrow_{\mathcal{R}^{jo}_{i}} \vec{c}\sigma'$.

Comme $l\sigma \to_{\mathcal{R}(\beta)^{jo}_{i+1}} r\sigma$, il existe des termes $\vec{v}$ tels que

$$\vec{d}\sigma \quad \to^*_{\beta\cup\mathcal{R}(\beta)^{jo}_{i}} \quad \vec{v} \quad \leftarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_{i}} \quad \vec{c}\sigma \ .$$

Par hypothèse d'induction sur i, on a donc

$$\beta nf(\vec{d}\sigma) \quad \to^*_{\mathcal{R}^{jo}_{i}} \quad \beta nf(\vec{v}) \quad \leftarrow^*_{\mathcal{R}^{jo}_{i}} \quad \beta nf(\vec{c}\sigma)$$

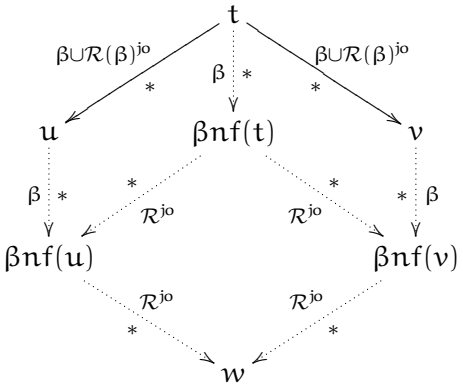
c'est-à-dire $\vec{d}\sigma' \downarrow_{\mathcal{R}^{jo}_{i}} \vec{c}\sigma'$, car $\vec{d}$ et $\vec{c}$ sont faits de termes algébriques.

On en déduit (7.15) en raisonnant exactement comme au lemme 7.1.15. □

La préservation de la confluence est une conséquence directe de la projection de la réécriture sur les formes β-normales.

Théorème 7.2.11 *Soit $\mathcal{R}$ un système conditionnel algébrique qui respecte $a : \Sigma \to \mathbb{N}$. Si $\to_{\mathcal{R}^{jo}}$ est confluent sur $\mathcal{AN}_a$ alors $\to_{\beta\cup\mathcal{R}(\beta)^{jo}}$ est confluent sur $\mathcal{AN}_a$.*

Preuve. Soient t, u, v tels que $t \in \mathcal{AN}_a$ et $u \leftarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}} t \to^*_{\beta\cup\mathcal{R}(\beta)^{jo}} v$. En appliquant deux fois le lemme 7.2.10, on obtient $\beta nf(u) \leftarrow^*_{\mathcal{R}^{jo}} \beta nf(t) \to^*_{\mathcal{R}^{jo}} \beta nf(v)$ avec $\beta nf(t), \beta nf(u)$ et $\beta nf(v) \in \mathcal{AN}_a$. On conclut par la confluence de $\to_{\mathcal{R}^{jo}}$ sur $\mathcal{AN}_a$. En image :



□

7.2.3 Systèmes orthonormaux

Dans cette section, nous donnons un critère pour la confluence de $\to_{\beta\cup\mathcal{R}(\beta)^{jo}}$ quand les conditions et les membres droits ne sont pas algébriques. Ce critère exploite la possible non-faisabilité des paires critiques de la réécriture conditionnelle.

Dans [Ohl02], il est remarqué que les résultats de confluence pour la réécriture normale et semi-équationnelle pourraient être étendus aux systèmes n'ayant pas de paire critique faisable. Mais ce raisonnement n'est pas directement applicable car l'infaisabilité des

paires critiques (qui implique la confluence) peut elle-même nécessiter la confluence. Nous circonvenons à ce problème en assurant l'infaisabilité des paires critiques par un critère syntaxique qui ne nécessite pas la confluence.

Exemple 7.2.12 *Considérons les deux règles suivantes, issues du système* $\mapsto_{\mathsf{occ}}$ *présenté à la section 4.2.1 :*

$$\begin{array}{lllllll} > (\mathsf{length}\ \mathsf{l})\ \mathsf{x} & = & \mathsf{false} & \supset & \mathsf{occ}\ (\mathsf{x} :: \mathsf{o})\ (\mathsf{node}\ \mathsf{y}\ \mathsf{l}) & \mapsto & \mathsf{false} \\ > (\mathsf{length}\ \mathsf{l})\ \mathsf{x} & = & \mathsf{true} & \supset & \mathsf{occ}\ (\mathsf{x} :: \mathsf{o})\ (\mathsf{node}\ \mathsf{y}\ \mathsf{l}) & \mapsto & \mathsf{occ}\ \mathsf{o}\ (\mathsf{get}\ \mathsf{l}\ \mathsf{x}) \end{array}$$

Ces deux règles génèrent la paire critique

$$(\geq (\mathsf{length}\ \mathsf{l})\ \mathsf{x} = \mathsf{true} \quad \wedge \quad \geq (\mathsf{length}\ \mathsf{l})\ \mathsf{x} = \mathsf{false}) \quad \supset \quad (\mathsf{false}, \mathsf{occ}\ \mathsf{o}\ (\mathsf{get}\ \mathsf{l}\ \mathsf{x}))\ .$$

Les conditions de cette paire critique ne peuvent donc pas être satisfaites par une relation confluente. En utilisant la stratification de $\rightarrow_{\mathcal{R}(\beta)^{\mathsf{jo}}}$*, il est aisé de voir que la confluence de* $\rightarrow_{\mathcal{R}(\beta)^{\mathsf{jo}}_i}$ *implique l'infaisabilité de la paire critique, qui à son tour implique la confluence de* $\rightarrow_{\mathcal{R}(\beta)^{\mathsf{jo}}_{i+1}}$*.*

Remarque 7.2.13 *Dans le cadre de la réécriture du premier ordre, des idées similaires ont été appliquées dans [GM88, KW97].*

D'autre part, dans le cadre de la combinaison du λ*-calcul avec la réécriture conditionnelle, il convient de mentionner le travail de Takahashi [Tak93]. Son approche est de conditionner les règles de réécriture par un prédicat* P *arbitraire sur les termes. La confluence est prouvée sous l'hypothèse que* P *est stable par réduction : si* $\mathsf{P}\sigma$ *est vrai et* $\sigma \rightarrow \sigma'$*, alors* $\mathsf{P}\sigma'$ *est vrai. Ceci ne s'applique pas directement aux systèmes orthogonaux, pour lesquels la stabilité des conditions par réécriture provient précisément de la confluence.*

Notre critère est une généralisation du raisonnement appliqué à l'exemple 7.2.12. Il ne dépend pas du fait que $\rightarrow_{\mathcal{R}(\beta)^{\mathsf{jo}}}$ soit ou non inclus dans $=_{\beta\cup\mathcal{R}^{\mathsf{jo}}}$ et peut donc être utilisé sur des systèmes ayant des conditions ou des membres droits non-algébriques. Les systèmes satisfaisant ce critère sont dits « orthonormaux ».

Définition 7.2.14 (Systèmes orthonormaux) *Un système conditionnel* $\mathcal{R}$ *est* orthonormal *si*

- *il est linéaire à gauche,*
- *dans chaque règle* $\vec{c} = \vec{d} \supset l \mapsto_{\mathcal{R}} r$*, les termes dans* $\vec{c}$ *sont clos, en forme* β*-normale et ne contiennent pas de symboles définis,*
- *pour chaque paire critique* $\vec{d} = \vec{c} \supset (s, t)$*, il existe* $i \neq j$ *tels que* $d_i = d_j$ *et* $c_i \neq c_j$*.*

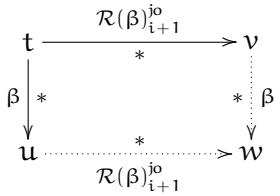
Un système orthonormal est donc linéaire à gauche et semi-clos, mais il peut n'être compatible avec aucune fonction d'arité applicative $a : \Sigma \rightarrow \mathbb{N}$, et n'être pas non plus algébrique. De plus, dans une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, les termes $\vec{c}$ sont en forme $\rightarrow_{\beta\cup\mathcal{R}(\beta)^{\mathsf{jo}}}$-normale. La relation $\rightarrow_{\mathcal{R}(\beta)^{\mathsf{jo}}}$ est donc une relation de réécriture normale au sens de la définition 4.1.8.

Exemple 7.2.15 *Le système présenté en 4.2.1 est orthonormal.*

Nous prouvons maintenant que $\to_{\beta\cup\mathcal{R}(\beta)^{jo}}$ est confluent en profondeur (c.-à-d. que les relations $\to_{\beta\cup\mathcal{R}(\beta)^{jo}_i}$ et $\to_{\beta\cup\mathcal{R}(\beta)^{jo}_j}$ commutent pour tout $i,j\in\mathbb{N}$) lorsque $\mathcal{R}$ est orthonormal.

Le premier lemme dit que la confluence de $\to_{\beta\cup\mathcal{R}(\beta)^{jo}_i}$ implique la commutation de $\to_{\beta\cup\mathcal{R}(\beta)^{jo}_{i+1}}$ avec $\to_\beta$.

Lemme 7.2.16 *Soit $\mathcal{R}$ un système orthonormal et $i\in\mathbb{N}$ tel que $\to_{\beta\cup\mathcal{R}(\beta)_i}$ est confluent. Pour tout $t,u,v\in\Lambda(\Sigma)$, si $u\leftarrow^*_\beta t\to^*_{\mathcal{R}(\beta)^{jo}_{i+1}} v$, alors il existe un terme w tel que $u\to^*_{\mathcal{R}(\beta)^{jo}_{i+1}} w\leftarrow^*_\beta v$. En image,*



Preuve. La preuve est presqu'identique à celle du lemme 2.1.15. Nous prouvons seulement que s'il y a une règle $\vec{d}=\vec{c}\supset l\mapsto_{\mathcal{R}} r$, une substitution σ et un terme u tels que $u\lhd_\beta l\sigma\to_{\mathcal{R}(\beta)^{jo}_{i+1}} r\sigma$, alors il existe un terme v tel que $u\to_{\mathcal{R}(\beta)^{jo}_{i+1}} v\lhd_\beta r\sigma$.

Par la linéarité de l et la proposition 6.1.2, il existe une substitution σ' telle que $\sigma\rhd_\beta\sigma'$ et $u=l\sigma'$. Il s'en suit par le lemme 2.3.6 que $r\sigma\rhd_\beta r\sigma'$ et $\vec{d}\sigma\rhd_\beta\vec{d}\sigma'$, soit $\vec{d}\sigma\to^*_\beta\vec{d}\sigma'$ par le lemme 5.1.12.

Pour conclure que le terme $w=_{\text{def}} r\sigma'$ convient, il reste à montrer que $l\sigma'\to_{\mathcal{R}^{jo}_{i+1}} r\sigma'$, c'est-à-dire que $\vec{d}\sigma'\downarrow_{\beta\cup\mathcal{R}(\beta)^{jo}_i}\vec{c}$. Or, on a $\vec{d}\sigma\downarrow_{\beta\cup\mathcal{R}(\beta)^{jo}_i}\vec{c}$, donc $\vec{d}\sigma'\leftrightarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_i}\vec{c}$. Comme la relation $\to_{\beta\cup\mathcal{R}(\beta)^{jo}_i}$ est confluente par hypothèse, on a $\vec{d}\sigma'\downarrow_{\beta\cup\mathcal{R}(\beta)^{jo}_i}\vec{c}$, donc $\vec{d}\sigma'\to^*_{\beta\cup\mathcal{R}(\beta)^{jo}_i}\vec{c}$, car les termes $\vec{c}$ sont $\to_{\beta\cup\mathcal{R}(\beta)^{jo}}$-normaux. On en déduit que $l\sigma'\to_{\mathcal{R}(\beta)^{jo}_{i+1}} r\sigma'$. □

Le reste de la preuve suit la démarche standard des preuves de confluence pour les systèmes conditionnels orthogonaux, voir [Ohl02]. La propriété importante est celle des mouvements parallèles. On utilise les relations $\to_{\|\mathcal{R}(\beta)^{jo}_i}$ qui sont les clôtures parallèles des relations $\to_{\mathcal{R}(\beta)^{jo}_i}$, au sens de 2.3.5. Dans notre cas, la propriété des mouvements parallèles s'énonce comme suit :

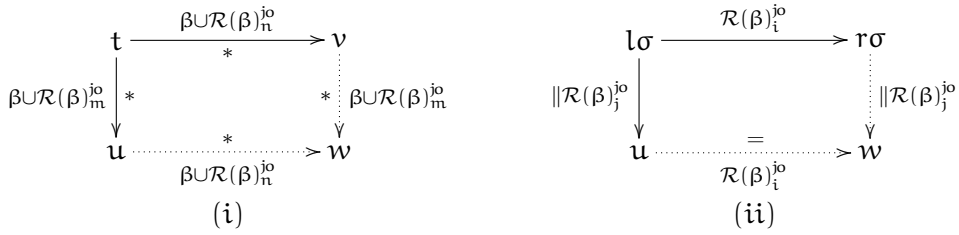
Mouvements parallèles. Pour tout $i,j\in\mathbb{N}$, si

$$\forall n,m\in\mathbb{N}.\quad \{n,m\}<_{\text{mul}}\{i,j\}\quad\Longrightarrow\quad \to_{\beta\cup\mathcal{R}(\beta)^{jo}_n}\text{ commute avec }\to_{\beta\cup\mathcal{R}(\beta)^{jo}_m}$$

alors $\to_{\|\mathcal{R}(\beta)^{jo}_i}$ et $\to_{\|\mathcal{R}(\beta)^{jo}_j}$ commutent.

La preuve occupe les lemmes 7.2.17 et 7.2.18.

Lemme 7.2.17 *Soit $\mathcal{R}$ un système orthonormal et $i, j \geq 0$. Supposons que la propriété* (i) *soit vérifiée pour tout n, m tels que $\{n, m\} <_{mul} \{i, j\}$. Alors, la propriété* (ii) *est satisfaite pour toute règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$.*



(i) (ii)

PREUVE. La cas $i = 0$ est trivial car $\rightarrow_{\mathcal{R}(\beta)^{jo}_0} = \emptyset$. Si $j = 0$, alors $u = l\sigma$ et $w = r\sigma$ convient. Soient $i, j > 0$ et supposons que dans le pas $l\sigma \rightarrow_{\|\mathcal{R}(\beta)^{jo}_j} u$, il y ait une paire critique entre $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et une règle $\vec{d}' = \vec{c}' \supset l' \mapsto_{\mathcal{R}} r'$. Il existe donc $p \in \mathcal{P}os(l)$ et deux substitutions μ et μ' telles qu'on ait une paire critique conditionnelle de la forme

$$\vec{d}'' \;=\; \vec{c}'' \;\supset\; (l[r']_p\mu, r\mu)\ ,$$

avec $\sigma = \mu' \circ \mu$. Comme $\mathcal{R}$ est orthonormal, on a $|\vec{d}''| \geq 2$ et il existe $n \neq m$ tels que $d''_n = d''_m$ et $c_n \neq c_m$. Soit $h =_{\text{def}} max(i, j) - 1$ (donc $\{h, h\} <_{\text{mul}} \{i, j\}$). On a alors

$$c''_n \quad \leftarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_h} \quad d''_n\sigma \quad = \quad d''_m\sigma \quad \rightarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_h} \quad c''_m\ ,$$

ce qui est impossible car c''_n et c''_m sont des formes $\rightarrow_{\beta\cup\mathcal{R}(\beta)^{jo}_h}$-normales distinctes alors que $\rightarrow_{\beta\cup\mathcal{R}(\beta)^{jo}_h}$ est confluente.

Il s'en suit que tous les rédexes contractés dans le pas $l\sigma \rightarrow_{\|\mathcal{R}(\beta)^{jo}_j} u$ sont dans la substitution σ. Comme l est linéaire, on a donc $u = l\sigma'$ avec $\sigma \rightarrow_{\|\mathcal{R}(\beta)^{jo}_j} \sigma'$. Il s'en suit que $r\sigma \rightarrow_{\|\mathcal{R}(\beta)^{jo}_j} r\sigma'$ et il reste à montrer que $l\sigma' \rightarrow_{\mathcal{R}(\beta)^{jo}_i} r\sigma'$. Or, on a

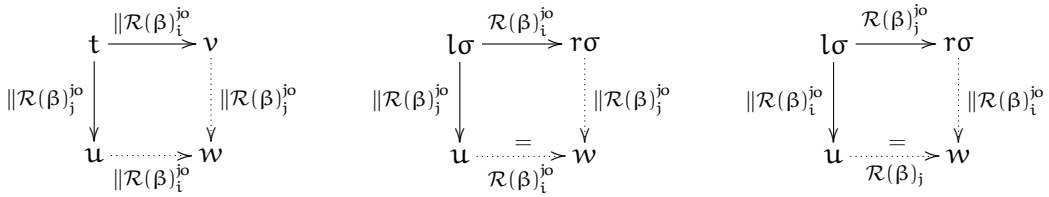
$$\vec{d}\sigma' \quad \leftarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_j} \quad \vec{d}\sigma \quad \rightarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_{i-1}} \quad \vec{c}$$

donc comme $\{j, i-1\} <_{\text{mul}} \{i, j\}$, il existe $\vec{w}$ tels que

$$\vec{d}\sigma' \quad \rightarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_{i-1}} \quad \vec{w} \quad \leftarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_j} \quad \vec{c}\ ,$$

d'où $\vec{d}\sigma' \rightarrow^*_{\beta\cup\mathcal{R}(\beta)^{jo}_{i-1}} \vec{c}$ car $\mathcal{R}$ est orthonormal. □

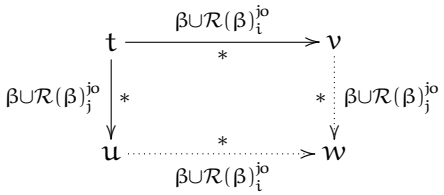
Lemme 7.2.18 *Soit $\mathcal{R}$ un système orthonormal et $i, j \geq 0$. La propriété* (iii) *est satisfaite si et seulement si, pour toute règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, les propriétés* (iv) *et* (v) *le sont.*



PREUVE. Le sens « seulement si » est trivial. Pour le sens « si », soient trois termes t, u et v tels que $u \leftarrow_{\|\mathcal{R}(\beta)_j^{jo}} t \rightarrow_{\|\mathcal{R}(\beta)_i^{jo}} v$. Si $t = v$ (resp. $t = u$), alors $w =_{def} u$ (resp. $w =_{def} v$) convient. Sinon, on raisonne par induction sur t. Si il une des deux réductions est une réduction de tête, alors on conclut par (iv) et (v). Supposons maintenant qu'il n'y a pas de réduction de tête. Si t est une abstraction alors on conclut par hypothèse d'induction. Sinon, t est soit une application $t_1 t_2$, soit de la forme $f(t_1, \dots, t_n)$. Les deux cas se traitent de la même manière. Prenons celui de l'application. Par hypothèse, $u = u_1 u_2$ et $v = v_1 v_2$ et on a $u_k \leftarrow_{\|\mathcal{R}(\beta)_j^{jo}} t_k \rightarrow_{\|\mathcal{R}(\beta)_i^{jo}} v_k$ pour tout $k \in \{1, 2\}$. Par hypothèse d'induction, il existe w_k tel que $u_k \rightarrow_{\|\mathcal{R}(\beta)_i^{jo}} w_k \leftarrow_{\|\mathcal{R}(\beta)_j^{jo}} v_k$, donc $u_1 u_2 \rightarrow_{\|\mathcal{R}(\beta)_i^{jo}} w_1 w_2 \leftarrow_{\|\mathcal{R}(\beta)_j^{jo}} v_1 v_2$. □

Maintenant, une induction sur $<_{mul}$ permet d'obtenir la commutation $\rightarrow_{\beta\cup\mathcal{R}(\beta)_i^{jo}}$ avec $\rightarrow_{\beta\cup\mathcal{R}(\beta)_j^{jo}}$ pour tout $i, j \in \mathbb{N}$. Il s'en suit que $\rightarrow_{\beta\cup\mathcal{R}(\beta)^{jo}}$ est confluente « en profondeur ».

Théorème 7.2.19 *Si $\mathcal{R}$ est un système orthonormal, alors pour tout $i, j \in \mathbb{N}$ on a*



(7.17)

PREUVE. Par induction sur $>_{mul}$, on prouve la commutation de (7.17) pour tout $i, j \in \mathbb{N}$. La cas de base, dans lequel $i = j = 0$, est trivial.

Maintenant, soit $i > 0$ et supposons que (7.17) commute pour tout $\{n, m\} <_{mul} \{i, 0\}$. Comme $\{i-1, i-1\} <_{mul} \{i, 0\}$, la relation $\rightarrow_{\beta\cup\mathcal{R}(\beta)_{i-1}^{jo}}$ est confluente, donc la commutation de $\rightarrow_{\beta\cup\mathcal{R}(\beta)_i^{jo}}$ avec $\rightarrow_{\beta\cup\mathcal{R}(\beta)_0^{jo}}$ $(=\rightarrow_\beta)$ suit du lemme 7.2.16.

Le dernier cas est celui où $i, j > 0$. par hypothèse d'induction, en utilisant les lemmes 7.2.17 et 7.2.18, on obtient la commutation de $\rightarrow_{\|\mathcal{R}(\beta)_i^{jo}}$ avec $\rightarrow_{\|\mathcal{R}(\beta)_j^{jo}}$. Comme $\{i-1, i-1\} <_{mul} \{i, j\}$ (resp. $\{j-1, j-1\} <_{mul} \{i, j\}$), par le lemme 7.2.16, les relations $\rightarrow_\beta$ et $\rightarrow_{\mathcal{R}(\beta)_i^{jo}}$ (resp. $\rightarrow_{\mathcal{R}(\beta)_j^{jo}}$) commutent. Il s'en suit que $\rightarrow_{\beta\cup\mathcal{R}(\beta)_i^{jo}}$ commute avec $\rightarrow_{\beta\cup\mathcal{R}(\beta)_j^{jo}}$. □

Exemple 7.2.20 *La relation $\rightarrow_{\beta\cup\mathcal{R}(\beta)^{jo}}$ induite par le système présenté en 4.2.1 est confluente.*

Chapitre 8

Préservation de la confluence par curryfication

L'objectif de ce chapitre est d'étendre à la réécriture conditionnelle le résultat de Kahrs [Kah95] sur la préservation de la confluence par curryfication de la réécriture non conditionnelle. Nos résultats ne sont cependant pas complètement satisfaisants car nous nous restreignons aux règles conditionnelles *non-effondrantes*, c'est à dire aux règles dont le membre droit n'est pas une variable (voir la définition 4.1.1), alors que cette hypothèse n'est pas nécessaire pour la réécriture non conditionnelle [Kah95]. Nous n'avons cependant pas d'exemple de non préservation de la confluence par curryfication pour la réécriture conditionnelle effondrante.

La curryfication est une traduction, évoquée en 3.1.13, des termes de $\mathcal{T}er(\Sigma, \mathcal{X})$ vers un sous-ensemble des termes algébriques de $\mathcal{L}(\Sigma_{\mathcal{C}})$, où $\Sigma_{\mathcal{C}}$ contient un symbole $\mathsf{f}_{\mathcal{C}}$ d'arité nulle pour chaque $\mathsf{f} \in \Sigma$.

Nous procédons selon les étapes suivantes :

(i) Nous utilisons et adaptons le résultat de [Kah95] pour montrer que la confluence de la réécriture conditionnelle non-effondrante est préservée lors du passage de $\mathcal{T}er(\Sigma, \mathcal{X})$ à $\mathcal{T}er(\Sigma_{\mathcal{C}\mathrm{App}}, \mathcal{X})$.

(ii) Nous utilisons le théorème 5.3.4 pour passer de $\mathcal{T}er(\Sigma_{\mathcal{C}\mathsf{App}}, \mathcal{X})$ à $\Lambda(\Sigma_{\mathcal{C}})$.

D'autre part, nous montrons que la confluence sur les termes clos n'est pas forcément préservée par curryfication.

8.1 Curryfication

Nous définissons la curryfication et donnons quelques exemples. En particulier, nous montrons que la confluence sur les termes clos n'est pas forcément préservée par curryfication.

Nous avons présenté brièvement aux exemples 3.1.13 et 3.1.14 les curryfications des signatures Σ_{Nat} et Σ_{List} et du système de réécriture $\mapsto_{\mathsf{ite}}$. Pour voir comment curryfier un système de réécriture conditionnel, considérons une possible spécialisation au premier ordre de la fonction filter du système (4.1) :

$$\begin{array}{lllll} & & \mathsf{filter}(\mathsf{nil}) & \mapsto_{\mathsf{filter}(\_)} & \mathsf{nil} \\ \mathsf{test}(\mathsf{x}) = \mathsf{true} & \supset & \mathsf{filter}(\mathsf{x} :: \mathsf{xs}) & \mapsto_{\mathsf{filter}(\_)} & \mathsf{x} :: \mathsf{filter}(\mathsf{xs}) \\ \mathsf{test}(\mathsf{x}) = \mathsf{false} & \supset & \mathsf{filter}(\mathsf{x} :: \mathsf{xs}) & \mapsto_{\mathsf{filter}(\_)} & \mathsf{filter}(\mathsf{xs}) \end{array} \tag{8.1}$$

où $\mathsf{test}(\_)$ est un symbole unaire. C'est donc un système conditionnel sur la signature $\Sigma_{\mathsf{filter}(\_)} =_{\text{def}} \Sigma_{\mathsf{List}} \uplus \{\mathsf{true}, \mathsf{false}, \mathsf{test}(\_), \mathsf{filter}(\_)\}$.

On peut lui associer le système suivant, construit avec un symbole d'application et des symboles d'arité nulle :

$$\begin{array}{lcl@{\quad}l@{\quad}l@{\quad}l} & & & \mathsf{filter}\cdot\mathsf{nil} & \mapsto_{\mathsf{filter}(\_)_{\mathcal{C}}} & \mathsf{nil} \\ \mathsf{test}\cdot x & = & \mathsf{true} \supset & \mathsf{filter}\cdot(\mathsf{cons}\cdot x\cdot xs) & \mapsto_{\mathsf{filter}(\_)_{\mathcal{C}}} & \mathsf{cons}\cdot x\cdot(\mathsf{filter}\cdot xs) \\ \mathsf{test}\cdot x & = & \mathsf{false} \supset & \mathsf{filter}\cdot(\mathsf{cons}\cdot x\cdot xs) & \mapsto_{\mathsf{filter}(\_)_{\mathcal{C}}} & \mathsf{filter}\cdot xs \end{array} \tag{8.2}$$

C'est donc un système conditionnel sur la signature $(\Sigma_{\mathsf{filter}(\_)})_{\mathcal{C}\mathrm{App}}$, où

$$(\Sigma_{\mathsf{filter}(\_)})_{\mathcal{C}} \quad =_{\text{def}} \quad \Sigma_{\mathsf{List}\mathcal{C}} \uplus \{\mathsf{true}, \mathsf{false}, \mathsf{test}, \mathsf{filter}\} \ .$$

La curryfication est le processus qui permet de passer de $(\mathcal{T}\mathrm{er}(\Sigma_{\mathsf{filter}(\_)}, \mathcal{X}), \to_{\mathsf{filter}(\_)})$ à $(\mathcal{T}\mathrm{er}((\Sigma_{\mathsf{filter}(\_)})_{\mathcal{C}\mathrm{App}}, \mathcal{X}), \to_{\mathsf{filter}(\_)_{\mathcal{C}}})$. Plus généralement, étant donné un ensemble $\mathcal{R}$ de règles conditionnelles sur $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$, et une relation de réécriture conditionnelle $\to_{\mathcal{R}^X}$ (avec $X \in \{\mathsf{se}, \mathsf{jo}, \mathsf{o}\}$), nous sommes intéressés par la relation curryfiée $\to_{\mathcal{R}^X_{\mathcal{C}}}$, définie sur $\Lambda(\Sigma_{\mathcal{C}})$ et issue des règles $\mathcal{R}_{\mathcal{C}}$. Voyons d'abord comment curryfier un ensemble de termes.

Définition 8.1.1 (Curryfication) *Étant données deux signatures Σ, $\Sigma_{\mathcal{C}}$ et une injection $(\_)_{\mathcal{C}} : \Sigma \to \Sigma_{\mathcal{C}}$ telle que tout $f \in \Sigma$ est associé à un symbole $f_{\mathcal{C}}$ d'arité nulle, la fonction $(\_)_{\mathcal{C}} : \mathcal{T}\mathrm{er}(\Sigma, \mathcal{X}) \to \mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}}, \mathcal{X})$ définie inductivement par*

$$\begin{array}{ll} x_{\mathcal{C}} & =_{def} \ x \\ (f(t_1, \ldots, t_n))_{\mathcal{C}} & =_{def} \ (f_{\mathcal{C}}\cdot(t_1)_{\mathcal{C}})\cdots(t_n)_{\mathcal{C}} \end{array}$$

est la fonction de curryfication de $(\_)_{\mathcal{C}} : \Sigma \to \Sigma_{\mathcal{C}}$.

Notons que $t_{\mathcal{C}}$ est un terme algébrique pour tout $t \in \mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$. Il est utile de pouvoir considérer la curryfication comme une fonction de $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}}, \mathcal{X})$ dans $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}}, \mathcal{X})$, en posant $(t_1 \cdot t_2)_{\mathcal{C}} =_{\text{def}} (t_1)_{\mathcal{C}} \cdot (t_2)_{\mathcal{C}}$. Lorsque le contexte le permet, nous désignons par $\Sigma_{\mathcal{C}}$ le couple $(\Sigma_{\mathcal{C}}, (\_)_{\mathcal{C}})$ où $(\_)_{\mathcal{C}}$ est la fonction de curryfication de $(\_)_{\mathcal{C}} : \Sigma \to \Sigma_{\mathcal{C}}$.

Définition 8.1.2 (Règles conditionnelles curryfiées) *Étant données une fonction de curryfication $(\_)_{\mathcal{C}} : \Sigma \to \Sigma_{\mathcal{C}}$ et un ensemble $\mathcal{R}$ de règles conditionnelles sur $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$, l'ensemble des* règles curryfiées *par $(\_)_{\mathcal{C}}$ est l'ensemble*

$$\mathcal{R}_{\mathcal{C}} \quad =_{def} \quad \{\vec{d}_{\mathcal{C}} = \vec{c}_{\mathcal{C}} \supset l_{\mathcal{C}} \mapsto r_{\mathcal{C}} \mid \vec{d} = \vec{c} \supset l \mapsto r \in \mathcal{R}\} \ .$$

Remarque 8.1.3 *L'ensemble de règles conditionnelles $\mathcal{R}_{\mathcal{C}}$ est un TRS sur $\mathcal{L}(\Sigma_{\mathcal{C}})$ au même titre que $\mathcal{R}$ est un TRS sur $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$.*

Voici maintenant un exemple que la confluence sur les termes clos n'est pas préservée par curryfication : si $\to_{\mathcal{R}}$ est confluent sur $\mathcal{T}\mathrm{er}(\Sigma)$ alors $\to_{\mathcal{R}_{\mathcal{C}}}$ n'est pas nécessairement confluent sur $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}})$.

Exemple 8.1.4 (La confluence close n'est pas préservée par curryfication) *Soit la signature* $\Sigma =_{def} \{\mathsf{f}(\_), \mathsf{a}\}$. *Le système* $\mathcal{R}$ *suivant est confluent sur* $\mathcal{T}\mathrm{er}(\Sigma)$ *mais pas sur* $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$ *:*

$$\mathsf{f}(\mathsf{x}) \quad \mapsto \quad \mathsf{x} \qquad\qquad \mathsf{f}(\mathsf{x}) \quad \mapsto \quad \mathsf{a}\ .$$

En effet, tout terme de $\mathcal{T}\mathrm{er}(\Sigma)$ *est de la forme* $\mathsf{f}^n(\mathsf{a})$ *et se réduit vers* a. *Soit* $\Sigma_{\mathcal{C}}$ *une signature qui contient* $\{\mathsf{f}_{\mathcal{C}}, \mathsf{a}_{\mathcal{C}}\}$. *Alors dans* $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}})$, *où* $\mathcal{R}_{\mathcal{C}}$ *est le système*

$$\mathsf{f}_{\mathcal{C}} \cdot \mathsf{x} \quad \mapsto \quad \mathsf{x} \qquad\qquad \mathsf{f}_{\mathcal{C}} \cdot \mathsf{x} \quad \mapsto \quad \mathsf{a}_{\mathcal{C}}\ ,$$

on a la paire critique injoignable suivante :

$$\mathsf{f}_{\mathcal{C}} \quad \leftarrow \quad \mathsf{f}_{\mathcal{C}} \cdot \mathsf{f}_{\mathcal{C}} \quad \rightarrow \quad \mathsf{a}_{\mathcal{C}}\ .$$

Remarque 8.1.5

— *Cet exemple n'est pas très surprenant : d'après le théorème 5.3.8, comme* $\mathcal{R}$ *est confluent sur* $\mathcal{T}\mathrm{er}(\Sigma)$ *mais pas sur* $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$, *il existe une extension* Σ' *de* Σ *telle que* $\mathcal{R}$ *n'est pas confluent sur* $\mathcal{T}\mathrm{er}(\Sigma')$ *(c'est en fait le cas pour tout* Σ' *contenant strictement* Σ*). C'est moralement ce que permet de faire la curryfication, d'où la non confluence de* $\mathcal{R}_{\mathcal{C}}$ *sur* $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}}, \mathcal{X})$.

— *Cet exemple peut être bien typé, par exemple dans le système F si* $\mathsf{f}_{\mathcal{C}}$ *est l'identité* $\mathsf{id} : \forall \mathsf{X}.\mathsf{X} \Rightarrow \mathsf{X}$, *et si* $\tau(\mathsf{a}_{\mathcal{C}}) = \forall \mathsf{X}.\mathsf{X}$.
Par contre, l'exemple ne peut être bien typé si on impose que le type des symboles curryfiés provient directement du type des symboles non curryfiés. Dans ce cadre, la préservation de la confluence sur les termes bien typés a été montrée dans [BTG94] pour le système F.
D'autre part, Dougherty montre dans [Dou92] que la confluence est préservée par curryfication vers les termes $\mathcal{R}$*-stables. Ce sous-ensemble ne contient pas le terme* $\mathsf{id} \cdot \mathsf{id} \cdot \mathsf{id}$ *(voir la remarque 6.1.15), qui peut pourtant être typé si* $\mathsf{id} : \forall \mathsf{X}.\mathsf{X} \Rightarrow \mathsf{X}$.

— *La première et seule preuve de préservation de la confluence par curryfication sur l'ensemble des* λ*-termes applicatifs est à notre connaissance celle de [Kah95].*
Notons que la preuve de Kahrs fait intervenir la confluence du système $\rightarrow_{\mathrm{PP}}$ *(voir section 8.3), qui est la combinaison hiérarchique de deux systèmes confluents. Rappelons que les combinaisons hiérarchiques de systèmes confluents de sont pas confluentes en général (voir exemple 5.3.10).*

8.2 Un essai de preuve directe

Commençons par voir pourquoi nous n'avons pas réussi à appliquer directement le résultat de [Kah95] à la réécriture conditionnelle. Le résultat final de préservation de la confluence de Kahrs est le suivant (voir aussi la définition 3.1 dans [Kah95]) :

Théorème 8.2.1 (Théorème 5.2 dans [Kah95]) *Soit un TRS* R *sur* $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$ *et* $\Sigma_{\mathcal{C}}$ *une curryfication de* Σ. *Si* $\rightarrow_{\mathsf{R}}$ *est confluent sur* $\mathcal{T}\mathrm{er}(\Sigma, \mathcal{X})$, *alors la relation*

$$\mathcal{C}(\rightarrow_{\mathsf{R}}) \quad =_{def} \quad \{(\mathsf{t}_{\mathcal{C}}, \mathsf{u}_{\mathcal{C}}) \mid \mathsf{t} \rightarrow_{\mathsf{R}} \mathsf{u}\}$$

est confluente sur $\mathcal{T}\mathrm{er}(\Sigma_{\mathcal{C}\mathrm{App}}, \mathcal{X})$.

En particulier, si $\mathcal{R}$ est un ensemble de règles non conditionnelles, alors $\mathcal{C}(\rightarrow_{\mathcal{R}}) = \rightarrow_{\mathcal{R}_\mathcal{C}}$ et on obtient la confluence du TRS $(\mathcal{T}er(\Sigma_{\mathcal{C}\mathsf{App}}, \mathcal{X}), \mathcal{R}_\mathcal{C})$.

Une relation de réécriture conditionnelle étant aussi un TRS, et il est tentant d'utiliser le théorème 8.2.1 pour obtenir la confluence de $\mathcal{C}(\rightarrow_{\mathcal{R}^X})$ à partir de celle de $\rightarrow_{\mathcal{R}^X}$ et en déduire la confluence de $\rightarrow_{\mathcal{R}^X_\mathcal{C}}$. La difficulté, lorsque $\mathcal{R}$ contient des règles conditionnelles, est qu'il n'est pas évident que $\rightarrow_{\mathcal{R}^X_\mathcal{C}}$ et $\mathcal{C}(\rightarrow_{\mathcal{R}^X})$ coïncident.

Bien qu'il soit facile de voir que $\mathcal{C}(\rightarrow_{\mathcal{R}^X}) \subseteq \rightarrow_{\mathcal{R}^X_\mathcal{C}}$, le contraire (s'il est vrai) ne semble pas avoir de preuve simple.

Lemme 8.2.2 *Soit un ensemble $\mathcal{R}$ de règles conditionnelles sur $\mathcal{T}er(\Sigma, \mathcal{X})$ et $\Sigma_\mathcal{C}$ une curryfication de Σ. Pour tout $X \in \{se, jo, o\}$, si $t\ \mathcal{C}(\rightarrow_{\mathcal{R}^X})\ u$ alors $t \rightarrow_{\mathcal{R}^X_\mathcal{C}} u$.*

Preuve. Si $t\ \mathcal{C}(\rightarrow_{\mathcal{R}^X})\ u$ alors par définition il existe $t', u' \in \mathcal{T}er(\Sigma, \mathcal{X})$ tels que $t = t'_\mathcal{C}$, $t' \rightarrow_{\mathcal{R}^X} u'$ et $u'_\mathcal{C} = u$. La proposition 8.3.5 implique que $t'_\mathcal{C} \rightarrow_{\mathcal{R}^X_\mathcal{C}} u'_\mathcal{C}$. □

Considérons la propriété inverse :

$$t \quad \rightarrow_{\mathcal{R}^X_\mathcal{C}} \quad u \qquad \text{implique} \qquad t \quad \mathcal{C}(\rightarrow_{\mathcal{R}^X}) \quad u\ .$$

Pour voir où se situe la difficulté, essayons une preuve directe dans le cadre de la réécriture par joignabilité. L'idée est d'essayer de prouver par induction sur $i \in \mathbb{N}$ que $t \rightarrow_{\mathcal{R}_\mathcal{C}{}^i} u$ implique $t\,\mathcal{C}(\rightarrow_{\mathcal{R}_i})\,u$. Le cas de base $i = 0$ est évident. Supposons la propriété vraie pour $i \geq 0$ et essayons de la montrer pour $i + 1$.

Soit une règle $\vec{d} = \vec{c} \supset l \rightarrow r \in \mathcal{R}$ et une substitution σ telles que $l_\mathcal{C}\sigma \rightarrow_{\mathcal{R}_\mathcal{C}{}^{i+1}} r_\mathcal{C}\sigma$. Il existe donc des termes $\vec{v}$ tels que $\vec{d}_\mathcal{C}\sigma \rightarrow^*_{\mathcal{R}_\mathcal{C}{}^i} \vec{v}$ et $\vec{c}_\mathcal{C}\sigma \rightarrow^*_{\mathcal{R}_\mathcal{C}{}^i} \vec{v}$. Par hypothèse d'induction on obtient que $\vec{d}_\mathcal{C}\sigma\ \mathcal{C}(\rightarrow_\mathcal{R})^*\ \vec{v}$ et $\vec{c}_\mathcal{C}\sigma\ \mathcal{C}(\rightarrow_\mathcal{R})^*\ \vec{v}$. On aimerait pouvoir en déduire que $l_\mathcal{C}\sigma\,\mathcal{C}(\rightarrow_\mathcal{R})^*\,r_\mathcal{C}\sigma$, mais pour ce faire, il faudrait pouvoir trouver une substitution θ et des termes $\vec{w}$ tels que $\vec{d}\theta \rightarrow^*_\mathcal{R} \vec{w} \leftarrow^*_\mathcal{R} \vec{c}\theta$, avec de plus $(\vec{d}\theta)_\mathcal{C} = \vec{d}_\mathcal{C}\sigma$, $(\vec{c}\theta)_\mathcal{C} = \vec{c}_\mathcal{C}\sigma$ et $\vec{w}_\mathcal{C} = \vec{v}$.

Nous pensons qu'il est peut-être possible d'arriver à le montrer en utilisant un outillage approprié, peut être comme celui utilisé dans les preuves de modularité [Toy87, KMTV94], mais cela ne semble pas évident.

8.3 Paramétrisation partielle d'un système de réécriture

Nous devons donc utiliser le résultat de Kahrs de manière un peu plus fine. Cette section est dévolue à l'étude et à la présentation des outils de sa preuve que nous réutilisons.

8.3.1 Paramétrisation partielle

L'élément principal est un ensemble de termes dans lequel cohabitent les termes curryfiés, les termes non curryfiés et les termes partiellement curryfiés. Sur ces termes nous considérons la relation $\rightarrow_\mathcal{R}$ issue d'un TRS $\mathcal{R}$ ainsi que des relations de réécriture implantant la curryfication et la décurryfication. Le TRS ainsi obtenu est la *paramétrisation partielle* du TRS de départ.

Dans ce qui suit, il est pratique pour l'homogénéité des notations de considérer des termes dont les symboles sont annotés par leur arité. Ainsi, on se place dans une Σ-algèbre $\mathcal{A} = (A, \Sigma_{\mathcal{A}})$ libre sur $\mathcal{X}$ telle que $f_{\mathcal{A}} \in \Sigma_{\mathcal{A}}$ s'écrive f_n pour tout $f \in \Sigma_n$. Bien entendu, cela ne change rien aux résultats, qui sont valables dans n'importe quelle Σ-algèbre libre sur $\mathcal{X}$.

Définition 8.3.1 (Paramétrisation partielle) La paramétrisation partielle *d'un TRS* $(\mathcal{T}er(\Sigma, \mathcal{X}), \mathcal{R})$ *est le TRS* $(\mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X}), \mathsf{PP}(\mathcal{R}))$ *dans lequel*

— *la signature* $\mathsf{PP}(\Sigma)$ *est* $\{f_m \mid f \in \Sigma_n \wedge 0 \leq m \leq n\}_{App}$ *où chaque* f_m *est d'arité* m,

— *la relation* $\mathsf{PP}(\mathcal{R})$ *est* $\mathcal{R} \cup \mathcal{U}$ *où* $\mathcal{U}$ *est la plus petite relation telle que pour tout* $f_n, f_{n+1} \in \mathsf{PP}(\Sigma)$,

$$f_n(t_1, \ldots, t_n) \cdot t_{n+1} \quad \mapsto_{\mathcal{U}} \quad f_{n+1}(t_1, \ldots, t_n, t_{n+1}) \ .$$

Notons $\to_{\mathcal{C}}$ pour la relation de réécriture sur $\mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$ issue des règles $\mathcal{U}^{-1}$. Il est aisé de voir que les systèmes $\to_{\mathcal{C}}$ et $\to_{\mathcal{U}}$ sont tout deux convergents. De plus, pour tout $t \in \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$, la forme $\mathcal{C}$-normale de t est $t_{\mathcal{C}}$.

Proposition 8.3.2 *La fonction qui à* $t \in \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$ *associe sa forme* $\mathcal{C}$*-normale est la fonction de currysification de* $(\_)_{\mathcal{C}} : f \in \Sigma \mapsto f_0 \in \mathsf{PP}(\Sigma)_0$.

Preuve. On montre que pour tout $t \in \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$, la forme $\mathcal{C}$-normale de t est $t_{\mathcal{C}}$. On raisonne par induction sur t. Selon le lemme 3.2.6, on est dans un des cas suivants :

$t = x \cdot t_1 \ldots t_n$. Dans ce cas, la forme $\to_{\mathcal{C}}$-normale de t est $x t'_1 \ldots t'_n$ où pour tout $i \in \{1, \ldots, n\}$, t'_i est la forme $\mathcal{C}$-normale de t_i. Par hypothèse d'induction on a $t'_i = (t_i)_{\mathcal{C}}$, et comme $t_{\mathcal{C}} = x \cdot (t_1)_{\mathcal{C}} \ldots (t_n)_{\mathcal{C}}$, il s'en suit que $t_{\mathcal{C}}$ est la forme $\mathcal{C}$-normale de t.

$t = f_n(t_1, \ldots, t_n) \cdot t_{n+1} \ldots t_{n+m}$. La forme $\mathcal{C}$-normale de t est $f_0 t'_1 \ldots t'_{n+m}$, où pour tout $i \in \{1, \ldots, n+m\}$, t'_i est la forme $\mathcal{C}$-normale de t_i. Par hypothèse d'induction on a $t'_i = (t_i)_{\mathcal{C}}$, et comme $t_{\mathcal{C}} = f_0(t_1)_{\mathcal{C}} \ldots (t_{n+m})_{\mathcal{C}}$, il s'en suit que $t_{\mathcal{C}}$ est la forme $\mathcal{C}$-normale de t. □

Il est donc légitime de noter $t_{\mathcal{C}}$ la forme $\mathcal{C}$-normale de t ; de façon analogue, nous écrivons $t_{\mathcal{U}}$ sa forme $\mathcal{U}$-normale. Notons que tout terme $t \in \mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$ est en forme $\mathcal{C}$-normale.

D'autre part, les termes partiellement paramétrés de $\mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$ contiennent les termes curryfiées par $(\_)_{\mathcal{C}} : f \in \Sigma \mapsto f_0 \in \mathsf{PP}(\Sigma)_0$. Nous prouvons la préservation de la confluence pour cette curryfication particulière. Le résultat pour une curryfication quelconque $\Sigma_{\mathcal{C}'}$ suit du fait que $\mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$ et $\mathcal{T}er(\Sigma_{\mathcal{C}'App}, \mathcal{X})$ sont toutes deux des $\Sigma_{\mathcal{C}App}$-algèbres libres sur $\mathcal{X}$.

Remarque 8.3.3 *On a donc deux fonctions* $(\_)_{\mathcal{C}}$ *et* $(\_)_{\mathcal{U}}$ *sur* $\mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$ *qui sont inverses l'une de l'autre : pour tout* $t \in \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$, *on a* $(t_{\mathcal{C}})_{\mathcal{U}} = t = (t_{\mathcal{U}})_{\mathcal{C}}$. *Elles définissent donc des isomorphismes* $(\_)_{\mathcal{C}} \circ (\_)_{\mathcal{U}}$ *et* $(\_)_{\mathcal{U}} \circ (\_)_{\mathcal{C}}$ *sur* $\mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$. *Notons que* $(\_)_{\mathcal{U}} \circ (\_)_{\mathcal{C}}$ *est aussi un isomorphisme sur* $\mathcal{T}er(\Sigma, \mathcal{X})$.

8.3.2 Propriétés de la currification

La currification commute avec les contextes et les substitutions.

Proposition 8.3.4 *Soit* $t \in \mathcal{T}er(\Sigma, \mathcal{X})$. *Alors,*
— *pour tout* $\mathcal{T}er(\Sigma, \mathcal{X})$*-contexte* $C[\ ]$, *on a* $C[t]_{\mathcal{C}} = C_{\mathcal{C}}[t_{\mathcal{C}}]$,
— *pour toute* $\mathcal{T}er(\Sigma, \mathcal{X})$*-substitution* σ, *on a* $(t\sigma)_{\mathcal{C}} = t_{\mathcal{C}}\sigma_{\mathcal{C}}$.

Preuve.

— Par induction sur $C[\ ]$.

$C[\ ] = [\ ]$. Dans ce cas on a $C[t] = t$, donc $C[t]_{\mathcal{C}} = t_{\mathcal{C}}$ et $C_{\mathcal{C}}[\ ] = C[\ ]$.

$C[\ ] = f(t_1, \ldots, t_{i-1}, D[\ ], t_{i+1}, \ldots, t_n)$. Par hypothèse d'induction on a

$$C[t]_{\mathcal{C}} \quad = \quad f_{\mathcal{C}}\ (t_1)_{\mathcal{C}} \ldots (t_{i-1})_{\mathcal{C}}\ D_{\mathcal{C}}[t_{\mathcal{C}}]\ (t_{i+1})_{\mathcal{C}} \ldots (t_n)_{\mathcal{C}}\ .$$

D'où $C[t]_{\mathcal{C}} = C_{\mathcal{C}}[t_{\mathcal{C}}]$.

— Par induction sur t.

$t = x$. Dans ce cas on a $(t\sigma)_{\mathcal{C}} = \sigma(x)_{\mathcal{C}}$, donc $(t\sigma)_{\mathcal{C}} = t\sigma_{\mathcal{C}}$ car $t_{\mathcal{C}} = t$.

$t = f(t_1, \ldots, t_n)$. Dans ce cas on a $(t\sigma)_{\mathcal{C}} = f((t_1\sigma)_{\mathcal{C}}, \ldots, (t_n\sigma)_{\mathcal{C}})$, soit par hypothèse d'induction, $(t\sigma)_{\mathcal{C}} = f(t_1\sigma_{\mathcal{C}}, \ldots, t_n\sigma_{\mathcal{C}}) = t\sigma_{\mathcal{C}}$. □

La commutation de la currification avec les contextes et substitutions permet d'avoir la projection des pas de réécriture sur les formes $\mathcal{C}$-normales.

Proposition 8.3.5 *Soit* $\mathcal{R}$ *un système conditionnel sur* $\mathcal{T}er(\Sigma, \mathcal{X})$ *et* $X \in \{se, jo, o\}$. *Pour tout* $t, u \in \mathcal{T}er(\Sigma, \mathcal{X})$,

$$t \to_{\mathcal{R}^X} u \qquad \textit{implique} \qquad t_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}}^X} u_{\mathcal{C}}\ .$$

Preuve. Nous détaillons la preuve pour la réécriture par joignabilité. Par induction sur $i \in \mathbb{N}$, montrons que $t \to_{\mathcal{R}_i} u$ implique $t_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}i}} u_{\mathcal{C}}$. Le cas de base $i = 0$ étant trivial, montrons que pour tout $i \in \mathbb{N}$ la propriété est vraie pour $i+1$ dès lors qu'elle l'est pour i.

Si $t \to_{\mathcal{R}_{i+1}} u$, alors il existe un contexte $C[\ ]$, une règle $\vec{d} = \vec{c} \supset l \to r$ et une substitution σ tels que $t = C[l\sigma]$, $u = C[r\sigma]$ et $\vec{d}\sigma \downarrow_{\mathcal{R}_i} \vec{c}\sigma$. Par l'hypothèse d'induction et la proposition 8.3.4, on a $\vec{d}_{\mathcal{C}}\sigma_{\mathcal{C}} \downarrow_{\mathcal{R}_{\mathcal{C}i}} \vec{c}_{\mathcal{C}}\sigma_{\mathcal{C}}$ soit $t_{\mathcal{C}} = C_{\mathcal{C}}[l_{\mathcal{C}}\sigma_{\mathcal{C}}] \to_{\mathcal{R}_{\mathcal{C}i}} C_{\mathcal{C}}[r_{\mathcal{C}}\sigma_{\mathcal{C}}] = u_{\mathcal{C}}$. □

8.3.3 Propriétés de la décurryfication

À la différence de la currification, le décurryfication ne commute pas avec les contextes.

Exemple 8.3.6 *Avec* $f_1(\_) \in \Sigma_1$ *et* $C[\ ] =_{def} [\ ] \cdot x$, *on a*

$$C[f_0]_{\mathcal{U}} \quad = \quad f_1(x) \quad \neq \quad f_0 \cdot x \quad = \quad C_{\mathcal{U}}[(f_0)_{\mathcal{U}}]\ .$$

De ce fait, on ne peut pas, en général, projeter la réécriture sur les formes $\mathcal{U}$-normales.

Exemple 8.3.7 *Considérons une version décurryfiée de la fonction* id *:*

$$\mathsf{id}_1(x) \quad \mapsto_{\mathsf{id}_1} \quad x\ .$$

Alors, en utilisant le contexte $C[\ ] = [\ ] \cdot x$ *de l'exemple 8.3.6, on a*

$$\begin{array}{ccc} \mathsf{id}_1(\mathsf{f}_0) \cdot x & \xrightarrow{\quad\mathsf{id}_1\quad} & \mathsf{f}_0 \cdot x \\ {\scriptstyle \mathcal{U}}\big\downarrow{\scriptstyle *} & & {\scriptstyle *}\big\downarrow{\scriptstyle \mathcal{U}} \\ \mathsf{id}_1(\mathsf{f}_0) \cdot x & & \mathsf{f}_1(x) \end{array}$$

mais $\mathsf{id}_1(\mathsf{f}_0) \cdot x$ *ne se* $\mapsto_{\mathsf{id}_1}$*-réduit pas en* $\mathsf{f}_1(x)$*.*

On ne peut donc pas avoir l'équivalent de la proposition 8.3.4 pour la décurryfication. Toutefois, la décurryfication commute avec les substitutions appliquées aux termes du premier ordre.

Proposition 8.3.8 *Si* $t \in \mathcal{T}er(\Sigma, \mathcal{X})$ *et* σ *est une substitution dans* $\mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$*, alors* $(t\sigma)_{\mathcal{U}} = t\sigma_{\mathcal{U}}$*.*

Preuve. On raisonne par induction sur t.

$t = x \in \mathcal{X}$. Dans ce cas $(t\sigma)_{\mathcal{U}} = \sigma(x)_{\mathcal{U}} = \sigma_{\mathcal{U}}(x)$.

$t = \mathsf{f}_n(t_1, \ldots, t_n)$. On a $(t\sigma)_{\mathcal{U}} = \mathsf{f}_n((t_1\sigma)_{\mathcal{U}}, \ldots, (t_n\sigma)_{\mathcal{U}})$ car $t\sigma = \mathsf{f}_n(t_1\sigma, \ldots, t_n\sigma)$. Il s'en suit par hypothèse d'induction que $(t_i\sigma)_{\mathcal{U}} = t_i\sigma_{\mathcal{U}}$ pour tout $i \in \{1, \ldots, n\}$; donc $(t\sigma)_{\mathcal{U}} = t\sigma_{\mathcal{U}}$. □

De ce fait, la décurryfication commute avec les contextes dans lesquels on a mis un terme de la forme $t\sigma$ avec $t \in \mathcal{T}er(\Sigma, \mathcal{X}) \setminus \mathcal{X}$.

Proposition 8.3.9 *Soit* $t \in \mathcal{T}er(\Sigma, \mathcal{X}) \setminus \mathcal{X}$ *et* $\sigma : \mathcal{X} \to \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$*. Alors,*

(i) pour tout $u_1, \ldots, u_m \in \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$*, on a*

$$(t\sigma\ u_1 \ldots u_m)_{\mathcal{U}} \quad = \quad t\sigma_{\mathcal{U}}\ (u_1)_{\mathcal{U}} \ldots (u_m)_{\mathcal{U}}\ ,$$

(ii) pour tout $C[\ ] : \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X}) \to \mathcal{T}er(\mathsf{PP}(\Sigma), \mathcal{X})$*, on a*

$$C[t\sigma]_{\mathcal{U}} \quad = \quad C_{\mathcal{U}}[t\sigma_{\mathcal{U}}]\ .$$

Preuve.

(i) Comme il n'est pas une variable, t est de la forme $\mathsf{f}_n(t_1, \ldots, t_n)$ avec $\mathsf{f}_n \in \Sigma_n$ et $t_1, \ldots, t_n \in \mathcal{T}er(\Sigma, \mathcal{X})$; donc

$$(t\sigma\ t_1 \ldots t_n)_{\mathcal{U}} \quad = \quad \mathsf{f}((t_1\sigma)_{\mathcal{U}}, \ldots, (t_n\sigma)_{\mathcal{U}})\ (u_1)_{\mathcal{U}} \ldots (u_m)_{\mathcal{U}}\ .$$

Or, par la proposition 8.3.8, on a $(t_i\sigma)_{\mathcal{U}} = t_i\sigma_{\mathcal{U}}$ pour tout $i \in \{1, \ldots, n\}$; d'où le résultat.

(ii) Par induction sur $C[\ ]$, en utilisant le lemme 3.2.6.

$C[\ ] = [\ ]\, t_1 \dots t_n$. Il s'en suit que $C_{\mathcal{U}}[\ ] = [\ ]\, (t_1)_{\mathcal{U}} \dots (t_n)_{\mathcal{U}}$, et comme par (i) on a $C[t\sigma]_{\mathcal{U}} = t\sigma_{\mathcal{U}}\, (t_1)_{\mathcal{U}} \dots (t_n)_{\mathcal{U}}$, on en déduit que $C[t\sigma]_{\mathcal{U}} = C_{\mathcal{U}}[t\sigma_{\mathcal{U}}]$.

$C[\ ] = f_n(t_1, \dots, D[\ ], \dots, t_n)\, t_{n+1} \dots t_{n+m}$. Il existe $k \in \{0, \dots, m\}$ tel que

$$C_{\mathcal{U}}[\ ] \quad = \quad f_{n+k}((t_1)_{\mathcal{U}}, \dots, D_{\mathcal{U}}[\ ], \dots, (t_{n+k})_{\mathcal{U}})\ (t_{n+k+1})_{\mathcal{U}} \dots (t_{n+m})_{\mathcal{U}}\ .$$

Or, par hypothèse d'induction,

$$C[t\sigma]_{\mathcal{U}} \quad = \quad f_{n+k}((t_1)_{\mathcal{U}}, \dots, D_{\mathcal{U}}[t\sigma_{\mathcal{U}}], \dots, (t_{n+k})_{\mathcal{U}})\ (t_{n+k+1})_{\mathcal{U}} \dots (t_{n+m})_{\mathcal{U}}\ .$$

D'où le résultat.

$C[\ ] = f_n(t_1, \dots, t_n)\, t_{n+1} \dots D[\ ] \dots t_{n+m}$. Il existe $k \in \{0, \dots, m\}$ tel que

$$C_{\mathcal{U}}[\ ] \quad = \quad f_{n+k}((t_1)_{\mathcal{U}}, \dots, (t_{n+k})_{\mathcal{U}})\ (t_{n+k+1})_{\mathcal{U}} \dots D_{\mathcal{U}}[\ ] \dots (t_{n+m})_{\mathcal{U}}$$

ou bien

$$C_{\mathcal{U}}[\ ] \quad = \quad f_{n+k}((t_1)_{\mathcal{U}}, \dots, D_{\mathcal{U}}[\ ], \dots, (t_{n+k})_{\mathcal{U}})\ (t_{n+k+1})_{\mathcal{U}} \dots (t_{n+m})_{\mathcal{U}}\ .$$

Or par hypothèse d'induction, on a respectivement

$$C[t\sigma]_{\mathcal{U}} \quad = \quad f_{n+k}((t_1)_{\mathcal{U}}, \dots, (t_{n+k})_{\mathcal{U}})\ (t_{n+k+1})_{\mathcal{U}} \dots D_{\mathcal{U}}[t\sigma_{\mathcal{U}}] \dots (t_{n+m})_{\mathcal{U}}$$

et

$$C[t\sigma]_{\mathcal{U}} \quad = \quad f_{n+k}((t_1)_{\mathcal{U}}, \dots, D_{\mathcal{U}}[t\sigma_{\mathcal{U}}], \dots, (t_{n+k})_{\mathcal{U}})(t_{n+k+1})_{\mathcal{U}} \dots (t_{n+m})_{\mathcal{U}}\ .$$

D'où le résultat. □

8.4 Préservation de la confluence

Dans cette section, nous montrons que la confluence est préservée par curryfication pour les systèmes conditionnels non-effondrants. Nous utilisons deux résultats de [Kah95]. Le premier dit que la confluence de $\to_{PP(R)}$ suit de la confluence de $\to_R$.

Théorème 8.4.1 (Théorème 5.1 dans [Kah95]) *Soit R un TRS sur $\mathcal{T}er(\Sigma, \mathcal{X})$. Si $(\mathcal{T}er(\Sigma, \mathcal{X}), \to_R)$ est confluent, alors $(\mathcal{T}er(PP(\Sigma), \mathcal{X}), \to_{PP(R)})$ l'est aussi.*

Remarque 8.4.2 *Le théorème 8.4.1, dont la preuve dans [Kah95] n'est pas triviale, énonce la confluence de la combinaison hiérarchique des systèmes confluents $\to_R$ et $\to_{\mathcal{U}}$. On peut penser que la difficulté vient du fait que les combinaisons hiérarchiques ne préservent en général pas la confluence (voir l'exemple 5.3.10).*

Pour déduire du théorème 8.4.1 la confluence de $\to_{\mathcal{R}_C^X}$, on utilise la propriété suivante des ARS, elle aussi proposée par Kahrs.

Proposition 8.4.3 (Proposition 2.1 dans [Kah95]) *Soient deux ARS $(A, \to_A)$ et $(B, \to_B)$ tels que $(B, \to_B)$ est confluent. S'il existe deux fonctions $F : A \to B$ et $G : B \to A$ telles que*

$$\textit{pour tout } t, u \in A, \quad t \to_A u \quad \textit{implique} \quad F(t) \leftrightarrow_B^* F(u)\ , \tag{a}$$

$$\textit{pour tout } t \in A, u' \in B, \quad F(t) \to_B^* u' \quad \textit{implique} \quad t \to_A^* G(u')\ ; \tag{b}$$

alors $(A, \to)$ est confluent.

Nous utilisons le théorème 8.4.1 et la proposition 8.4.3 de manière similaire à [Kah95]. Soit un ensemble $\mathcal{R}$ de règles conditionnelles sur $\mathcal{T}er(\Sigma,\mathcal{X})$ et $X \in \{se, jo, o\}$. On note $(\mathcal{T}er(PP(\Sigma),\mathcal{X}), PP)$ la paramétrisation partielle de $(\mathcal{T}er(\Sigma,\mathcal{X}), \to_{\mathcal{R}^X})$. Pour obtenir la préservation de la confluence sur les termes applicatifs, nous utilisons la proposition 8.4.3 avec les ARS $(\mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X}), \to_{\mathcal{R}_{\mathcal{C}}^X})$ et $(\mathcal{T}er(PP(\Sigma),\mathcal{X}), \to_{PP})$ et les fonctions

$$\begin{aligned} F : \mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X}) &\to \mathcal{T}er(PP(\Sigma),\mathcal{X}) \\ &\text{définie par} \quad F(t) =_{def} t \qquad \text{pour tout } t \in \mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X}) \ , \\ G : \mathcal{T}er(PP(\Sigma),\mathcal{X}) &\to \mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X}) \\ &\text{définie par} \quad G(t') =_{def} t'_{\mathcal{C}} \quad \text{pour tout } t' \in \mathcal{T}er(PP(\Sigma),\mathcal{X}) \ . \end{aligned}$$

Pour pouvoir appliquer la proposition 8.4.3 aux fonctions F et G, il faut donc vérifier les deux conditions suivantes : pour tout $t, u \in \mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X})$ et $u' \in \mathcal{T}er(PP(\Sigma),\mathcal{X})$,

$$t \to_{\mathcal{R}_{\mathcal{C}}^X} u \quad \text{implique} \quad t \leftrightarrow_{PP}^* u \ , \tag{a}$$

$$t \to_{PP}^* u' \quad \text{implique} \quad t \to_{\mathcal{R}_{\mathcal{C}}^X}^* u'_{\mathcal{C}} \ . \tag{b}$$

On note $\to_{\overline{\mathcal{R}}^X}$ la plus petite relation de réécriture sur $\mathcal{T}er(PP(\Sigma),\mathcal{X})$ contenant $\to_{\mathcal{R}^X}$. De ce fait, $\to_{PP} = \to_{\overline{\mathcal{R}}^X} \cup \to_{\mathcal{U}}$. La relation de réécriture $\to_{\overline{\mathcal{R}}^X}$ est donc la clôture de $\to_{\mathcal{R}^X}$ par $\mathcal{T}er(PP(\Sigma),\mathcal{X})$-contextes et par $\mathcal{T}er(PP(\Sigma),\mathcal{X})$-substitutions : $t \to_{\overline{\mathcal{R}}^X} u$ si et seulement si

$$\begin{aligned} &\text{il existe} \quad C[\ ] : \mathcal{T}er(PP(\Sigma),\mathcal{X}) \to \mathcal{T}er(PP(\Sigma),\mathcal{X}), \ \sigma : \mathcal{X} \to \mathcal{T}er(PP(\Sigma),\mathcal{X}) \\ &\qquad \text{et} \quad t', u' \in \mathcal{T}er(\Sigma,\mathcal{X}) \\ &\text{tels que} \quad t = C[t'\sigma] \quad \text{et} \quad t' \to_{\mathcal{R}^X} u' \quad \text{et} \quad C[u'\sigma] = u \ . \end{aligned}$$

Bien que la décurryfication ne commute pas avec la réécriture (exemple 8.3.7), pour la condition (a) on peut, dans le cas de la réécriture non-conditionnelle, procéder de la manière suivante [Kah95] :

$$t \quad \to_{\mathcal{R}_{\mathcal{C}}} \quad u \qquad \text{implique} \qquad t \quad \to_{\mathcal{U}}^* \quad t_{\mathcal{U}} \quad \to_{\overline{\mathcal{R}}} \quad u' \quad \leftarrow_{\mathcal{U}}^* \quad u \ ,$$

avec en général $u' \neq u_{\mathcal{U}}$. Dans le cas de la réécriture conditionnelle, les choses sont plus compliquées. Essayons de montrer (a) par induction sur la stratification de la réécriture conditionnelle par joignabilité, en utilisant l'hypothèse d'induction que

$$t \quad \to_{\mathcal{R}_{\mathcal{C}i}^{jo}} \quad u \qquad \text{implique} \qquad t \quad \to_{\mathcal{U}}^* \quad t_{\mathcal{U}} \quad \to_{\overline{\mathcal{R}}^{jo}} \quad u' \quad \leftarrow_{\mathcal{U}}^* \quad u \ .$$

Soit une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et supposons que

$$l_{\mathcal{C}}\sigma \quad \to_{\mathcal{R}_{\mathcal{C}i+1}^{jo}} \quad r_{\mathcal{C}}\sigma \qquad \text{avec} \qquad \vec{d}_{\mathcal{C}}\sigma \quad \to_{\mathcal{R}_{\mathcal{C}i}^{jo}}^* \quad \vec{v} \quad \leftarrow_{\mathcal{R}_{\mathcal{C}i}^{jo}}^* \quad \vec{c}_{\mathcal{C}}\sigma \ .$$

Alors l'hypothèse d'induction nous permet de conclure qu'il existe des termes $\vec{v}'$ et $\vec{v}''$ tels que

$$\vec{d}\sigma_{\mathcal{U}} \quad \to_{\overline{\mathcal{R}}^{jo}}^* \quad \vec{v}' \quad \leftarrow_{\mathcal{U}}^* \quad \vec{v} \quad \to_{\mathcal{U}}^* \quad \vec{v}'' \quad \leftarrow_{\overline{\mathcal{R}}^{jo}}^* \quad \vec{c}\sigma_{\mathcal{U}}$$

mais cela n'implique pas $\vec{d}\sigma_{\mathcal{U}} \downarrow_{\overline{\mathcal{R}}^{jo}} \vec{c}\sigma_{\mathcal{U}}$. Donc, a priori, il n'est pas clair que nous ayons $l\sigma_{\mathcal{U}} \to_{\overline{\mathcal{R}}^{jo}} r\sigma_{\mathcal{U}}$.

Pour échapper à ce problème, nous nous restreignons à un cas dans lequel les pas de réécriture peuvent être projetés sur les formes $\mathcal{U}$-normales. Cela suppose de pouvoir appliquer la proposition 8.3.9 : dans $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$, ni l ni r ne doivent être des variables, ce qui impose que les règles soient non-effondrantes.

Lemme 8.4.4 (Projection de $\to_{PP}$ sur les formes $\mathcal{U}$- et $\mathcal{C}$-normales) *Soient* t *et* u *dans* $\mathcal{T}er(PP(\Sigma), \mathcal{X})$.

(i) Si $\mathcal{R}$ *est non-effondrant, alors* $t \to_{PP}^* u$ *implique* $t_{\mathcal{U}} \to_{\overline{\mathcal{R}}^x}^* u_{\mathcal{U}}$.

(ii) $t \to_{PP}^* u$ *implique* $t_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}}^x}^* u_{\mathcal{C}}$.

Preuve.

(i) On montre que $t \to_{\overline{\mathcal{R}}^x} u$ implique $t_{\mathcal{U}} \to_{\overline{\mathcal{R}}^x} u_{\mathcal{U}}$. Comme $\to_{PP} = \to_{\overline{\mathcal{R}}^x} \cup \to_{\mathcal{U}}$, il s'en suit que $t \to_{PP} u$ implique $t_{\mathcal{U}} \to_{\overline{\mathcal{R}}^x}^= u_{\mathcal{U}}$, d'où le résultat.

Si $t \to_{\overline{\mathcal{R}}^x} u$, alors il existe un contexte $C[\,] : \mathcal{T}er(PP(\Sigma), \mathcal{X}) \to \mathcal{T}er(PP(\Sigma), \mathcal{X})$, une substitution $\sigma : \mathcal{X} \to \mathcal{T}er(PP(\Sigma), \mathcal{X})$ et deux termes $t', u' \in \mathcal{T}er(\Sigma, \mathcal{X})$ tels que $t = C[t'\sigma]$, $u = C[u'\sigma]$ et $t' \to_{\mathcal{R}^x} u'$. Comme $\mathcal{R}$ est non-effondrant, ni t' ni u' n'est une variable, donc par la proposition 8.3.9.(ii), on a $t_{\mathcal{U}} = C_{\mathcal{U}}[t'\sigma_{\mathcal{U}}]$ et $u_{\mathcal{U}} = C_{\mathcal{U}}[u'\sigma_{\mathcal{U}}]$, soit $t_{\mathcal{U}} \to_{\overline{\mathcal{R}}^x} u_{\mathcal{U}}$.

(ii) On montre que $t \to_{\overline{\mathcal{R}}^x} u$ implique $t_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}}^x} u_{\mathcal{C}}$. Comme $\to_{PP} = \to_{\overline{\mathcal{R}}^x} \cup \to_{\mathcal{U}}$, il s'en suit que $t \to_{PP} u$ implique $t_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}}^x}^= u_{\mathcal{C}}$, d'où le résultat.

Si $t \to_{\overline{\mathcal{R}}^x} u$ alors il existe donc un contexte $C[\,]$, une substitution σ et deux termes $t', u' \in \mathcal{T}er(\Sigma, \mathcal{X})$ tels que $t = C[t'\sigma]$, $u' = C[u'\sigma]$ et $t' \to_{\mathcal{R}^x} u'$. Par la proposition 8.3.5, on a $t'_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}}^x} u'_{\mathcal{C}}$. De plus, par la proposition 8.3.4, on a $t_{\mathcal{C}} = C_{\mathcal{C}}[t'_{\mathcal{C}}\sigma_{\mathcal{C}}]$ et $u_{\mathcal{C}} = C_{\mathcal{C}}[u'_{\mathcal{C}}\sigma_{\mathcal{C}}]$. On en déduit que $t_{\mathcal{C}} \to_{\mathcal{R}_{\mathcal{C}}^x} u_{\mathcal{C}}$. □

Théorème 8.4.5 (Préservation de la confluence sur les termes applicatifs) *Soit* $\mathcal{R}$ *un ensemble de règles conditionnelles non-effondrantes sur* $\mathcal{T}er(\Sigma, \mathcal{X})$ *et* $X \in \{se, jo, o\}$. *Si* $\to_{\mathcal{R}^x}$ *est confluent sur* $\mathcal{T}er(\Sigma, \mathcal{X})$ *alors* $\to_{\mathcal{R}_{\mathcal{C}}^x}$ *est confluent sur* $\mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$.

Preuve. Par le théorème 8.4.1, on a la confluence de $\to_{PP}$. On applique la proposition 8.4.3 avec les fonctions F et G définies ci-dessus. On vérifie les conditions (a) et (b).

Condition (a). On doit montrer que pour tout $t, u \in \mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$,

$$t \to_{\mathcal{R}_{\mathcal{C}}^x} u \qquad \text{implique} \qquad t \leftrightarrow_{PP}^* u\ .$$

On montre par induction sur $i \in \mathbb{N}$ que $t \to_{\mathcal{R}_{\mathcal{C}i}^x} u$ implique $t \leftrightarrow_{PP}^* u$. Le cas de base $i = 0$ est trivial car $\to_{\mathcal{R}_{\mathcal{C}0}^x} = \emptyset$. Supposons la propriété vraie pour $i \geq 0$ et montrons la pour $i + 1$.

Si $t \to_{\mathcal{R}_{\mathcal{C}i+1}^x} u$, alors il existe

— un contexte $C[\,]$,

— une substitution σ,
— une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$,
— des termes $\vec{v} \in \mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$,
tels que

$$t = C[l_{\mathcal{C}}\sigma], \qquad u = C[r_{\mathcal{C}}\sigma] \qquad \text{et} \qquad \vec{d}_{\mathcal{C}}\sigma \ \to^*_{\mathcal{R}^X_{\mathcal{C}i}} \ \vec{v} \ \leftarrow^*_{\mathcal{R}^X_{\mathcal{C}i}} \ \vec{c}_{\mathcal{C}}\sigma \ .$$

Par hypothèse d'induction, on a $\vec{d}_{\mathcal{C}}\sigma \leftrightarrow^*_{PP} \vec{c}_{\mathcal{C}}\sigma$. Comme $\vec{d}_{\mathcal{C}}$ est la forme $\mathcal{C}$-normale de $\vec{d}$ (proposition 8.3.2), on a $\vec{d}_{\mathcal{C}} \to^*_{\mathcal{U}} \vec{d}$, donc $\vec{d}_{\mathcal{C}}\sigma \to^*_{\mathcal{U}} \vec{d}\sigma$. En appliquant le même raisonnement à $\vec{c}$, on obtient $\vec{d}\sigma \leftrightarrow^*_{PP} \vec{c}\sigma$. La relation $\to_{PP}$ étant confluente (théorème 8.4.1), il existe $\vec{w}$ tels que $\vec{d}\sigma \to^*_{PP} \vec{w} \leftarrow^*_{PP} \vec{c}\sigma$, soit, par le lemme 8.4.4.(i),

$$(\vec{d}\sigma)_{\mathcal{U}} \quad \to^*_{\overline{\mathcal{R}}^X} \quad \vec{w}_{\mathcal{U}} \quad \leftarrow^*_{\overline{\mathcal{R}}^X} \quad (\vec{c}\sigma)_{\mathcal{U}} \ .$$

Or, par la proposition 8.3.8, on a

$$(\vec{d}\sigma)_{\mathcal{U}} = \vec{d}\sigma_{\mathcal{U}}, \qquad (\vec{c}\sigma)_{\mathcal{U}} = \vec{c}\sigma_{\mathcal{U}}, \qquad (r\sigma)_{\mathcal{U}} = r\sigma_{\mathcal{U}} \qquad \text{et} \qquad (l\sigma)_{\mathcal{U}} = l\sigma_{\mathcal{U}} \ .$$

On en déduit que

$$t = C[l_{\mathcal{C}}\sigma] \quad \to^*_{\mathcal{U}} \quad C[l\sigma_{\mathcal{U}}] \quad \to_{\overline{\mathcal{R}}^X} \quad C[r\sigma_{\mathcal{U}}] \quad \leftarrow^*_{\mathcal{U}} \quad C[r_{\mathcal{C}}\sigma] = u \ ,$$

c'est-à-dire $t \leftrightarrow^*_{PP} u$.

Condition (b). On montre que pour tout $t \in \mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$ et tout $u \in \mathcal{T}er(PP(\Sigma))$,

$$t \to^*_{PP} u \qquad \text{implique} \qquad t \to^*_{\mathcal{R}^X_{\mathcal{C}}} u_{\mathcal{C}} \ .$$

Or, par le lemme 8.4.4.(ii), pour tout $t, u \in \mathcal{T}er(PP(\Sigma), \mathcal{X})$, si $t \to^*_{PP} u$ alors $t_{\mathcal{C}} \to^*_{\mathcal{R}^X_{\mathcal{C}}} u_{\mathcal{C}}$. Dans le cas où $t \in \mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$, on a $t = t_{\mathcal{C}}$; donc $t \to^*_{\mathcal{R}^X_{\mathcal{C}}} u_{\mathcal{C}}$. □

En utilisant le théorème 5.3.4, on en déduit aisément que la confluence est préservée par curryfication de $\mathcal{T}er(\Sigma, \mathcal{X})$ à $\Lambda(\Sigma_{\mathcal{C}})$.

Corollaire 8.4.6 (Préservation de la confluence sur les lambda-termes) *Soit $\mathcal{R}$ un ensemble de règles conditionnelles non-effondrantes sur $\mathcal{T}er(\Sigma, \mathcal{X})$ et $X \in \{se, jo\}$. Si $\to_{\mathcal{R}^X}$ est confluent sur $\mathcal{T}er(\Sigma, \mathcal{X})$ alors $\to_{\mathcal{R}^X_{\mathcal{C}}}$ est confluent sur $\Lambda(\Sigma_{\mathcal{C}})$.*

Rappelons que si Σ est une signature, alors $\Sigma_{App} =_{def} \Sigma \uplus \{\_ \cdot \_\}$ et $\Sigma^{App} =_{def} \Sigma \backslash \{\_ \cdot \_\}$, de telle sorte que $(\Sigma_{App})^{App} = \Sigma$ (voir la section 5.3.2).

Preuve. Le théorème 8.4.5 implique que $\to_{\mathcal{R}^X_{\mathcal{C}\mathcal{T}er}}$ est confluent sur $\mathcal{T}er(\Sigma_{\mathcal{C}App}, \mathcal{X})$, et on en déduit que $\to_{\mathcal{R}^X_{\mathcal{C}\Lambda}}$ est confluent sur $\Lambda((\Sigma_{\mathcal{C}App})^{App})$ par le théorème 5.3.4. On a le résultat recherché car $(\Sigma_{\mathcal{C}App})^{App} = \Sigma_{\mathcal{C}}$. □

Troisième partie

Normalisation forte et réductibilité

Chapitre 9

Normalisation forte dans le lambda-calcul typé

Ce chapitre présente certains aspects des preuves par réductibilité de la normalisation forte de λ-calculs typés.

L'idée centrale de ces preuves est d'associer à chaque type un ensemble de termes fortement normalisants et de montrer que cette interprétation est adéquate vis-à-vis du typage : chaque terme typé appartient à l'interprétation de son type. Afin que cela soit possible, l'interprétation des types doit respecter certaines propriétés.

Il existe plusieurs interprétations possibles, nous abordons trois d'entre elles : les ensembles saturés de Tait, les candidats de réductibilité de Girard, et les ensembles clos par biorthogonalité.

Chacune d'elles a ses propres caractéristiques. Dans le cas de calculs dont les règles de réduction correspondent à des coupures du système de type, comme par exemple avec le λ-calcul pur ou avec produits, les propriétés mises en valeur par les ensembles de Tait sont très proches des règles de typage, et sûrement parmi les plus simples que l'on puisse avoir. En contrepartie, cette interprétation ne s'adapte pas de manière élégante aux réductions qui ne correspondent pas forcément aux coupures, comme c'est le cas en général avec la réécriture.

Les candidats de réductibilité de Girard, quand à eux, paraissent être un peu plus loin du système de types. Ils ont l'avantage d'avoir une formulation plus abstraite, qui s'adapte très bien à la réécriture.

Les biorthogonaux reposent sur une méthode différente des deux autres familles considérées. L'idée est de caractériser un type par un ensemble de contextes tel que chaque terme de ce type se comporte d'une manière déterminée avec chaque contexte de l'ensemble. Cette manière de faire permet de capturer des propriétés qui semblent plus difficiles à manier avec les autres approches. C'est en particulier le cas des interprétations calculatoires de la déduction naturelle classique. Mais les biorthogonaux sont aussi très utiles dans d'autres cas. Par exemple nous les utilisons au chapitre 10 afin de mieux comprendre l'impact des types unions sur la normalisation forte.

Les résultats que nous présentons dans ce chapitre sont pour la plupart déjà connus. Nous nous sommes attachés à les présenter de manière précise, dans l'objectif de faire ressortir les propriétés importantes utilisées de manière explicite et implicite dans toutes les familles d'interprétations que nous considérons.

Pour ce faire, nous avons adopté une présentation basée sur les *contextes d'élimination*.

Ces contextes sont utilisés dans [Abe04, Mat05] pour formuler de manière élégante les propriétés sur lesquelles reposent les preuves de [Luo90, Alt93] pour la normalisation forte d'extensions du calcul des constructions. Les contextes d'élimination permettent de donner un traitement uniforme aux ensembles saturés, candidats de réductibilité et biorthogonaux. Ceci nous sera utile au chapitre 10 pour étudier la stabilité part union de familles de réductibilité. Notons que nous donnons un principe de définition des termes neutres qui semble être original (voir définition 9.3.13).

Le chapitre se déroule comme suit. Nous commençons, à la section 9.1, par présenter de façon détaillée une preuve de normalisation forte du λ-calcul simplement typé, pur (section 9.1.2) et avec produits (section 9.1.3). Ensuite (section 9.1.4) nous considérons le cas de la réécriture. Nous ne donnons pas de critère de terminaison, mais essayons de voir comment s'adaptent à la réécriture les principes dégagés en 9.1.2 et 9.1.3.

Nous abordons ensuite la normalisation forte du système F. La section 9.2 est consacrée aux problèmes logiques spécifiques à la normalisation de ce calcul, qui comporte une difficulté particulière car elle implique la cohérence de l'arithmétique du second ordre. Nous essayons d'approcher cette difficulté logique en présentant ce que pourrait être une preuve naïve (et fausse). Nous présentons ensuite certains principes importants que satisfont ses preuves de normalisation correctes, que nous appliquons à la section 9.3 aux ensembles saturés, candidats de réductibilité et biorthogonaux.

Enfin, nous discutons brièvement à la section 9.4 la préservation de la normalisation forte pour la combinaison de la réécriture du premier ordre au système F.

9.1 Lambda-calculs avec types simples

Dans cette section, nous présentons les idées importantes des preuves de normalisation par réductibilité pour des calculs simples. Nous nous concentrons sur des systèmes issus du λ-calcul simplement typé, auquel on ajoute des constructeurs de type et des règles de réécriture. La définition générale de tels systèmes se trouve à la section 3.6.

Nous considérons le système $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}, \Sigma, \tau)$, équipé d'un système de réécriture $\mathcal{R}$ sur $\Lambda(\Sigma)$. L'idée des preuves par réductibilité est d'associer à chaque type $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B})$ un ensemble de termes fortement $\beta\mathcal{R}$-normalisants $[\![T]\!] \subseteq \mathcal{SN}_{\beta\mathcal{R}}$, de telle sorte que

$$\text{si } \Gamma \vdash_{\Rightarrow\tau} t : T \text{ et } \sigma(x) \in [\![U]\!] \text{ pour tout } (x : U) \in \Gamma, \text{ alors } t\sigma \in [\![T]\!]. \tag{9.1}$$

Définition 9.1.1 *Soit $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}, \Sigma, \tau)$ un système de types avec constantes et $\mathcal{R}$ un TRS sur $\Lambda(\Sigma)$.*

(i) Une interprétation *de $(\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}, \Sigma, \tau), \mathcal{R})$ est une fonction $[\![\_]\!] : \mathcal{T}_{\Rightarrow}(\mathcal{B}) \to \mathcal{P}(\Lambda(\Sigma))$.*

(ii) Étant donnée une interprétation $[\![\_]\!]$, on dit qu'une substitution $\sigma : \mathcal{X} \to \Lambda(\Sigma)$ valide un contexte Γ dans $[\![\_]\!]$, *notation $\sigma \models_{[\![\_]\!]} \Gamma$, lorsque $\mathcal{D}om(\Gamma) \subseteq \mathcal{D}om(\sigma)$ et $\sigma(x) \in [\![\Gamma(x)]\!]$ pour tout $x \in \mathcal{D}om(\Gamma)$.*

(iii) Une interprétation $[\![\_]\!]$ est valide *si $\mathcal{X} \subseteq [\![T]\!] \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B})$.*

(iv) Une interprétation $[\![\_]\!]$ est adéquate *si*

$$\big(\Gamma \vdash_{\Rightarrow\tau} t : T \quad \textit{et} \quad \sigma \models_{[\![\_]\!]} \Gamma\big) \quad \Longrightarrow \quad t\sigma \in [\![T]\!]\ .$$

Lorsque le contexte le permet, nous désignons $\sigma \models_{[\![\_]\!]} \Gamma$ par $\sigma \models \Gamma$.

Si l'on dispose d'une interprétation valide et adéquate, alors tous les termes typables sont fortement normalisants.

Théorème 9.1.2 (Normalisation forte) *Soit $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}, \Sigma, \tau)$ un système de types avec constantes et $\mathcal{R}$ un TRS sur $\Lambda(\Sigma)$. S'il existe une interprétation valide et adéquate de $(\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}, \Sigma, \tau), \mathcal{R})$, alors*

$$\Gamma \vdash_{\Rightarrow\tau} t : T \quad \Longrightarrow \quad t \in \mathcal{SN}_{\beta\mathcal{R}} \ .$$

Preuve. Soit $[\![\_]\!]$ une interprétation valide et adéquate de $(\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}, \Sigma, \tau), \mathcal{R})$ et supposons que $\Gamma \vdash_{\Rightarrow\tau} t : T$. Comme $[\![\_]\!]$ est valide, la substitution $\sigma =_{\text{def}} [x/x \mid x \in \mathcal{D}om(\Gamma)]$ valide Γ dans $[\![\_]\!]$, et comme $[\![\_]\!]$ est adéquate, on a $t = t\sigma \in [\![T]\!]$, soit $t \in \mathcal{SN}_{\beta\mathcal{R}}$ car $[\![\_]\!]$ est valide. □

9.1.1 Lambda-calcul avec produits

Dans cette section, nous nous plaçons dans un sous-système du λ-calcul simplement typé avec produits $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B})$ présenté à la section 3.7.1. Nous supposons que tous les types de base ont une arité nulle. De ce fait, on est dans un système de la forme $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$. De plus, nous ne prenons pas en compte la règle $\mapsto_{SP}$.

Les termes de ce système sont donc

$$t, u \in \Lambda(\Sigma_\pi) \quad ::= \quad x \ \mid \ tu \ \mid \ \lambda x.t \ \mid \ (t,u) \ \mid \ \pi_1 t \ \mid \ \pi_2 t \ ,$$

où $x \in \mathcal{X}$; et ses types sont

$$T, U \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0) \quad ::= \quad B \ \mid \ U \Rightarrow T \ \mid \ T \times U \ ,$$

où $B \in \mathcal{B}_0$. Rappelons ses règles de typage :

$$(\mathrm{Ax})\ \frac{}{\Gamma, x : T \vdash x : T}$$

$$(\Rightarrow\mathrm{I})\ \frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \Rightarrow T} \qquad (\Rightarrow\mathrm{E})\ \frac{\Gamma \vdash t : U \Rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

$$(\times\mathrm{I})\ \frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash (t_1, t_2) : T_1 \times T_2} \qquad (\times\mathrm{E})\ \frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash \pi_i t : T_i}\ (i \in \{1,2\})$$

Les règles de réduction que nous considérons sont les suivantes :

$$(\lambda x.t)u \quad \mapsto_\beta \quad t[u/x] \qquad\qquad \pi_1(u,v) \quad \mapsto_\pi \quad u \qquad\qquad \pi_2(u,v) \quad \mapsto_\pi \quad v \ .$$

L'interprétation des types est construite inductivement à partir de l'interprétation des types de base

$$\forall B \in \mathcal{B}_0. \quad [\![B]\!] \quad = \quad \mathcal{SN}_{\beta\pi} \ , \tag{9.2}$$

en utilisant une interprétation des constructeurs de types $\_ \Rightarrow \_$ et $\_ \times \_$. Pour cela, il faut donc leur associer des fonctions $\_ \Rightarrow \_, \_ \times \_ : \mathcal{P}(\Lambda(\Sigma_\pi))^2 \to \mathcal{P}(\Lambda(\Sigma_\pi))$. Nous allons voir quelles propriétés on peut demander à $\_ \Rightarrow \_$ et $\_ \times \_$ pour obtenir une interprétation valide et adéquate.

Mécanismes de base

Commençons par rappeler quelques mécanismes de base de la réductibilité. Nous nous basons sur un système très simple dans lequel ne figurent que des produits, et où les règles de typage n'ont pas de contexte.

Les règles

$$(\times E_0)\ \frac{t : T_1 \times T_2}{\pi_i t : T_i}\ (i \in \{1,2\}) \qquad\qquad (\times I_0)\ \frac{t_1 : T_1 \qquad t_2 : T_2}{(t_1, t_2) : T_1 \times T_2}$$

imposent que

$$\text{si } t \in [\![T_1]\!] \times [\![T_2]\!] \text{ alors } \pi_1 t \in [\![T_1]\!] \text{ et } \pi_2 t \in [\![T_2]\!], \qquad (\times E_0)$$
$$\text{si } t_1 \in [\![T_1]\!] \text{ et } t_2 \in [\![T_2]\!] \text{ alors } (t_1, t_2) \in [\![T_1]\!] \times [\![T_2]\!]\,; \qquad (\times I_0)$$

c'est à dire

$$\{(t,u) \mid t \in [\![T_1]\!] \wedge u \in [\![T_2]\!]\} \subseteq [\![T_1]\!] \times [\![T_2]\!] \subseteq \{t \mid \pi_1 t \in [\![T_1]\!] \wedge \pi_2 t \in [\![T_2]\!]\}\,. \quad (9.3)$$

La propriété (9.3) implique que

$$\forall t,u.\quad (t \in [\![T_1]\!] \quad\wedge\quad u \in [\![T_2]\!]) \quad\Longrightarrow\quad (\pi_1(t,u) \in [\![T_1]\!] \quad\wedge\quad \pi_2(t,u) \in [\![T_2]\!])\ .$$

En particulier, par (9.2), il faut que pour tout $T \in \mathcal{T}_\times(\mathcal{B}_0)$,

$$\forall t,u.\quad (t \in [\![T]\!] \wedge u \in \mathcal{SN}_\pi) \quad\Longrightarrow\quad (\pi_1(t,u) \in [\![T]\!] \wedge \pi_2(u,t) \in [\![T]\!])\ . \quad (9.4)$$

Voyons maintenant comment assurer la propriété (9.4) pour tout $T \in \mathcal{T}_\times(\mathcal{B}_0)$. On raisonne par induction sur $T \in \mathcal{T}_\times(\mathcal{B}_0)$, en supposant que $\_ \times \_$ est une fonction quelconque de $\mathcal{P}(\Lambda(\Sigma_\pi))^2$ dans $\mathcal{P}(\Lambda(\Sigma_\pi))$ satisfaisant (9.3).

$T = B \in \mathcal{B}_0$. On doit montrer que

$$\forall t,u \in \mathcal{SN}_\pi.\quad \pi_1(t,u) \in \mathcal{SN}_\pi \quad\wedge\quad \pi_2(t,u) \in \mathcal{SN}_\pi\ . \quad (9.5)$$

$T = T_1 \times T_2$. On doit montrer que pour tout $t \in [\![T_1]\!] \times [\![T_2]\!]$ et tout $u \in \mathcal{SN}_\pi$,

$$\pi_1\pi_1(t,u),\ \pi_1\pi_2(u,t) \in [\![T_1]\!] \quad\wedge\quad \pi_2\pi_1(t,u),\ \pi_2\pi_2(u,t) \in [\![T_2]\!]\ . \quad (9.6)$$

L'induction sur la structure des types impose, à travers (9.6), que (9.4) soit satisfaite pour des termes à l'intérieur de contextes de la forme $\pi_{i_1} \cdots \pi_{i_n} \_$. Ces contextes correspondent, au niveau des termes, aux règles d'*élimination*. On les appelle *contextes d'élimination*. Dans le cas des produits, ils sont générés par la grammaire suivante :

$$E[\,] \in \mathcal{E}_\times \quad ::= \quad [\,] \quad | \quad \pi_1 E[\,] \quad | \quad \pi_2 E[\,]\ .$$

Notons que le « trou » de ces contextes n'est jamais sous un lieur. Par la remarque 1.4.6, ce sont donc des $\mathcal{T}er(\Sigma_\pi, \mathcal{X})$-contextes, qui peuvent être définis par des termes à trous, comme en 1.1.19.

En reformulant (9.4) dans les contextes d'élimination, on aboutit à la propriété suivante : pour tout $T \in \mathcal{T}_\times(\mathcal{B}_0)$ et tout t_1, t_2,

$$\begin{array}{llllll} \forall E[\ ] \in \mathcal{E}_\times. & (t_2 \in \mathcal{SN}_\pi & \wedge & E[t_1] \in [\![T]\!]) & \Longrightarrow & E[\pi_1(t_1,t_2)] \in [\![T]\!]\ , \\ \forall E[\ ] \in \mathcal{E}_\times. & (t_1 \in \mathcal{SN}_\pi & \wedge & E[t_2] \in [\![T]\!]) & \Longrightarrow & E[\pi_2(t_1,t_2)] \in [\![T]\!]\ . \end{array} \tag{9.7}$$

En d'autres termes, l'interprétation des types doit être stable par expansion dans les contextes d'élimination.

Remarque 9.1.3 *La relation* $\{(E[t], E[u]) \mid E[\ ] \in \mathcal{E}_\times \ \wedge\ t \mapsto_\pi u\}$ *est l'analogue, pour les produits, de la* β*-réduction faible de tête définie en 3.2.7.*

9.1.2 Normalisation forte du lambda-calcul simplement typé

Nous allons maintenant montrer la normalisation forte du λ-calcul simplement typé en nous basant sur les intuitions que nous venons de dégager sur les produits. D'autres preuves peuvent être trouvées dans [Kri90, Bar92, GLT89].

Supposons définie une interprétation $[\![\_]\!] : \mathcal{T}_\Rightarrow(\mathcal{B}_0) \to \mathcal{P}(\mathcal{SN}_\beta)$. Considérons les règles

$$(\Rightarrow \mathrm{E})\ \frac{\Gamma \vdash t : U \Rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T} \qquad\qquad (\Rightarrow \mathrm{I})\ \frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \Rightarrow T}$$

En raisonnant comme pour (9.3), on aboutit au fait que $\_ \Rightarrow \_$ doit vérifier, pour toute substitution σ telle que $\sigma \models \Gamma$,

$$\begin{array}{l} \{(\lambda x.t)\sigma \mid \forall u.\ u \in [\![U]\!] \implies t\sigma[u/x] \in [\![T]\!]\} \\ \qquad\qquad \subseteq\ [\![U]\!] \Rightarrow [\![T]\!]\ \subseteq \\ \qquad\qquad\qquad \{t\sigma \mid \forall u.\ u \in [\![U]\!] \implies t\sigma u \in [\![T]\!]\}\ . \end{array} \tag{9.8}$$

Il existe plusieurs définitions de $\_ \Rightarrow \_$ vérifiant (9.8). Nous utilisons des interprétations « basées sur l'élimination », au sens de [Mat98].

Définition 9.1.4 (Interprétation de la flêche) *Soit* Σ *une signature. On définit la fonction* $\_ \Rightarrow \_ : \mathcal{P}(\Lambda(\Sigma)) \times \mathcal{P}(\Lambda(\Sigma)) \to \mathcal{P}(\Lambda(\Sigma))$ *par*

$$A \Rightarrow B \quad =_{def} \quad \{t \mid \forall u. \quad u \in A \quad \Longrightarrow \quad tu \in B\}\ .$$

On peut alors définir l'interprétation des types de $\mathcal{T}_\Rightarrow(\mathcal{B}_0)$ en utilisant, dans le cas de base, l'interprétation $[\![B]\!] = \mathcal{SN}_\beta$ suggérée en (9.2).

Définition 9.1.5 (Interprétation des types) L'interprétation *d'un type* $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$ *est l'ensemble* $[\![T]\!]$ *défini par induction sur* T *de la manière suivante :*

$$\begin{array}{rll} [\![B]\!] & =_{def} & \mathcal{SN}_\beta \qquad\qquad si\ B \in \mathcal{B}_0 \\ [\![U \Rightarrow T]\!] & =_{def} & [\![U]\!] \Rightarrow [\![T]\!]\ . \end{array}$$

Nous allons maintenant vérifier que $[\![\_]\!]$ est bien une interprétation valide et adéquate. Pour l'adéquation, en s'inspirant de ce qu'on a fait avec (9.4), il ressort de (9.8) que pour tout $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B}_0)$,

$$\forall t, u. \quad u \in \mathcal{SN}_\beta \quad \Longrightarrow \quad (t[u/x] \in [\![T]\!] \quad \Longrightarrow \quad (\lambda x.t)u \in [\![T]\!]) \ . \tag{9.9}$$

En ce qui concerne la validité de $[\![\_]\!]$, on doit montrer que pour tout $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B}_0)$ on a $\mathcal{X} \subseteq [\![T]\!]$ et $[\![T]\!] \subseteq \mathcal{SN}_\beta$. En fait, ces deux propriétés ne sont pas indépendantes l'une de l'autre. En effet, $\mathcal{X} \subseteq [\![T]\!]$ implique que $[\![T]\!]$ n'est pas vide, et d'un autre coté on a

$$\emptyset \Rightarrow B \quad = \quad \{t \mid \forall u. \quad u \in \emptyset \quad \Longrightarrow \quad tu \in B\} \quad = \quad \Lambda(\Sigma) \ .$$

D'autre part, pour montrer que $x \in A \Rightarrow B$, il faut montrer que $xu \in B$ pour tout $u \in A$, et si $B \subseteq \mathcal{SN}_\beta$ cela impose que $A \subseteq \mathcal{SN}_\beta$.

Ces propriétés doivent être vérifiées pour les types de base, mais aussi pour l'interprétation de la flèche. Cela nous amène, comme en (9.7), à les formuler en utilisant des contextes d'élimination. Notre utilisation des contextes d'élimination est inspirée de [Abe04, Mat05].

Définition 9.1.6 (Contextes d'élimination des flêches) *Soit Σ une signature. L'ensemble $\mathcal{E}_{\Rightarrow}$ des* contextes d'élimination des flèches *est généré par la grammaire suivante :*

$$E[\ ] \in \mathcal{E}_{\Rightarrow} \quad ::= \quad [\ ] \quad | \quad E[\ ]\, t \ ,$$

où $t \in \Lambda(\Sigma)$.

Ces contextes peuvent donc s'écrire sous la forme $[\ ]t_1 \dots t_n$ avec $n \geq 0$. Notons que dans un contexte $E[\ ] \in \mathcal{E}_{\Rightarrow}$, le « trou » $[\ ]$ n'apparaît jamais sous un lieur, $E[\ ]$ est donc représenté par un unique terme de $\Lambda(\Sigma)$. De plus, comme $E[\ ]$ a une occurrence de $[\ ]$, on a $t \in \mathcal{SN}_\beta$ dès lors que $E[t] \in \mathcal{SN}_\beta$.

Si on formule dans les contextes d'élimination les conditions que $[\![\_]\!]$ doit satisfaire pour être valide et adéquate, on aboutit au fait que pour tout $T \in \mathcal{T}_{\Rightarrow}(\mathcal{B}_0)$, $[\![T]\!]$ doit être une partie de $\mathcal{SN}_\beta$ telle que

— si $E[\ ] \in \mathcal{SN}_\beta$ et $x \in \mathcal{X}$ alors $E[x] \in [\![T]\!]$,

— si $E[t[u/x]] \in [\![T]\!]$ et $u \in \mathcal{SN}_\beta$ alors $E[(\lambda x.t)u] \in [\![T]\!]$.

Ces conditions sont celles des ensembles saturés de Tait, voir [Kri90, Bar92, Gal89].

Définition 9.1.7 (Ensembles saturés) *L'ensemble $\mathbb{SAT}_\beta$ des* ensembles β-saturés *est l'ensemble des $S \subseteq \mathcal{SN}_\beta$ tels que*

($\mathbb{SAT}1$) *si $E[\ ] \in \mathcal{SN}_\beta$ et $x \in \mathcal{X}$ alors $E[x] \in S$,*

($\mathbb{SAT}2$) *si $E[t[u/x]] \in S$ et $u \in \mathcal{SN}_\beta$ alors $E[(\lambda x.t)u]$.*

Nous allons maintenant vérifier que $\mathbb{SAT}_\beta$ n'est pas vide, c'est à dire qu'il existe un ensemble saturé. Pour cela, nous montrons que $\mathcal{SN}_\beta \in \mathbb{SAT}_\beta$.

La condition ($\mathbb{SAT}1$) est immédiate.

Proposition 9.1.8 *Si $E[\ ] \in \mathcal{E}_{\Rightarrow} \cap \mathcal{SN}_\beta$ et $x \in \mathcal{X}$, alors $E[x] \in \mathcal{SN}_\beta$.*

PREUVE. Par induction sur $E[\ ] \in \mathcal{SN}_\beta$, en utilisant la propriété suivante :

$$\forall \nu.\quad E[x] \to_\beta \nu \quad\Longrightarrow\quad \big(\nu = E'[x] \quad \text{avec} \quad E'[\ ] \in \mathcal{E}_\Rightarrow \quad \text{et} \quad E[\ ] \to_\beta E'[\ ]\big)\ . \tag{9.10}$$

□

Pour montrer que (SAT2) est vérifiée par $\mathcal{SN}_\beta$, nous utilisons une propriété simple mais fondamentale du λ-calcul, appelée *standardisation faible* dans [Alt93].

Lemme 9.1.9 (Standardisation faible) *Si* $t \mapsto_\beta u$ *et* $E[t] \to_\beta \nu$ *avec* $\nu \neq E[u]$, *alors* $\nu = E'[t']$ *avec* $(E[\ ], t) \to_\beta (E'[\ ], t')$ *et il existe* u' *tel que* $E[u] \to_\beta^* E'[u']$. *En image :*

$$\begin{array}{ccc} E[t] & \xrightarrow{\ t \mapsto_\beta u\ } & E[u] \\ {\scriptstyle\beta}\downarrow & & \vdots\,{\scriptstyle *}\,{\scriptstyle\beta} \\ E'[t'] & \xrightarrow[\ t' \mapsto_\beta u'\ ]{} & E'[u'] \end{array} \tag{9.11}$$

PREUVE. Comme $E[\ ]$ est de la forme $[\ ]t_1 \dots t_n$, la propriété (9.11) s'écrit sous la forme

$$\begin{array}{ccc} (\lambda x.t)u t_1 \dots t_n & \xrightarrow{\ wh_\beta\ } & t[u/x]t_1 \dots t_n \\ {\scriptstyle\beta}\downarrow & & \vdots\,{\scriptstyle *}\,{\scriptstyle\beta} \\ (\lambda x.t')u' t'_1 \dots t'_n & \xrightarrow[\ wh_\beta\ ]{} & t'[u'/x]t'_1 \dots t'_n \end{array}$$

où $(t, u, t_1, \dots, t_n) \to_\beta (t', u', t'_1, \dots, t'_n)$. Le fait que $t[u/x]t_1 \dots t_n \to_\beta^* t'[u'/x]t'_1 \dots t'_n$ provient du lemme 2.3.1. □

La standardisation faible est tout ce dont on a besoin pour montrer que $\mathcal{SN}_\beta$ vérifie (SAT2).

Lemme 9.1.10 (Stabilité par expansion faible de tête) *Si* $E[t[u/x]] \in \mathcal{SN}_\beta$ *et* $u \in \mathcal{SN}_\beta$ *alors* $E[(\lambda x.t)u] \in \mathcal{SN}_\beta$.

PREUVE. On doit montrer que pour tout ν, si $E[(\lambda x.t)u] \to_\beta \nu$ alors $\nu \in \mathcal{SN}_\beta$. Notons que $E[\ ], t \in \mathcal{SN}_\beta$ car $E[t[u/x]] \in \mathcal{SN}$. On raisonne par induction sur les triplets $(E[\ ], t, u)$ ordonnés par $\to_\beta$.

Soit ν tel que $E[(\lambda x.t)u] \to_\beta \nu$. D'après le lemme 9.1.9 il y a deux possibilités.

— $\nu = E[t[u/x]]$ et $\nu \in \mathcal{SN}_\beta$ par hypothèse,

— $\nu = E'[(\lambda x.t')u']$ avec $(E[\ ], t, u) \to_\beta (E'[\ ], t', u')$ et $E[t[u/x]] \to_\beta^* E'[t'[u'/x]]$. Comme $u, E[t[u/x]] \in \mathcal{SN}_\beta$, on a $u', E'[t'[u'/x]] \in \mathcal{SN}_\beta$. On peut donc appliquer l'hypothèse d'induction à $E'[(\lambda x.t')u'] \to_{wh_\beta} E'[t'[u'/x]]$, d'où $\nu \in \mathcal{SN}_\beta$. □

Remarque 9.1.11 *Dans le cas du λ-calcul pur, la standardisation faible est triviale, et la stabilité de $\mathcal{SN}$ par expansions faibles de tête pourrait être prouvée sans référence explicite à cette propriété. L'intérêt de cette méthode est qu'elle s'adapte bien à différentes extensions du λ-calcul.*

Comme $\mathcal{SN}_\beta$ satisfait ($\mathbb{SAT}1$) et ($\mathbb{SAT}2$), on en déduit que $\mathcal{SN}_\beta \in \mathbb{SAT}_\beta$.

Lemme 9.1.12 *$\mathcal{SN}_\beta \in \mathbb{SAT}_\beta$.*

Pour que $[\![\_]\!]$ soit une interprétation valide, et pour pouvoir montrer qu'elle est adéquate, il nous reste à voir que $[\![T]\!] \in \mathbb{SAT}_\beta$ pour tout $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$. Comme $[\![B]\!] = \mathcal{SN}_\beta$ pour tout $B \in \mathcal{B}_0$, il suffit de montrer que $\_ \Rightarrow \_$ est une fonction de $\mathbb{SAT}_\beta^2$ dans $\mathbb{SAT}_\beta$.

Proposition 9.1.13 *Si $A, B \in \mathbb{SAT}_\beta$ alors $A \Rightarrow B \in \mathbb{SAT}_\beta$.*

Preuve. Montrons tout d'abord que $A \Rightarrow B \subseteq \mathcal{SN}_\beta$. Si $t \in A \Rightarrow B$, alors comme $A \in \mathbb{SAT}_\beta$, on a $tx \in B$ pour tout $x \in \mathcal{X}$, donc $tx \in \mathcal{SN}_\beta$ car $B \in \mathbb{SAT}_\beta$, d'où $t \in \mathcal{SN}_\beta$.

Montrons maintenant que les conditions ($\mathbb{SAT}1$) et ($\mathbb{SAT}2$) sont satisfaites.

($\mathbb{SAT}1$) Si $E[\ ] \in \mathcal{SN}$ et $u \in A$, alors comme $A \in \mathbb{SAT}_\beta$ on a $u \in \mathcal{SN}$, donc $E[x]u \in B$ pour tout $x \in \mathcal{X}$ car $B \in \mathbb{SAT}_\beta$. Il s'en suit que $E[x] \in A \Rightarrow B$ pour tout $x \in \mathcal{X}$.

($\mathbb{SAT}2$) Si $E[t[u/x]] \in A \Rightarrow B$ avec $u \in \mathcal{SN}$, alors pour tout $v \in A$ on a $E[t[u/x]]x \in B$, donc $E[(\lambda x.t)u]v \in B$ car $B \in \mathbb{SAT}_\beta$. Il s'en suit que $E[(\lambda x.t)u] \in A \Rightarrow B$. □

Proposition 9.1.14 *$[\![\_]\!]$ est une interprétation valide.*

Montrons maintenant qu'elle est adéquate.

Lemme 9.1.15 (Adéquation) *Si $\Gamma \vdash_\Rightarrow t : T$ et $\sigma \models \Gamma$, alors $t\sigma \in [\![T]\!]$.*

Preuve. On raisonne par induction sur $\Gamma \vdash_\Rightarrow t : T$.

(Ax)

$$\frac{}{\Gamma, x : T \vdash x : T}$$

On a $\sigma(x) \in [\![T]\!]$ par hypothèse.

($\Rightarrow$I)

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \Rightarrow T}$$

Soit $\sigma \models \Gamma$. On doit montrer que $(\lambda x.t)\sigma \in [\![U]\!] \Rightarrow [\![T]\!]$.

Tout d'abord, par le lemme 1.2.10, on peut supposer que $x \notin FV(\sigma) \cup \mathcal{D}om(\sigma)$. On a donc $(\lambda x.t)\sigma = \lambda x.(t\sigma)$ par le lemme 1.3.9.(i).

Soit maintenant $u \in [\![U]\!]$. Par hypothèse d'induction, on a $t(\sigma[u/x]) \in [\![T]\!]$. Or, comme $x \notin FV(\sigma)$, la proposition 1.3.5 implique que $t(\sigma[u/x]) = (t\sigma)[u/x]$. Comme $[\![T]\!] \in \mathbb{SAT}_\beta$, on a $(\lambda x.t\sigma)u \in [\![T]\!]$ car $u \in [\![U]\!] \subseteq \mathcal{SN}_\beta$ et $t\sigma[u/x] \in [\![T]\!]$. Il s'en suit que $\lambda x.(t\sigma) \in [\![U]\!] \Rightarrow [\![T]\!]$.

($\Rightarrow$E)

$$\frac{\Gamma \vdash t : U \Rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

Par définition de $\_ \Rightarrow \_$. □

Corollaire 9.1.16 *Si $\Gamma \vdash_\Rightarrow t : T$ alors $t \in \mathcal{SN}_\beta$.*

Preuve. On a une interprétation $[\![\_]\!]$ qui est valide par la proposition 9.1.14 et adéquate par le lemme 9.1.15. On peut donc appliquer le théorème 9.1.2, d'où $t \in \mathcal{SN}_\beta$. □

9.1.3 Normalisation forte du lambda-calcul avec produits

Nous allons maintenant adapter au λ-calcul simplement typé avec produits ce que nous venons de faire avec le λ-calcul pur.

Comme pour $\_ \Rightarrow \_$, nous utilisons une interprétation de $\_ \times \_$ basée sur les éliminations.

Définition 9.1.17 (Interprétation des produits) *Soit Σ une signature. On définit la fonction $\_ \times \_ : \mathcal{P}(\Lambda(\Sigma)) \times \mathcal{P}(\Lambda(\Sigma)) \to \mathcal{P}(\Lambda(\Sigma))$ par*

$$A \times B \quad =_{def} \quad \{t \mid \pi_1 t \in A \quad \wedge \quad \pi_2 t \in B\} .$$

On peut alors prolonger naturellement l'interprétation $[\![\_]\!]$ au cas des produits. Notons toutefois que nous avons ici $[\![B]\!] = \mathcal{SN}_{\beta\pi}$.

Définition 9.1.18 (Interprétation des types) L'interprétation *de $T \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ est l'ensemble $[\![T]\!]$ défini par induction sur T de la manière suivante :*

$$\begin{array}{lll} [\![B]\!] & =_{def} & \mathcal{SN}_{\beta\pi} \qquad \text{si } B \in \mathcal{B}_0 \\ [\![U \Rightarrow T]\!] & =_{def} & [\![U]\!] \Rightarrow [\![T]\!] \\ [\![U \times T]\!] & =_{def} & [\![U]\!] \times [\![T]\!] . \end{array}$$

Les contextes d'élimination que nous utilisons sont des combinaisons des contextes d'élimination des flèches et des produits.

Définition 9.1.19 (Contextes d'élimination des flêches et des produits) *L'ensemble $\mathcal{E}_{\Rightarrow\times}$ des* contextes d'élimination des flèches et des produits *est généré par la grammaire suivante :*

$$E[\,] \in \mathcal{E}_{\Rightarrow\times} \quad ::= \quad [\,] \quad | \quad E[\,]\,t \quad | \quad \pi_1 E[\,] \quad | \quad \pi_2 E[\,] ,$$

où $t \in \Lambda(\Sigma)$.

Ils donnent lieu aux ensembles saturés suivants :

Définition 9.1.20 (Ensembles saturés) *L'ensemble $\mathbb{SAT}_{\beta\pi}$ des* ensembles $\beta\pi$-saturés *est l'ensemble des $S \subseteq \mathcal{SN}_{\beta\pi}$ tels que*

($\mathbb{SAT}1$) *pour tout $E[\,] \in \mathcal{E}_{\Rightarrow\times} \cap \mathcal{SN}_{\beta\pi}$ et tout $x \in \mathcal{X}$ on a $E[x] \in S$,*

($\mathbb{SAT}2_{\Rightarrow}$) *pour tout $E[\,] \in \mathcal{E}_{\Rightarrow\times}$, tout $t \in \Lambda(\Sigma_\pi)$ et tout $u \in \mathcal{SN}_{\beta\pi}$, si $E[t[u/x]] \in S$ alors $E[(\lambda x.t)u] \in S$,*

($\mathbb{SAT}2_{\times 1}$) *pour tout $E[\,] \in \mathcal{E}_{\Rightarrow\times}$ et tout $t_1, t_2 \in \Lambda(\Sigma_\pi)$, si $t_2 \in \mathcal{SN}_{\beta\pi}$ et $E[t_1] \in S$ alors $E[\pi_1(t_1, t_2)] \in S$,*

($\mathbb{SAT}2_{\times 2}$) *pour tout $E[\,] \in \mathcal{E}_{\Rightarrow\times}$ et tout $t_1, t_2 \in \Lambda(\Sigma_\pi)$, si $t_1 \in \mathcal{SN}_{\beta\pi}$ et $E[t_2] \in S$ alors $E[\pi_2(t_1, t_2)] \in S$.*

Nous allons maintenant montrer que $\mathcal{SN}_{\beta\pi} \in \mathbb{SAT}_{\beta\pi}$. En ce qui concerne ($\mathbb{SAT}1$), comme la propriété (9.10) est toujours vraie avec les contextes de $\mathcal{E}_{\Rightarrow\times}$, on peut raisonner de la même manière qu'à la proposition 9.1.8.

Proposition 9.1.21 *Si* $E[\ ] \in \mathcal{E}_{\Rightarrow\times} \cap \mathcal{SN}_{\beta\pi}$ *et* $x \in \mathcal{X}$*, alors* $E[x] \in \mathcal{SN}_{\beta\pi}$.

PREUVE. Comme pour la proposition 9.1.8, en utilisant la propriété suivante :

$$\forall \nu.\quad E[x] \to_{\beta\pi} \nu \quad\Longrightarrow\quad \big(\nu = E'[x] \quad \text{avec} \quad E'[\ ] \in \mathcal{E}_{\Rightarrow\times} \quad \text{et} \quad E[\ ] \to_{\beta\pi} E'[\ ]\big)\ . \tag{9.12}$$

□

La standardisation faible est bien plus intéressante pour $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$ que pour $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}_0)$. Nous la montrons au lemme 9.1.23, en utilisant un résultat intermédiaire, qui est une généralisation de (9.12) aux termes ayant un symbole d'élimination en tête. Les *contextes atomiques d'élimination* $\epsilon[\ ]$ sont engendrés par la grammaire

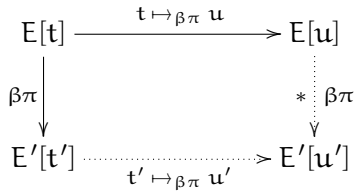
$$\epsilon[\ ] \quad ::= \quad [\ ]\,t \quad | \quad \pi_1[\ ] \quad | \quad \pi_2[\ ]\ .$$

Proposition 9.1.22 *Pour tout* $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$*, tout* $t \in \Lambda(\Sigma_\pi)$ *et tout contexte atomique d'élimination* $\epsilon[\ ]$*, si* $E[\epsilon[t]] \to_{\beta\pi} \nu$*, alors* $\nu = E'[t']$ *avec* $(E[\ ], \epsilon[t]) \to (E'[\ ], t')$.

PREUVE. Par induction sur $E[\ ]$.

- $E[\ ] = [\ ]$. Évident.
- $E[\ ] = F[\ ]u$. Si $F[\epsilon[t]]u \to_{\beta\pi} \nu$, alors, comme $F[\epsilon[t]]$ n'est pas une abstraction, on a forcément $\nu = \nu' u'$ avec $(F[\epsilon[t]], u) \to_{\beta\pi} (\nu', u')$. Si $F[\epsilon[t]] \to_{\beta\pi} \nu'$, alors par hypothèse d'induction $\nu' = F'[t']$ avec $(F[\ ], \epsilon[t]) \to_{\beta\pi} (F'[\ ], t')$, donc $(F[\ ]u, \epsilon[t]) \to_{\beta\pi} (F'u, t')$. Sinon, $u \to_{\beta\pi} u'$ et on a $(F[\ ]u, \epsilon[t]) \to_{\beta\pi} (F[\ ]u', \epsilon[t])$.
- $E[\ ] = \pi_i F[\ ]$. Si $\pi_i F[\epsilon[t]] \to_{\beta\pi} \nu$, alors, comme $F[\epsilon[t]]$ n'est pas une paire, on a forcément $\nu = \pi_i \nu'$ avec $F[\epsilon[t]] \to_{\beta\pi} \nu'$. Par hypothèse d'induction, $\nu' = F'[t']$ avec $(F[\ ], \epsilon[t]) \to_{\beta\pi} (F'[\ ], t')$, donc $(\pi_i F[\ ], \epsilon[t]) \to_{\beta\pi} (\pi_i F'[\ ], t')$. □

Lemme 9.1.23 (Standardisation faible) *Si* $t \mapsto_{\beta\pi} u$ *et* $E[t] \to_{\beta\pi} \nu$ *avec* $\nu \neq E[u]$*, alors* $\nu = E'[t']$ *avec* $(E[\ ], t) \to_{\beta\pi} (E'[\ ], \nu')$ *et il existe un terme* u' *tel que* $t' \mapsto_{\beta\pi} u'$ *et* $E[u] \to^*_{\beta\pi} E'[u']$. *En image :*



PREUVE. Comme $t \mapsto_{\beta\pi} u$, on a $t = \epsilon[t']$. Donc par la proposition 9.1.22, on a $\nu = E'[t'']$ avec $(E[\ ], \epsilon[t']) \to_{\beta\pi} (E'[\ ], t'')$. Si $E[\ ] \to_{\beta\pi} E'[\ ]$ et $\epsilon[t'] = t''$, alors $\nu = E'[t]$,$t \mapsto_{\beta\pi} u$ et $E'[u] \leftarrow_{\beta\pi} E[u]$.

Sinon $\nu = E[\nu']$ avec $t \to_{\beta\pi} \nu'$, et on raisonne par cas sur $t \mapsto_{\beta\pi} u$.

- $t \mapsto_\beta u$ Dans ce cas, $t = (\lambda x.t_1)t_2 \mapsto_\beta t_1[t_2/x] = u$, et $\nu' = (\lambda x.\nu_1)\nu_2$ avec $(t_1, t_2) \to_{\beta\pi} (\nu_1, \nu_2)$. Alors $\nu' \mapsto_\beta \nu_1[\nu_2/x] \leftarrow^*_{\beta\pi} u$ par le lemme 2.3.1.

$t \mapsto_\pi u$ Dans ce cas, $t = \pi_i(t_1, t_2) \mapsto_\pi t_i = u$ et $v' = \pi_i(v_1, v_2)$ avec $(t_1, t_2) \to_{\beta\pi} (v_1, v_2)$. Alors $v' \mapsto_\pi v_i \leftarrow^{=}_{\beta\pi} t_i$. □

Nous allons maintenant montrer que $\mathcal{SN}_{\beta\pi}$ est clos par expansion faible de tête. La formulation de cette propriété avec les produits est un peu plus compliquée que pour le λ-calcul pur.

Lemme 9.1.24 (Stabilité par expansion faible de tête) *Pour tout* $T \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$, *tout* $t_1, t_2 \in \Lambda(\Sigma_\pi)$ *et tout* $E[\,] \in \mathcal{E}_{\Rightarrow\times}$,

$$(t_2 \in \mathcal{SN}_{\beta\pi} \wedge E[t_1] \in \mathcal{SN}_{\beta\pi}) \implies E[\pi_1(t_1, t_2)] \in \mathcal{SN}_{\beta\pi}\ , \qquad (\times_1)$$
$$(t_1 \in \mathcal{SN}_{\beta\pi} \wedge E[t_2] \in \mathcal{SN}_{\beta\pi}) \implies E[\pi_2(t_1, t_2)] \in \mathcal{SN}_{\beta\pi}\ , \qquad (\times_2)$$
$$(t_2 \in \mathcal{SN}_{\beta\pi} \wedge E[t_1[t_2/x]] \in \mathcal{SN}_{\beta\pi}) \implies E[(\lambda x.t_1)t_2] \in \mathcal{SN}_{\beta\pi}\ . \qquad (\Rightarrow)$$

Preuve. On a donc $t \mapsto_{\beta\pi} u$ avec

$$t = \pi_i(t_1, t_2) \quad \text{et} \quad u = t_i\ , \qquad (\times_i)$$
$$t = (\lambda x.t_1)t_2 \quad \text{et} \quad u = t_1[t_2/x]\ . \qquad (\Rightarrow)$$

Dans le cas $(\Rightarrow)$, on a $t_1 \in \mathcal{SN}_{\beta\pi}$ car $t_1[t_2/x] \in \mathcal{SN}_{\beta\pi}$ et toute réduction à partir de t_1 donne lieu à une réduction à partir de $t_1[t_2/x]$ de même longueur, et pour $(\times_i)$, on a $t_1, t_2 \in \mathcal{SN}_{\beta\pi}$ par hypothèse. Donc $t_1, t_2 \in \mathcal{SN}_{\beta\pi}$ dans les deux cas.

On raisonne par induction sur les triplets $(E[\,], t_1, t_2)$, ordonnées par l'extension produit de $\to_{\beta\pi}$. Pour avoir $E[t] \in \mathcal{SN}_{\beta\pi}$, il suffit de montrer que $(E[t])_{\beta\pi} \subseteq \mathcal{SN}_{\beta\pi}$. Soit v tel que $E[t] \to v$. Par le lemme 9.1.23, on a deux possibilités.

(i) $v = E[u] \in \mathcal{SN}_{\beta\pi}$ par hypothèse.

(ii) $v = E'[t']$ avec $(E[\,], t) \to_{\beta\pi} (E'[\,], t')$. Par le lemme de standardisation faible (lemme 9.1.23), il existe u' tel que $t' \mapsto_{\beta\pi} u'$ et $E[u] \to^*_{\beta\pi} E'[u']$.

Cas $(\Rightarrow)$. Comme pour le lemme 9.1.10.

Cas $(\times_1)$ et $(\times_2)$. Si $t = \pi_i(t_1, t_2)$, alors $u = t_i$ et on a $t' = \pi_1(t'_1, t'_2)$ et $u' = t'_i$ avec $(E'[\,], t'_1, t'_2) \leftarrow_{\beta\pi} (E[\,], t_1, t_2)$. Il s'en suit que $t'_1, t'_2 \in \mathcal{SN}_{\beta\pi}$ car $t_1, t_2 \in \mathcal{SN}_{\beta\pi}$ et que $E'[t'_i] \in \mathcal{SN}_{\beta\pi}$ car $E[t_i] \in \mathcal{SN}_{\beta\pi}$. On peut donc appliquer l'hypothèse d'induction à $v = E'[\pi_i(t'_1, t'_2)]$, d'où $v \in \mathcal{SN}_{\beta\pi}$. □

On en déduit que $\mathcal{SN}_{\beta\pi} \in \mathbb{SAT}_{\beta\pi}$. Montrons maintenant que $\_ \times \_$ et $\_ \Rightarrow \_$ sont des fonctions de $\mathbb{SAT}^2_{\beta\pi}$ dans $\mathbb{SAT}_{\beta\pi}$.

Proposition 9.1.25 *Si* $A_1, A_2 \in \mathbb{SAT}_{\beta\pi}$ *alors,*

$$A_2 \Rightarrow A_1 \in \mathbb{SAT}_{\beta\pi}\ , \qquad (\Rightarrow)$$
$$A_1 \times A_2 \in \mathbb{SAT}_{\beta\pi}\ . \qquad (\times)$$

Preuve. Le fait que $A_2 \Rightarrow A_1 \in \mathcal{SN}_{\beta\pi}$ se montre exactement comme à la proposition 9.1.13. De plus, si $t \in A_1 \times A_2$, alors $\pi_1 t \in A_1$, donc $t \in \mathcal{SN}_{\beta\pi}$ car $A_1 \in \mathbb{SAT}_{\beta\pi}$. Pour $(\Rightarrow)$, la clause $(\mathbb{SAT}1)$ est traitée comme à la proposition 9.1.13. Le cas $(\times)$ est similaire.

($\mathbb{SAT}1$) Si $E[\ ] \in \mathcal{SN}$ et $x \in \mathcal{X}$, alors $\pi_i E[x] \in A_i$ car $A_i \in \mathbb{SAT}_{\beta\pi}$ et $\pi_i E[\ ] \in \mathcal{E}_{\Rightarrow\times}$. Il s'en suit que $E[x] \in A_1 \times A_2$ pour tout $x \in \mathcal{X}$.

Considérons maintenant les clauses ($\mathbb{SAT}2_\beta$), ($\mathbb{SAT}2_{\pi 1}$) et ($\mathbb{SAT}2_{\pi 2}$). La satisfaction de ($\mathbb{SAT}2_\beta$) pour ($\Rightarrow$) se passe comme à la proposition 9.1.13. Cependant, il est intéressant de voir que ($\Rightarrow$) et ($\times$) peuvent se traiter de manière uniforme. Soient $i \in \{1,2\}$ et $t \mapsto_{\beta\pi} u$ avec

$$t = (\lambda x.t_1)t_2 \quad \text{et} \quad u = t_1[t_2/x]\ , \tag{$\mathbb{SAT}2_\beta$}$$
$$t = \pi_i(t_1, t_2) \quad \text{et} \quad u = t_i\ . \tag{$\mathbb{SAT}2_{\pi i}$}$$

Supposons que $t_2 \in \mathcal{SN}_{\beta\pi}$ dans le cas ($\mathbb{SAT}2_\beta$) et que $t_{3-i} \in \mathcal{SN}_{\beta\pi}$ dans le cas ($\mathbb{SAT}2_{\pi i}$).

($\Rightarrow$) On a $E[u] \in A_2 \Rightarrow A_1$ et on doit montrer que $E[t] \in A_2 \Rightarrow A_1$. Pour tout $v \in A_2$, comme $E[\ ]v \in \mathcal{E}_{\Rightarrow\times}$ et $E[u]v \in A_1$, on a $E[t]v \in A_1$ car $A_1 \in \mathbb{SAT}_{\beta\pi}$. D'où $E[t] \in A_2 \Rightarrow A_1$.

($\times$) On a $E[u] \in A_1 \times A_2$ et on doit montrer que $E[t] \in A_1 \times A_2$. Pour tout $j \in \{1,2\}$, comme $\pi_j E[\ ] \in \mathcal{E}_{\Rightarrow\times}$ et $\pi_j E[u] \in A_j$, on a $\pi_j E[t] \in A_j$ car $A_j \in \mathbb{SAT}_{\beta\pi}$. D'où $E[t] \in A_1 \times A_2$. □

On en déduit que $[\![\_]\!]$ est une interprétation valide. Montrons qu'elle est adéquate.

Lemme 9.1.26 (Adéquation) *Si* $\Gamma \vdash_{\Rightarrow\times} t : T$ *et* $\sigma \models \Gamma$, *alors* $t\sigma \in [\![T]\!]$.

Preuve. On raisonne comme au lemme 9.1.15. Les seuls changements sont les cas des règles ($\times$I) et ($\times$E).

($\times$I)
$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash (t_1, t_2) : T_1 \times T_2}$$

Soit $\sigma \models \Gamma$. On doit montrer que $(t_1, t_2)\sigma = (t_1\sigma, t_2\sigma) \in [\![T_1]\!] \times [\![T_2]\!]$.

Par hypothèse d'induction, on a $t_1\sigma \in [\![T_1]\!]$ et $t_2\sigma \in [\![T_2]\!]$. Pour tout $i \in \{1,2\}$, on a donc $\pi_i(t_1\sigma, t_2\sigma) \in [\![T_i]\!]$ par le lemme 9.1.24, d'où $(t_1\sigma, t_2\sigma) \in [\![T_1]\!] \times [\![T_2]\!]$.

($\times$E)
$$\frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash \pi_i t : T_i}\ (i \in \{1,2\})$$

Par définition de $\_ \times \_$. □

Corollaire 9.1.27 *Si* $\Gamma \vdash_{\Rightarrow\times} t : T$ *alors* $t \in \mathcal{SN}_{\beta\pi}$.

Une autre preuve de ce résultat peut être trouvée dans [GLT89].

9.1.4 Lambda-calcul avec réécriture

Nous avons vu les ingrédients de base des preuves par réductibilité pour le λ-calcul pur à la section 9.1.2, et nous les avons étendus au λ-calcul avec produits à la section 9.1.3. Dans cette section, nous nous intéressons au cas de la réécriture typée, telle que présentée à la section 3.6.

On se place dans le système $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma, \tau)$. De ce fait, la règle de typage des symboles $f \in \Sigma$ est

$$(\textsc{Symb}\,\mathrm{I})\ \frac{\Gamma \vdash t_1 : T_1 \quad \dots \quad \Gamma \vdash t_n : T_n}{\Gamma \vdash f(t_1, \dots, t_n) : T}\ ((T_1, \dots, T_n, T) \in \tau(f))$$

De plus, nous considérons un système de réécriture $\mathcal{R}$ typé dans $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma, \tau)$, au sens de la définition 3.6.7.

L'objectif de cette section est de présenter quelques principes généraux sur la manière dont la relation $\to_{\mathcal{R}}$ s'insère dans les preuves de normalisation par réductibilité présentées aux sections 9.1.2 et 9.1.3. Ces principes ont été identifiés dans [BJO02, Bla07].

L'interprétation d'un type $T \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ est définie, comme en 9.1.18, par induction sur T, mais en posant cette fois $[\![B]\!] = \mathcal{SN}_{\beta\pi\mathcal{R}}$:

$$\begin{array}{lll} [\![B]\!] & =_{\text{def}} & \mathcal{SN}_{\beta\pi\mathcal{R}} \qquad \text{si } B \in \mathcal{B}_0\ , \\ [\![U \Rightarrow T]\!] & =_{\text{def}} & [\![U]\!] \Rightarrow [\![T]\!]\ , \\ [\![T \times U]\!] & =_{\text{def}} & [\![T]\!] \times [\![U]\!]\ . \end{array}$$

Les contextes d'élimination que nous utilisons sont les contextes $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$ d'élimination des flèches et des produits, définis en 9.1.19.

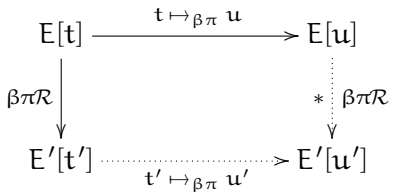
Nous raisonnons maintenant de la même manière qu'à la section 9.1.3 : nous montrons que les types sont interprétés par des ensembles saturés. Cela assure la validité de l'interprétation et son adéquation vis-à-vis des règles de typage ($\Rightarrow$I), ($\Rightarrow$E), ($\times$I) et ($\times$E). Ces ensembles saturés sont ceux de $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$, dans lesquels on utilise $\mathcal{SN}_{\beta\pi\mathcal{R}}$ au lieu de $\mathcal{SN}_{\beta\pi}$.

Comme les règles de réécriture sont algébriques à gauche, la propriété (9.10) est encore valable :

$$\forall v.\quad E[x] \to_{\beta\pi\mathcal{R}} v \quad\Longrightarrow\quad \big(v = E'[x] \ \text{ avec }\ E'[\ ] \in \mathcal{E}_{\Rightarrow\times} \ \text{ et }\ E[\ ] \to_{\beta\pi\mathcal{R}} E'[\ ]\big)\ . \tag{9.13}$$

On en déduit que si $E[\ ] \in \mathcal{E}_{\Rightarrow\times} \cap \mathcal{SN}_{\beta\pi\mathcal{R}}$ et $x \in \mathcal{X}$, alors $E[x] \in \mathcal{SN}_{\beta\pi\mathcal{R}}$; donc que $\mathcal{SN}_{\beta\pi\mathcal{R}}$ satisfait ($\mathcal{SAT}1$). Comme les règles de réécriture sont algébriques à gauche, les propriétés ($\mathcal{SAT}2_\beta$) et ($\mathcal{SAT}2_{\pi i}$) peuvent être montrées avec la même méthode qu'en 9.1.3.

Lemme 9.1.28 (Standardisation faible)



Il s'en suit que $\mathcal{SN}_{\beta\mathcal{R}}$ satisfait $(\mathbb{SAT}2_\beta)$ et $(\mathbb{SAT}2_{\pi i})$. Comme en 9.1.3, la flèche préserve les ensembles saturés.

Le point principal est le lemme d'adéquation. La réécriture y est présente à travers la règle (Symb I) qui impose que les symboles appartiennent à l'interprétation de leurs types [BJO02, Bla07].

Définition 9.1.29 *On dit que* $f \in \Sigma_n$ appartient à l'interprétation de son type*, notation* $f \in [\![\tau(f)]\!]$*, si pour tout* $t_1, \ldots, t_n$ *et tout* $(T_1, \ldots, T_n, T) \in \tau(f)$*, on a*

$$(\forall i \in \{1, \ldots, n\}. \quad t_i \in [\![T_i]\!]) \quad \Longrightarrow \quad f(t_1, \ldots, t_n) \in [\![T]\!] \ .$$

Lemme 9.1.30 (Adéquation) *Si* $f \in [\![\tau(f)]\!]$ *pour tout* $f \in \Sigma$*, alors*

$$(\Gamma \vdash_{\Rightarrow \times \tau} t : T \quad \wedge \quad \sigma \models \Gamma) \quad \Longrightarrow \quad t\sigma \in [\![T]\!] \ .$$

Preuve. On raisonne par induction sur $\Gamma \vdash_{\Rightarrow \times \tau} t : T$ comme pour 9.1.26. La seule différence est le cas de la règle (Symb I) :

$$\frac{\Gamma \vdash t_1 : T_1 \quad \ldots \quad \Gamma \vdash t_n : T_n}{\Gamma \vdash f : T} \ ((T_1, \ldots, T_n, T) \in \tau(f))$$

Si $\sigma \models \Gamma$, alors par hypothèse d'induction on a $t_i\sigma[\![T_i]\!]$ pour tout $i \in \{1, \ldots, n\}$, d'où $f(t_1, \ldots, t_n)\sigma \in [\![T]\!]$ car $f \in [\![\tau(f)]\!]$. □

La question principale est donc d'assurer $f \in [\![\tau(f)]\!]$. Pour cela, on va se concentrer sur des systèmes de réécriture typés dans $\boldsymbol{\lambda}_{\Rightarrow \times}(\mathcal{B}_0, \Sigma_0, \tau)$, où (Σ_0, τ) a une forme particulière.

Définition 9.1.31 (Signatures basées sur les produits) *On désigne par* $\mathcal{T}_\times(\mathcal{B}_0)$ *l'ensemble des types produits avec types de base dans* $\mathcal{B}_0$ *:*

$$T, U \in \mathcal{T}_\times(\mathcal{B}_0) \quad ::= \quad B \quad | \quad T \times U \ ,$$

où $B \in \mathcal{B}_0$.

Une signature (Σ_0, τ) *typée dans* $\mathcal{T}_{\Rightarrow \times}(\mathcal{B}_0)$ *est* basée sur les produits *si pour tout* $f \in \Sigma_0$*,* $\tau(f) = T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H$ *où* $H \in \mathcal{T}_\times(\mathcal{B}_0)$.

Nous utilisons des signatures basées sur les produits au chapitre 11 (voir aussi [BR06]).

Exemple 9.1.32 *Considérons la signature*

$$\Sigma \quad =_{def} \quad \Sigma_{\mathsf{Nat}\mathcal{C}} \uplus \Sigma_{\mathsf{List}\mathcal{C}} \uplus \Sigma_{\mathsf{Bool}} \uplus \{>, \mathsf{pivot}, \mathsf{ite}\}$$

où $\Sigma_{\mathsf{Nat}\mathcal{C}}$*,* $\Sigma_{\mathsf{List}\mathcal{C}}$*,* Σ_{Bool} *sont définies à la section 3.1 et où* ite *est la version curryfiée de l'itérateur sur les booléens présentée en 3.1.14. On la type de la manière suivante :*

$$\begin{array}{rcl@{\qquad}rcl}
\tau(0) & =_{def} & \mathsf{Nat} & \tau(\mathsf{nil}) & =_{def} & \mathsf{List} \\
\tau(S) & =_{def} & \mathsf{Nat} \Rightarrow \mathsf{Nat} & \tau(\mathsf{cons}) & =_{def} & \mathsf{Nat} \Rightarrow \mathsf{List} \Rightarrow \mathsf{List} \\
\tau(\mathsf{true}) & =_{def} & \tau(\mathsf{false}) \ =_{def} \ \mathsf{Bool} & \tau(>) & =_{def} & \mathsf{Nat} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Bool}
\end{array}$$

$$\begin{array}{rcl}
\tau(\mathsf{ite}) & =_{def} & \mathsf{Bool} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Nat} \\
\tau(\mathsf{pivot}) & =_{def} & \mathsf{Nat} \Rightarrow \mathsf{List} \Rightarrow \mathsf{List} \times \mathsf{List}
\end{array}$$

La signature typée (Σ, τ) *est basée sur les produits. Le symbole* pivot*, dont le type fait intervenir un produit, définit une fonction utilisée dans certaines implantations du tri* QuickSort *:*

$$\begin{array}{lll} \mathsf{pivot}\ x\ \mathsf{nil} & \mapsto_{\mathsf{pivot}} & (\mathsf{nil}, \mathsf{nil}) \\ \mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) & \mapsto_{\mathsf{pivot}} & \mathsf{ite}\ (>\ y\ x) \\ & & (\pi_1(\mathsf{pivot}\ x\ l), \mathsf{cons}\ y\ (\pi_2(\mathsf{pivot}\ x\ l))) \\ & & (\mathsf{cons}\ y\ (\pi_1(\mathsf{pivot}\ x\ l)), \pi_2(\mathsf{pivot}\ x\ l)) \end{array}$$

L'interprétation des types est stable par $\beta\pi\mathcal{R}$-réduction.

Proposition 9.1.33 (Stabilité par réduction) *Pour tout* $T \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$*, si* $t \in [\![T]\!]$ *et* $t \to_{\beta\pi\mathcal{R}} u$ *alors* $u \in [\![T]\!]$.

Preuve. Par induction sur T.

- $T = B \in \mathcal{B}$. Dans ce cas $[\![T]\!] = \mathcal{SN}_{\beta\pi\mathcal{R}}$ et $u \in \mathcal{SN}_{\beta\pi\mathcal{R}}$ par la définition 2.1.6.(i).
- $T = T_1 \times T_2$. Soit $i \in \{1, 2\}$. Comme $t \in [\![T_1]\!] \times [\![T_2]\!]$, on a $\pi_i t \in [\![T_i]\!]$, donc $\pi_i u \in [\![T_i]\!]$ hypothèse d'induction. Il s'en suit que $u \in [\![T_1]\!] \times [\![T_2]\!]$.
- $T = T_2 \Rightarrow T_1$. Soit $v \in [\![T_2]\!]$. Comme $t \in [\![T_2]\!] \Rightarrow [\![T_1]\!]$, on a $tv \in [\![T_1]\!]$, donc $uv \in [\![T_1]\!]$ hypothèse d'induction. Il s'en suit que $u \in [\![T_2]\!] \times [\![T_1]\!]$. □

De plus, comme les termes de la forme $f t_1 \dots t_n$ n'interagissent pas avec les contextes d'élimination des produits, pour tout $T \in \mathcal{T}_\times(\mathcal{B}_0)$, on a $f t_1 \dots t_n \in [\![T]\!]$ dès lors que tout les réduits en un pas de $f t_1 \dots t_n$ sont dans $[\![T]\!]$.

Proposition 9.1.34 *Soit* $T \in \mathcal{T}_\times(\mathcal{B}_0)$*. Pour tout* $f \in \Sigma_0$ *et tout* $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$ *on a*

$$(\forall v.\quad E[f] \to_{\beta\pi\mathcal{R}} v \quad\Longrightarrow\quad v \in [\![T]\!]) \quad\Longrightarrow\quad E[f] \in [\![T]\!]\ .$$

Preuve. Par induction sur T.

- $T = B \in \mathcal{B}_0$. Dans ce cas $[\![T]\!] = \mathcal{SN}_{\beta\pi\mathcal{R}}$ et par la définition 2.1.6.(i), $E[f] \in \mathcal{SN}_{\beta\pi\mathcal{R}}$ si $(E[f])_{\beta\pi\mathcal{R}} \in \mathcal{SN}_{\beta\pi\mathcal{R}}$.
- $T = T_1 \times T_2$. Soit $i \in \{1, 2\}$ et v tel que $\pi_i E[f] \to_{\beta\pi\mathcal{R}} v$. Alors $v = \pi_i v'$ avec $E[f] \to_{\beta\pi\mathcal{R}} v'$ et $\pi_i v' \in [\![T_i]\!]$ car $v' \in [\![T_1]\!] \times [\![T_2]\!]$. On a donc $\pi_i E[f] \in [\![T_i]\!]$ par hypothèse d'induction et il s'en suit que $E[f] \in [\![T_1]\!] \times [\![T_2]\!]$. □

Les propriétés 9.1.33 et 9.1.34 permettent de raffiner la propriété $f \in [\![\tau(f)]\!]$ de la définition 9.1.29. On obtient une condition classique, que l'on a utilisé dans [BR06], et qui correspond aux conditions formulées dans [BJO02, Bla07].

Lemme 9.1.35 (Computabilité des règles de réécriture) *Soit* (Σ_0, τ) *une signature typée dans* $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ *et basée sur les produits, et soit* $\mathcal{R}$ *un système de réécriture typé dans* $\lambda_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$.

Si pour toute règle $f l_1 \dots l_k \mapsto_{\mathcal{R}} r$ *avec* $\tau(f) = T_1 \Rightarrow \dots \Rightarrow T_n \Rightarrow H$ *et pour toute substitution* σ *on a*

$$(\forall i \in \{1, \dots, k\}.\quad l_i\sigma \in [\![T_i]\!]) \implies r\sigma \in [\![T_{k+1} \Rightarrow \dots \Rightarrow T_n \Rightarrow H]\!]\ ,$$

alors $\to_{\beta\pi\mathcal{R}}$ *est fortement normalisant sur les termes typables de* $\lambda_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$.

PREUVE. Par le lemme 9.1.30, il faut montrer que $f \in [\![\tau(f)]\!]$ pour tout $f \in \Sigma_0$. Soit $f \in \Sigma_0$ avec $\tau(f) = T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H$. Supposons que $t_i \in [\![T_i]\!]$ pour tout $i \in \{1, \ldots, n\}$. On doit montrer que $t =_{\text{def}} f t_1 \ldots t_n \in [\![H]\!]$. Par la proposition 9.1.34, il suffit de montrer que $v \in [\![H]\!]$ dès lors que $t \to_{\beta\pi\mathcal{R}} v$. On raisonne par induction sur les tuples $(t_1, \ldots, t_n)$ ordonnés par l'extension produit de $\to_{\beta\pi\mathcal{R}}$. Soit v tel que $t \to_{\beta\pi\mathcal{R}} v$.

— Si $v = f t'_1 \ldots t'_n$ avec $(t_1, \ldots, t_n) \to_{\beta\pi\mathcal{R}} (t'_1, \ldots, t'_n)$, alors, par la proposition 9.1.33, on a $t'_i \in [\![T_i]\!]$ pour tout $i \in \{1, \ldots, n\}$. On peut donc appliquer l'hypothèse d'induction à v, ce qui donne $v \in [\![H]\!]$.

— Sinon, il y a une règle $l \mapsto_{\mathcal{R}} r$ et une substitution σ telles que $t = l\sigma t_{k+1} \ldots t_n$ et $v = r\sigma t_{k+1} \ldots t_n$. Par hypothèse, on a $r\sigma \in [\![T_{k+1} \Rightarrow \cdots \Rightarrow T_n \Rightarrow H]\!]$, donc $v \in [\![H]\!]$ car $t_i \in [\![T_i]\!]$ pour tout $i \in \{k+1, \ldots, n\}$. □

9.2 Système F

Nous venons de voir les mécanismes combinatoires de base des preuves par réductibilité pour des systèmes de types simples. Dans cette section, nous nous intéressons aux preuves de normalisation forte par réductibilité pour le système F, présenté à la section 3.3.2.

Le système F comporte une difficulté particulière car sa normalisation implique la cohérence de l'arithmétique du second ordre. Sa preuve de normalisation, découverte par Girard [Gir72], doit donc utiliser des moyens logiques très puissants.

Nous considérons des types du second ordre construits avec des types de base pris dans une signature $\mathcal{B}$:

$$T, U \in \mathcal{T}_2(\mathcal{B}) \quad ::= \quad X \;\mid\; B(T_1, \ldots, T_n) \;\mid\; U \Rightarrow T \;\mid\; \forall X.T$$

où $X \in \mathcal{V}$ et $B \in \mathcal{B}_n$.

Rappelons qu'à cause du lieur $\forall X.\_$, les types du systèmes F sont des quotients modulo α-conversion (voir section 3.3.2.(a)).

9.2.1 Quantification du second ordre et produits infinis

Nous essayons dans cette section de mettre en évidence la difficulté logique intrinsèque aux preuves de normalisation du système F. En utilisant une variante du combinateur J de Girard [Gir72] proposée par [HM99], nous tentons d'expliquer pourquoi, dans les modèles de réductibilité, la quantification universelle du second ordre ne peut être naïvement interprétée par un produit infini.

On considère le système F *à la Church* $\boldsymbol{\lambda}_2^{\mathcal{C}h}(\mathcal{B}_0, \Sigma_0, \tau)$ avec des symboles d'arité nulle. On suppose que τ associe à chaque $f \in \Sigma_0$ un unique type $\tau(f) \in \mathcal{T}_2(\mathcal{B}_0)$. Ces termes sont donc décrits par la grammaire suivante :

$$\begin{array}{rcllll} t, u \in \Lambda_{2\Lambda}(\Sigma_0, \mathcal{B}_0) & ::= & x & \mid\; t\,u & \mid\; \lambda x.t \\ & \mid & t \cdot_\Lambda T & \mid\; \Lambda X.t & \mid\; f \end{array}$$

où $x \in \mathcal{X}$, $X \in \mathcal{V}$, $T \in \mathcal{T}_2(\mathcal{B}_0)$ et $f \in \Sigma_0$. Les règles de typage de ce système sont celles de $\boldsymbol{\lambda}_{\Rightarrow}^{\mathcal{C}h}(\mathcal{B}_0, \Sigma_0, \tau)$ plus les règles d'introduction et d'élimination de la quantification

universelle du second ordre :

$$(\forall_\wedge \mathrm{I})\ \frac{\Gamma \vdash t : T}{\Gamma \vdash \wedge X.t : \forall X.T}\ (X \notin \mathrm{FV}(\Gamma)) \qquad (\forall_\wedge \mathrm{E})\ \frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t \cdot U : T[U/X]}$$

Comme on ne considère que des symboles f d'arité nulle et ayant un unique type $\tau(f)$, leur règle de typage s'écrit de la manière suivante :

$$(\textsc{Symb}\,\mathrm{I})\ \frac{}{\Gamma \vdash f : \tau(f)}\ (f \in \Sigma_0)$$

Une façon naïve d'appréhender les règles $(\forall_\wedge \mathrm{I})$ et $(\forall_\wedge \mathrm{E})$ est de voir la quantification du second ordre comme une sorte de produit infini. Ainsi, la généralisation $\wedge X.t : \forall X.T$ serait une forme infinie de la paire $(t_1, t_2) : T_1 \times T_2$, et l'instantiation $t \cdot U : T[U/X]$ serait une forme infinie de projection $\pi_i t : T_i$.

L'interprétation de $\forall X.\_$ vu comme produit infini pourrait être définie de manière analogue à celle de $\_ \times \_$:

$$\begin{aligned} [\![T_1 \times T_2]\!] &= \{t \mid \forall i.\quad i \in \{1,2\} \implies \pi_i t \in [\![T_i]\!]\} \\ [\![\forall X.T]\!] &= \{t \mid \forall U.\quad U \in \mathcal{T}_2(\mathcal{B}_0) \implies t \cdot U \in [\![T[U/X]]\!]\}\,. \end{aligned} \tag{9.14}$$

Cependant, une telle interprétation de $\forall X.\_$ pose problème : le type $T[U/X]$ pouvant ne pas être plus petit que T, $[\![\_]\!]$ n'est pas définissable par induction sur T. Mais cela ne dit pas qu'il n'existe pas d'autre moyen de définir une interprétation de $\forall X.T$ vérifiant (9.14) et évitant ce problème d'induction sur T.

Nous allons voir, en utilisant une variante J' du combinateur J de Girard [Gir72] proposée dans [HM99], que toute interprétation qui vérifie (9.14) et qui est stable par expansion faible de tête fortement normalisante pour les règles de ce système contient des termes non fortement normalisants.

Les combinateurs J et J' sont des symboles définies par des règles de réécriture qui dépendent du type de leurs arguments, ils sont dits *non paramétriques*. Les conditions de saturation que nous utilisons étant très naturelles, il nous semble que le problème vient précisément de (9.14). Il se peut donc qu'il n'existe pas d'interprétation valide vérifiant (9.14) et qui soit adéquate vis-à-vis des règles de typage de $\boldsymbol{\lambda}^{\mathcal{C}h}_{2\tau}(\mathcal{B}_0, \Sigma_0, \tau)$.

Le combinateur J de Girard est un symbole d'arité nulle et de type $\forall X.\forall Y.X \Rightarrow Y$, défini par les règles suivantes :

$$J\ T\ U\ t \quad \mapsto_J \quad \begin{cases} t & \text{si } T = U \\ \mho\ U & \text{sinon} \end{cases}$$

où $\mho$ est un symbole de type $\forall X.X$. Les règles de J dépendent donc du type de ses arguments, c'est une fonction non paramétrique.

Sa variante J' de [HM99] évite l'utilisation du terme $\mho : \forall X.X$. Le symbole J' a pour type $\forall X.\forall Y.(X \Rightarrow X) \Rightarrow (Y \Rightarrow Y)$, et est défini par les règles

$$J'\ T\ U\ t \quad \mapsto_{J'} \quad \begin{cases} t & \text{si } T = U \\ \mathrm{Id}_{\mathcal{C}h}\ U & \text{sinon} \end{cases}$$

où $\mathrm{Id}_{\mathcal{C}h} = (\Lambda X.\lambda x : X.x)$ est l'identité dans le système F *à la Church* (voir exemple 3.3.19). Le combinateur J' peut être vu comme une restriction de J aux types flèches.

En posant $T_{Id} =_{def} \forall X.X \Rightarrow X$ et

$$f \quad =_{def} \quad \Lambda X.J'\ T_{Id}\ X\ (\lambda x : T_{Id}.x\ T_{Id}\ x)\ :\ T_{Id}\ ,$$

on a

$$f\ T_{Id}\ f\ \to_{\beta J'}\ J'\ T_{Id}\ T_{Id}\ (\lambda x : T_{Id}.x\ T_{Id}\ x)\ f\ \to_{\beta J'}\ (\lambda x : T_{Id}.x\ T_{Id}\ x)\ f\ \to_{\beta J'}\ f\ T_{Id}\ f\ .$$

On note $(\Sigma_{J'})$ la signature typée $\{J' : \forall X.\forall Y.(X \Rightarrow X) \Rightarrow (Y \Rightarrow Y)\}$.

Nous allons maintenant montrer que toute interprétation $[\![\_]\!]$ vérifiant (9.14) et certaines conditions de saturation simples est adéquate pour le système $\boldsymbol{\lambda}_2^{\mathcal{C}h}(\mathcal{B}_0, \Sigma_{J'})$, c'est-à-dire

$$\Gamma \vdash^{\mathcal{C}h}_{2\Sigma_{J'}} t : T \qquad \text{implique} \qquad t \in [\![T]\!]\ .$$

Il s'en suit que $f\ T_{Id}\ f \in [\![T_{Id}]\!]$, donc que $[\![T_{Id}]\!]$ contient des termes non normalisants. Commençons par la proposition suivante, prouvée par induction sur la hauteur des arbres de typage.

Proposition 9.2.1 *On note $\Gamma \vdash_n t : T$ si $\Gamma \vdash t : T$ est dérivable par un arbre de hauteur inférieure ou égale à n. Supposons que $\tau(f)$ est clos pour tout $f \in \Sigma_0$. Si $\Gamma \vdash_n t : T$ alors $\Gamma[U/X] \vdash_n t[U/X] : T[U/X]$ pour tout $U \in \mathcal{T}_2(\mathcal{B}_0)$.*

Preuve. Par induction sur $\Gamma \vdash t : T$.

(Ax)
$$\frac{}{\Gamma, x : T \vdash x : T}$$
Si $\Gamma, x : T \vdash_n x : T$ alors $\Gamma[U/X], x : T[U/X] \vdash_n x : T[U/X]$.

($\Rightarrow$ I)
$$\frac{\Gamma, x : T_2, \vdash_{n-1} t_1 : T_1}{\Gamma \vdash_n \lambda x : T_2.t_1 : T_2 \Rightarrow T_1}$$
Par hypothèse d'induction, car $(\lambda x : T_2.t_1)[U/X] = \lambda x : T_2[U/X].t_1[U/X]$.

($\forall_\Lambda$ I)
$$\frac{\Gamma \vdash_{n-1} t : T}{\Gamma \vdash_n \Lambda Y.t : \forall Y.T}\ (Y \notin FV(\Gamma))$$
Par hypothèse d'induction, en utilisant le fait qu'on peut supposer $X \notin FV(U) \cup \{Y\}$.

($\Rightarrow$ E) et ($\forall_\Lambda$ E) Par hypothèse d'induction.

(Symb I) Car par hypothèse $\tau(f)$ est clos pour tout $f \in \Sigma_0$. □

On considère des contextes d'élimination de la forme

$$E[\ ] \in \mathcal{E}_{2\mathcal{C}h} \quad ::= \quad [\ ] \quad | \quad E[\ ]\ t \quad | \quad E[\ ]\ T\ .$$

Les conditions de saturation que nous utilisons sont les suivantes : $S \subseteq \mathcal{SN}_{\beta J'}$ est $\beta J'$-*saturé* si

($\mathbb{SAT}1^{\mathcal{C}h}$) si $E[\ ] \in \mathcal{SN}_{\beta J'}$ alors $E[x] \in S$,

($\mathbb{SAT}2^{\mathcal{C}h}_{\beta}$) si $E[t[u/x]] \in S$ et $u \in \mathcal{SN}_{\beta J'}$ alors $E[(\lambda x : U)t] \in S$; si $E[t[U/X] \in S]$ alors $E[(\Lambda X.t)U] \in S$,

($\mathbb{SAT}2^{\mathcal{C}h}_{J'}$) si $E[t] \in S$ alors $E[J'\ U\ U\ t] \in S$; si $E[Id_{\mathcal{C}h}U] \in S$ et $t \in \mathcal{SN}_{\beta J'}$ avec $U \neq V$ alors $E[J'\ V\ U\ t] \in S$.

Le fait qu'il existe une partie $\beta J'$-saturée de $\mathcal{SN}_{\beta J'}$ provient de la standardisation faible (voir lemme 9.1.9) : si $t \mapsto_{\beta J'} u$ et $E[t] \to_{\beta J'} v$ avec $v \neq E[u]$, alors $v = E'[t']$ avec $(E[\ ], t) \to_{\beta J'} (E'[\ ], t')$ et il existe u' tel que $t' \mapsto_{\beta J'} u'$ et $E[u] \to^*_{\beta J'} E'[u']$.

Une interprétation $[\![\_]\!] : \mathcal{T}_2(\mathcal{B}) \to \mathcal{P}(\mathcal{SN}_{\beta J'})$ est $\beta J'$-*saturée* si $[\![T]\!]$ est $\beta J'$-saturé. Notons que si $[\![\_]\!]$ est $\beta J'$-saturé, alors on a $Id_{\mathcal{C}h}\ T \in [\![T]\!] \Rightarrow [\![T]\!]$.

Si σ est une valuation sur $\mathcal{X} \cup \mathcal{V}$ telle que $\sigma(x) \in [\![T]\!]$ si $x \in \mathcal{V}_T$ et $\sigma(X) \in \mathcal{T}_2(\mathcal{B}_0)$ si $X \in \mathcal{X}$, alors on note $\sigma \models_{2\mathcal{C}h} \Gamma$ si $\sigma(x) \in [\![\Gamma(x)]\!]$ pour tout $x \in \mathcal{D}om(\Gamma)$.

Lemme 9.2.2 (Adéquation) *Supposons qu'il existe une interprétation $[\![\_]\!]$ $\beta J'$-saturée et satisfaisant (9.14). Si $\Gamma \vdash^{\mathcal{C}h}_{2\Sigma_{J'}} t : T$ et $\sigma \models_{2\mathcal{C}h} \Gamma$ alors $t\sigma \in [\![T]\!]$.*

Preuve. Par induction sur la hauteur de $\Gamma \vdash t : T$. Les cas (Ax), ($\Rightarrow$ I) et ($\Rightarrow$ E) correspondent à ceux du lemme 9.1.15.

($\forall_\Lambda$ E) Par (9.14).

($\forall_\Lambda$ I) Pour tout $U \in \mathcal{T}_2(\mathcal{B}_0)$, par la proposition 9.2.1, comme $X \notin FV(\Gamma, t)$ on a $\Gamma \vdash t[U/X] : T[U/X]$ avec un arbre de même hauteur, donc par hypothèse d'induction $t\sigma[U/X] \in [\![T[U/X]]\!]$. Il s'en suit que $(\Lambda X.t)\sigma U \in [\![T[U/X]]\!]$ par ($\mathbb{SAT}2^{\mathcal{C}h}_{\beta}$). Donc $(\Lambda X.t)\sigma \in [\![\forall X.T]\!]$.

(Symb I) On doit montrer que pour tout $T, U \in \mathcal{T}_2(\mathcal{B}_0)$, $J'\ T\ U \in [\![(T \Rightarrow T) \Rightarrow (U \Rightarrow U)]\!]$. Soit $t \in [\![T \Rightarrow T]\!]$. Par ($\mathbb{SAT}2^{\mathcal{C}h}_{J'}$), si $T = U$, on a $J'\ T\ T\ t \in [\![T \Rightarrow T]\!]$ et sinon, comme $(\Lambda X.\lambda x : X.x)U \in [\![U \Rightarrow U]\!]$, on a $J'\ T\ U\ t \in [\![U \Rightarrow U]\!]$. □

Donc s'il existe une interprétation $\beta J'$-saturée satisfaisant (9.14), alors tout terme typable dans $\lambda^{\mathcal{C}h}_2(\mathcal{B}_0, \Sigma_{J'})$ est fortement $\beta J'$-normalisant. Comme le terme $f\ T_{Id}\ f$ évoqué plus haut est typable mais pas fortement normalisant, il s'en suit qu'il n'existe pas d'interprétation satisfaisant ces conditions.

Un système fortement non paramétrique

Les règles ($\forall_\Lambda$ I) et ($\forall_\Lambda$ E) disent, avec la proposition 9.2.1, que si le terme t a un type dépendant d'une variable X n'apparaissant dans le contexte, alors $t[U/X]$ a pour type $T[U/X]$ pour tout U. Cependant, en l'absence de réécriture, le calcul effectué par le terme $t[U/X]$ est essentiellement le même que celui effectué par t : le polymorphisme du système F pur est paramétrique.

En s'inspirant de (9.14), on peut imaginer un système fortement non paramétrique dans lequel une famille de termes $(t_U)_{U \in \mathcal{T}}$, chacun de type T_U, puisse être de type $\Pi_{U \in \mathcal{T}} T_U$. Une telle définition n'est pas bien fondée, et donc ce système n'existe probablement pas. Cependant, il semble représenter l'idéal de paramétricité, et il est intéressant de voir que l'on pourrait y coder les règles du combinateur J'.

On aurait des règles comme celles-ci :

$$(\Pi_2\,\mathrm{I})\ \frac{\forall U \in \mathcal{T}.\quad \Gamma \vdash t_U : T_U}{\Gamma \vdash (t_U)_{U\in\mathcal{T}} : \Pi_{U\in\mathcal{T}} T_U} \qquad (\Pi_2\,\mathrm{E})\ \frac{\Gamma \vdash t : \Pi_{U\in\mathcal{T}} T_U}{\Gamma \vdash t \cdot U : T_U}$$

Ainsi, $(t_U)_{U\in\mathcal{T}}$ serait une généralisation de $\Lambda X.t_X$ (avec $X \notin FV(\Gamma)$), et l'application $\_\cdot U$ serait vue comme une version infinie de la projection $\pi_i\_$.

Une règle de réduction $(t_U)_{U\in\mathcal{T}} \cdot U \mapsto_\Pi t_U$ pourrait alors généraliser la règle de β-réduction sur les types $(\Lambda X.t)U \mapsto t[U/X]$.

Une interprétation similaire à (9.14) conviendrait bien :

$$[\![\Pi_{U\in\mathcal{T}} T_U]\!] \quad = \quad \{t \mid \forall U \in \mathcal{T}.\quad t \cdot U \in [\![T_U]\!]\}\,. \tag{9.15}$$

Bien entendu, comme pour (9.14) une telle interprétation ne pourrait être définie par induction sur T. Les ensembles βΠ-saturés seraient les ensembles S de termes fortement βΠ-normalisants tels que

($\mathcal{SAT}1^{Ch}$) si $E[\ ] \in \mathcal{SN}_{\beta\Pi}$ alors $E[x] \in S$,

($\mathcal{SAT}2^{Ch}_\beta$) si $E[t[u/x]] \in S$ et $u \in \mathcal{SN}_{\beta\Pi}$ alors $E[(\lambda x : U)t] \in S$,

($\mathcal{SAT}2^{Ch}_\Pi$) si $E[t_U] \in S$ alors $E[(t_U)_{U\in\mathcal{T}} \cdot U] \in S$.

Comme au lemme 9.2.2, une interprétation βΠ-saturée vérifiant (9.15) serait correcte par rapport aux règles de typage, et on en déduirait que si $\Gamma \vdash t : T$ alors $t \in [\![T]\!] \subseteq \mathcal{SN}_{\beta\Pi}$.

Cependant, les règles de J' sont codables dans ce système, et donnent lieu à un terme typable non βΠ-normalisable. Avec $Id_\Pi =_{\text{def}} (\lambda x : U.x)_{U\in\mathcal{T}}$ et $U_{Id} =_{\text{def}} \Pi_{U\in\mathcal{T}} U \Rightarrow U$, on a $Id_\Pi : U_{Id}$. Posons, pour tout $U, V \in \mathcal{T}$,

$$t_{UV} \quad =_{\text{def}} \quad \begin{cases} \lambda x : U \Rightarrow U.x & \text{si } U = V \\ \lambda x : U \Rightarrow U.Id_\Pi \cdot V & \text{sinon} \end{cases}$$

Avec

$$f \quad =_{\text{def}} \quad (t_{U_{Id}W}\,(\lambda x : U_{Id}.x\ U_{Id}\ x))_{W\in\mathcal{T}} \quad : \quad \Pi_{W\in\mathcal{T}} W \Rightarrow W \quad = \quad U_{Id}\,,$$

on a

$$f\ U_{Id}\ f \ \to_{\beta\Pi}\ t_{U_{Id}U_{Id}}\ (\lambda x : U_{Id}.x\ U_{Id}\ x) f \ \to_{\beta\Pi}\ (\lambda x : U_{Id}.x\ U_{Id}\ x) f \ \to_{\beta\Pi}\ f\ U_{Id}\ f\,.$$

9.2.2 Interprétation du polymorphisme

Nous avons vu à la section précédente qu'il ne peut y avoir d'interprétation correcte du système F qui autorise les fonctions non paramétriques. Dans cette section, nous présentons l'interprétation de Girard [Gir72] du système F.

Commençons par rappeler l'idée fondamentale de cette interprétation. On se place dans le système F pur implicite $\lambda_2(\mathcal{B}_0)$ présenté à la section 3.3.2.(b). Rappelons que les termes de ce système sont :

$$t, u \in \Lambda \quad ::= \quad x \quad | \quad tu \quad | \quad \lambda x.t$$

où $x \in \mathcal{X}$, et que ses règles de typage sont celles de $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}_0)$ plus les règles suivantes :

$$(\forall\,\mathrm{I})\ \frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T}\ (X \notin \mathrm{FV}(\Gamma)) \qquad (\forall\,\mathrm{E})\ \frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : T[U/X]}$$

Essayons de voir ce que doit satisfaire une interprétation correcte de $\forall X.\_$. Considérons un terme $t : \forall X.T$. Par la règle $(\forall\,\mathrm{E})$, on doit avoir $t \in [\![T[U/X]]\!]$ pour tout $U \in \mathcal{T}_2(\mathcal{B}_0)$. Notons que ceci est en accord avec le fait que l'interprétation de $\forall X.\_$ ne doit pas autoriser les fonctions non paramétriques. Il serait tentant de poser $[\![\forall X.T]\!] = \bigcap_{U\in\mathcal{T}_2(\mathcal{B}_0)} [\![T[U/X]]\!]$. Mais de même que (9.14), cette interprétation n'est pas définissable par induction sur T.

C'est ici qu'est la clef de l'interprétation de Girard [Gir72] : on spécifie une famille d'ensembles de réductibilité $\mathcal{R}\mathrm{ed}$, et on interprète $\forall X.\_$ non plus en quantifiant sur les types, mais en quantifiant sur les éléments de $\mathcal{R}\mathrm{ed}$.

Pour cela, on définit l'interprétation des types à partir d'un assignement $\rho : \mathcal{V} \to \mathcal{R}\mathrm{ed}$ de ses variables de type, et on pose

$$[\![\forall X.T]\!]\rho \quad = \quad \bigcap_{R\in\mathcal{R}\mathrm{ed}} [\![T]\!]\rho[R/X]\ . \tag{9.16}$$

Remarque 9.2.3 *Notons que l'interprétation (9.16) doit être légèrement modifiée pour le système F* à la Church*, voir par exemple [GLT89, Gal89].*

En ce qui concerne la preuve que nous présentons ici pour le système F à la Curry, le lecteur peut consulter [Kri90, Bar92].

Il est utile, pour la suite, de définir les interprétations de type dans un cadre plus général que celui de $\boldsymbol{\lambda}_2(\mathcal{B}_0)$.

Définition 9.2.4 (Familles de réductibilité) *Soit $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma, \tau)$ un système de types avec constantes et $\to_R$ une relation de réécriture sur $\Lambda(\Sigma)$.*

(i) Une famille de réductibilité *est une partie $\mathcal{R}\mathrm{ed}$ de $\mathcal{P}(\Lambda(\Sigma))$ telle que*

$$\begin{aligned} A, B \in \mathcal{R}\mathrm{ed} \quad &\Longrightarrow \quad A \Rightarrow B \in \mathcal{R}\mathrm{ed} \\ \emptyset \neq \mathcal{R} \subseteq \mathcal{R}\mathrm{ed} \quad &\Longrightarrow \quad \bigcap \mathcal{R} \in \mathcal{R}\mathrm{ed}\ . \end{aligned}$$

(ii) Un assignement *dans une famille de réductibilité $\mathcal{R}\mathrm{ed}$ est une fonction totale ρ de $\mathcal{V}$ dans $\mathcal{R}\mathrm{ed}$.*

(iii) Une interprétation *est la donnée d'une famille de réductibilité $\mathcal{R}\mathrm{ed}$ et d'une famille de fonctions $[\![\_]\!] = ([\![\_]\!]_n)_{n\in\mathbb{N}}$ telle que $[\![\_]\!]_n : \mathcal{B}_n \to \mathcal{R}\mathrm{ed}^{n+1}$ pour tout $n \in \mathbb{N}$.*

(iv) L'interprétation des types $T \in \mathcal{T}_2(\mathcal{B})$ *dans une interprétation $(\mathcal{R}\mathrm{ed}, [\![\_]\!])$ est définie par induction sur T à partir d'un assignement $\rho : \mathcal{V} \to \mathcal{R}\mathrm{ed}$ comme suit :*

$$\begin{array}{lll} [\![B(T_1,\ldots,T_n)]\!]\rho & =_{def} & [\![B]\!]_n([\![T_1]\!]\rho,\ldots,[\![T_n]\!]\rho) \qquad \textit{si } B \in \mathcal{B}_n \\ [\![X]\!]\rho & =_{def} & \rho(X) \qquad\qquad\qquad\qquad\quad \textit{si } X \in \mathcal{V} \\ [\![U \Rightarrow T]\!]\rho & =_{def} & [\![U]\!]\rho \Rightarrow [\![T]\!]\rho \\ [\![\forall X.T]\!]\rho & =_{def} & \bigcap_{R\in\mathcal{R}\mathrm{ed}} [\![T]\!]\rho[R/X]\ . \end{array}$$

(v) Une valuation *dans une famille de réductibilité* $\mathcal{R}\mathrm{ed}$ *est la donnée d'une substitution* $\sigma : \mathcal{X} \to \Lambda(\Sigma)$ *et d'un assignement* $\rho : \mathcal{V} \to \mathcal{R}\mathrm{ed}$.

(vi) *Une valuation* (σ, ρ) valide *un contexte* Γ *dans une interprétation* $(\mathcal{R}\mathrm{ed}, [\![\_]\!])$, *notation* $(\sigma, \rho) \models_{(\mathcal{R}\mathrm{ed}, [\![\_]\!])} \Gamma$, *lorsque* $\mathcal{D}\mathrm{om}(\Gamma) \subseteq \mathcal{D}\mathrm{om}(\sigma)$ *et* $\sigma(x) \in [\![\Gamma(x)]\!]\rho$ *pour tout* $x \in \mathcal{D}\mathrm{om}(\Gamma)$.

(vii) *Une interprétation* $(\mathcal{R}\mathrm{ed}, [\![\_]\!])$ *est* valide *si* $\mathcal{X} \subseteq [\![T]\!]\rho \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ *pour tout* $T \in \mathcal{T}_2(\mathcal{B})$ *et tout* $\rho : \mathcal{V} \to \mathcal{R}\mathrm{ed}$.

(viii) *Une interprétation* $(\mathcal{R}\mathrm{ed}, [\![\_]\!])$ *est* adéquate *si*

$$\big(\Gamma \vdash_{2\tau} t : T \quad \textit{et} \quad (\sigma, \rho) \models_{(\mathcal{R}\mathrm{ed}, [\![\_]\!])} \Gamma\big) \quad \Longrightarrow \quad t\sigma \in [\![T]\!]\rho \ .$$

Lorsque le contexte le permet, nous désignons $(\sigma, \rho) \models_{(\mathcal{R}\mathrm{ed}, [\![\_]\!])} \Gamma$ par $\sigma \models \Gamma$. Si ρ est un assignement dans $\mathcal{R}\mathrm{ed}$ et $R \in \mathcal{R}\mathrm{ed}$, alors on désigne par $\rho[R/X]$ l'assignement qui vaut R en X est qui est égal à ρ partout ailleurs.

Comme à la section 9.1.2, il est clair que si l'on dispose d'une interprétation valide et adéquate, alors tout terme typable dans $\lambda_2(\mathcal{B}, \Sigma, \tau)$ est fortement $\beta\mathcal{R}$-normalisable.

Théorème 9.2.5 (Normalisation forte) *Soit* $\lambda_2(\mathcal{B}, \Sigma, \tau)$ *un système de types avec constantes et* $\mathcal{R}$ *un TRS sur* $\Lambda(\Sigma)$. *S'il existe une interprétation valide et adéquate de* $(\lambda_2(\mathcal{B}, \Sigma, \tau), \mathcal{R})$, *alors*

$$\Gamma \vdash_{2\tau} t : T \quad \Longrightarrow \quad t \in \mathcal{SN}_{\beta\mathcal{R}} \ .$$

Preuve. Même chose qu'au théorème 9.1.2. □

Revenons maintenant sur l'interprétation de la quantification universelle du second ordre (9.16) dans une interprétation $(\mathcal{R}\mathrm{ed}, [\![\_]\!])$. Considérons la règle d'élimination $(\forall\,\mathrm{E})$:

$$\frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : T[U/X]}$$

Supposons que $(\sigma, \rho) \models \Gamma$ et que $t\sigma \in \bigcap_{R \in \mathcal{R}\mathrm{ed}} [\![T]\!]\rho[R/X]$. On doit pouvoir en déduire que $t\sigma \in [\![T[U/X]]\!]\rho$. Or on a bien $t\sigma \in [\![T]\!]\rho[[\![U]\!]\rho/X]$, mais pour pouvoir conclure, il faut que $[\![T]\!]\rho[[\![U]\!]\rho/X] \subseteq [\![T[U/X]]\!]\rho$.

En fait, on a $[\![T]\!]\rho[[\![U]\!]\rho/X] = [\![T[U/X]]\!]\rho$ pour toute famille de réductibilité $\mathcal{R}\mathrm{ed}$ et tout assignement ρ.

Proposition 9.2.6 *Soient* $T, U \in \mathcal{T}_2(\mathcal{B})$ *et* ρ *un assignement dans une famille de réductibilité* $\mathcal{R}\mathrm{ed}$.

(i) *Si* ρ' *est un assignement dans* $\mathcal{R}\mathrm{ed}$ *tel que* $\rho'(X) = \rho(X)$ *pour tout* $X \in \mathsf{FV}(T)$ *alors on a* $[\![T]\!]\rho = [\![T]\!]\rho'$.

(ii) *On a* $[\![T[U/X]]\!]\rho = [\![T]\!]\rho[[\![U]\!]\rho/X]$.

Preuve.

(i) Par induction sur T.

$T = B(T_1, \ldots, T_n)$ avec $B \in \mathcal{B} \cup \{\_ \Rightarrow \_\}$. Par hypothèse d'induction.

$T = X \in \mathcal{V}$. Alors $[\![T]\!]\rho = \rho(T) = \rho'(T) = [\![T]\!]\rho'$.

$T = \forall X.T_1$. Alors $[\![T]\!]\rho = \bigcap_{R\in\mathcal{R}ed}[\![T_1]\!]\rho[R/X]$ et $[\![T]\!]\rho' = \bigcap_{R\in\mathcal{R}ed}[\![T_1]\!]\rho'[R/X]$. Or, pour tout $R \in \mathcal{R}ed$, $\rho[R/X]$ et $\rho'[R/X]$ coïncident sur les variables libres de T_1 donc par hypothèse d'induction $[\![T_1]\!]\rho[R/X] = [\![T_1]\!]\rho'[R/X]$. Il s'en suit que $[\![T]\!]\rho = [\![T]\!]\rho'$.

(ii) Par induction sur T.

$T = B(T_1, \dots, T_n)$ avec $B \in \mathcal{B} \cup \{\_ \Rightarrow \_\}$. Par hypothèse d'induction.

$T = Y \in \mathcal{V}$. Si $Y = X$ alors $[\![T[U/X]]\!]\rho = [\![U]\!]\rho = [\![T]\!]\rho[[\![U]\!]\rho/X]$. Sinon, $[\![T[U/X]]\!]\rho = \rho(Y) = [\![T]\!]\rho[[\![U]\!]\rho/X]$.

$T = \forall Y.T_1$. Par le lemme 1.2.10, on peut supposer que $Y \notin FV(U) \cup \{X\}$. On a donc $T[U/X] = \forall Y.T_1[U/X]$. Il s'en suit que $[\![T[U/X]]\!]\rho = \bigcap_{R\in\mathcal{R}ed}[\![T_1[U/X]]\!]\rho[R/Y]$ et que $[\![T]\!]\rho[[\![U]\!]\rho/X] = \bigcap_{R\in\mathcal{R}ed}[\![T_1]\!]\rho[[\![U]\!]\rho/X][R/Y]$.

Par hypothèse d'induction, pour tout $R \in \mathcal{R}ed$ on a $[\![T_1[U/X]]\!]\rho[R/Y] = [\![T_1]\!]\rho[R/Y][[\![U]\!]\rho[R/Y]/X]$. Comme $Y \notin FV(U)$, les assignements $\rho[R/Y]$ et ρ coïncident sur $FV(U)$, donc $[\![U]\!]\rho[R/Y] = [\![U]\!]\rho$ par (i). De plus, comme $X \neq Y$, $\rho[[\![U]\!]\rho/X][R/Y]$ et $\rho[R/Y][[\![U]\!]\rho/X]$ coïncident sur $FV(T_1)$, donc par (i), on a $[\![T_1]\!]\rho[R/Y][[\![U]\!]\rho/X] = [\![T_1]\!]\rho[[\![U]\!]\rho/X][R/Y]$.

Il s'en suit que $[\![T[U/X]]\!]\rho = [\![T]\!]\rho[[\![U]\!]\rho/X]$. □

La proposition 9.2.6.(ii) donne tout ce qu'il faut pour interpréter les règles ($\forall$ I) et ($\forall$ E). De ce fait, une interprétation $(\mathcal{R}ed, [\![\_]\!])$ telle que $[\![\_]\!]\rho$ est adéquate vis-à-vis des règles de typage de $\lambda_{\Rightarrow}(\mathcal{B}, \Sigma, \tau)$ pour tout $\rho : \mathcal{V} \to \mathcal{R}ed$ est adéquate pour $\lambda_2(\mathcal{B}, \Sigma, \tau)$. La clause 9.2.7.(ii) prend à la fois en compte les produits et la condition présentée en 9.1.29.

Lemme 9.2.7 (Adéquation partielle) *Soit $\lambda_2(\mathcal{B}, \Sigma, \tau)$ un système de types et $\mathcal{R}$ un TRS sur $\Lambda(\Sigma)$. Soit $(\mathcal{R}ed, [\![\_]\!])$ une interprétation telle que pour toute valuation (ρ, σ),*

(i) si $t\sigma[u/x] \in [\![T]\!]\rho$ pour tout $u \in [\![U]\!]\rho$ alors $\lambda x.t\sigma \in [\![U \Rightarrow T]\!]\rho$,

(ii) pour tout $f \in \Sigma$ et tout $(T_1, \dots, T_n, T) \in \tau(f)$, si $t_1\sigma \in [\![T_1]\!]\rho, \dots, t_n\sigma \in [\![T_n]\!]\rho$ alors $f(t_1, \dots, t_n)\sigma \in [\![T]\!]\rho$.

Alors $(\mathcal{R}ed, [\![\_]\!])$ est adéquate pour $\lambda_2(\mathcal{B}, \Sigma, \tau)$.

Preuve. On raisonne par induction sur $\Gamma \vdash t : T$.

(Ax) Comme pour le lemme 9.1.15.

($\Rightarrow$ I) et (Symb I) Par hypothèse.

($\Rightarrow$ E) Par définition de $\_ \Rightarrow \_$.

($\forall$ I)

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t : \forall X.T} \ (X \notin FV(\Gamma))$$

Soient σ et ρ tels que $(\sigma, \rho) \models \Gamma$. Comme $X \notin FV(\Gamma)$, pour tout $R \in \mathcal{R}ed$ on a $(\sigma, \rho[R/X]) \models \Gamma$ par la proposition 9.2.6.(i). Il s'en suit par hypothèse d'induction que $t\sigma \in [\![T]\!]\rho[R/X]$. Donc $t\sigma \in \bigcap_{R\in\mathcal{R}ed}[\![T]\!]\rho[R/X]$.

($\forall$ E)

$$(\forall\,\mathrm{E})\ \frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : T[U/X]}$$

Soient σ et ρ tels que $(\sigma, \rho) \models \Gamma$. On a $t\sigma \in \bigcap_{R \in \mathcal{R}ed} [\![T]\!]\rho[R/X]$ par hypothèse d'induction, donc $t\sigma \in [\![T]\!]\rho[[\![U]\!]\rho/X]$. On en déduit $t\sigma \in [\![T[U/X]]\!]\rho$ par la proposition 9.2.6.(ii). □

9.3 Familles de réductibilité

Ainsi, étant donnés un système de type $\boldsymbol{\lambda}_2(\mathcal{B}, \Sigma, \tau)$ et un TRS $\mathcal{R}$ sur $\Lambda(\Sigma)$, nous avons vu que pour obtenir la $\beta\mathcal{R}$-normalisation forte des termes typables, nous devons définir une interprétation $(\mathcal{R}ed, [\![\_]\!])$ qui satisfasse les conditions (i) et (ii) du lemme 9.2.7, et de plus telle que

$$A, B \in \mathcal{R}ed \implies A \Rightarrow B \in \mathcal{R}ed \tag{9.17}$$

$$\emptyset \neq \mathcal{R} \subseteq \mathcal{R}ed \implies \bigcap \mathcal{R} \in \mathcal{R}ed\ . \tag{9.18}$$

Dans cette section, nous passons en revue des interprétations connues du système F qui satisfont ces propriétés. Toutes ces interprétations ont en commun de pouvoir être définies par des opérateurs de clôture, ce qui permet d'avoir automatiquement la stabilité par intersections non vides (9.18).

9.3.1 Opérateurs de clôture

Nous rappelons ici les notions sur les opérateurs de clôture qui nous seront utiles par la suite. Notre utilisation des opérateurs de clôture est inspirée de [VM04].

Définition 9.3.1 *Soit $\mathcal{D} = (D, \leq, \bigvee, \bigwedge, \bot, \top)$ un treillis complet.* Un opérateur de clôture *sur $\mathcal{D}$ est une fonction $\overline{(\_)} : D \to D$ qui est*

idempotente : $\overline{\overline{d}} = \overline{d}$ *pour tout* $d \in D$,

extensive : $d \leq \overline{d}$ *pour tout* $d \in D$,

monotone : $d \leq e \implies \overline{d} \leq \overline{e}$ *pour tout* $d, e \in D$.

Un élément $d \in D$ est clos pour $\overline{(\_)}$ *s'il existe $e \in D$ tel que $d = \overline{e}$. On dénote par $\overline{D}$ l'ensemble des éléments de D qui sont clos pour $\overline{(\_)}$.*

Proposition 9.3.2 *Un élément $d \in D$ est clos pour $\overline{(\_)}$ si et seulement si $\overline{d} = d$.*

Preuve. En effet, si $d = \overline{d}$, alors d est clos par définition, et si $d = \overline{e}$, alors par idempotence $\overline{d} = \overline{\overline{e}} = \overline{e} = d$. □

Les opérateurs de clôture préservent les plus grandes bornes inférieures des treillis sur lesquels ils opèrent.

Proposition 9.3.3 *Soit $\overline{(\_)}$ un opérateur de clôture sur $\mathcal{D}$. Pour tout $X \subseteq \overline{D}$ on a $\bigwedge X \in \overline{D}$.*

PREUVE. Soit $X \subseteq \overline{D}$ et montrons que $\overline{\bigwedge X} = \bigwedge X$. Comme $\overline{(\_)}$ est extensive, il suffit de vérifier que $\overline{\bigwedge X} \leq \bigwedge X$. Par définition de $\bigwedge$, c'est le cas si $\overline{\bigwedge X} \leq d$ pour tout $d \in X$. Or, si $d \in X$, comme $\bigwedge X \leq d$, par monotonicité de $\overline{(\_)}$ on a $\overline{\bigwedge X} \leq \overline{d}$, soit $\overline{\bigwedge X} \leq d$ car d est clos. □

L'intérêt des familles de réductibilité qui sont des opérateurs de clôture est qu'elles valident automatiquement (9.18). De plus, l'ensemble des éléments clos de $\mathcal{D}$ forme un treillis complet.

Lemme 9.3.4 *Soit $\overline{(\_)}$ un opérateur de clôture sur $\mathcal{D}$. Alors $\overline{\mathcal{D}} =_{def} (\overline{D}, \leq, \overline{\bigvee}, \bigwedge, \overline{\bot}, \top)$, où $\overline{\bigvee} X = \overline{\bigvee X}$ pour tout $X \subseteq \overline{D}$ est un treillis complet.*

PREUVE. Tout d'abord, comme $\overline{(\_)}$ est extensif, on a $\top \leq \overline{\top}$, donc $\overline{\top} = \top$. Par monotonicité de $\overline{(\_)}$, pour tout $d \in D$, on a $\overline{\bot} \leq \overline{d} \leq \top$. De plus, la proposition 9.3.3 dit que les plus grandes bornes inférieures de $\overline{\mathcal{D}}$ sont données par $\bigwedge$.

Vérifions maintenant que $\overline{\bigvee}$ donne bien les plus petites bornes supérieures de $\overline{\mathcal{D}}$. Il faut montrer que pour tout $X \subseteq \overline{D}$, et tout $e \in \overline{D}$, si $d \leq e$ pour tout $d \in X$ alors $\overline{\bigvee X} \leq e$. Or, si $d \leq e$ pour tout $d \in X$ alors $\bigvee X \leq e$ par définition de $\bigvee$. Il s'en suit que $\overline{\bigvee X} \leq \overline{e}$ par monotonicité de $\overline{(\_)}$, d'où $\overline{\bigvee X} \leq e$ car e est clos. □

9.3.2 Ensembles saturés

Dans le cadre du λ-calcul pur et du λ-calcul avec produits, nous avons vu, aux sections 9.1.2 et 9.1.3, que la famille $\mathbb{SAT}$ des ensembles saturés satisfait la propriété (9.17) ainsi que les conditions 9.2.7.(i) et 9.2.7.(ii).

Il reste donc à vérifier que $\mathbb{SAT}$ est stable par intersections arbitraires. Pour ce faire, dans le cadre du λ-calcul pur, nous montrons que les ensembles saturés sont les éléments clos pour un opérateur de clôture sur le treillis complet $\mathcal{P}(\mathcal{SN}_\beta)$.

Définition 9.3.5 *On définit les fonctions $\mathbb{SAT}_i : \mathcal{P}(\mathcal{SN}_\beta) \to \mathcal{P}(\mathcal{SN}_\beta)$ par induction sur $i \in \mathbb{N}$ comme suit*

$$\begin{array}{lcrcl} \mathbb{SAT}_0(X) & =_{def} & X & \cup & \{E[x] \mid E[\,] \in \mathcal{E}_\Rightarrow \cap \mathcal{SN}_\beta \ \wedge\ x \in \mathcal{X}\} \\ \mathbb{SAT}_{i+1}(X) & =_{def} & \mathbb{SAT}_i(X) & \cup & \{E[(\lambda x.t)u] \mid E[t[u/x]] \in \mathbb{SAT}_i(X) \ \wedge\ u \in \mathcal{SN}_\beta\}\ . \end{array}$$

On définit la fonction $\mathbb{SAT} : \mathcal{P}(\mathcal{SN}_\beta) \to \mathcal{P}(\mathcal{SN}_\beta)$ par $\mathbb{SAT}(X) =_{def} \bigcup_{i\in\mathbb{N}} \mathbb{SAT}_i(X)$ pour tout $X \subseteq \mathcal{SN}_\beta$.

Montrons que $\mathbb{SAT}$ est bien l'opérateur de clôture qui définit les ensembles saturés.

Lemme 9.3.6 *Si $X \subseteq \mathcal{SN}_\beta$, $\mathbb{SAT}(X)$ est le plus petit ensemble saturé contenant X.*

PREUVE. On montre tout d'abord que $\mathbb{SAT}(X) \in \mathbb{SAT}$ pour tout $X \subseteq \mathcal{SN}_\beta$. Commençons pas vérifier par induction sur $i \in \mathbb{N}$ que $\mathbb{SAT}_i(X) \subseteq \mathcal{SN}_\beta$ pour tout $i \in \mathbb{N}$.

Cas de base. Comme $X \subseteq \mathcal{SN}_\beta$, on a $\mathbb{SAT}_0(X) \subseteq \mathcal{SN}_\beta$ car $E[x] \in \mathcal{SN}_\beta$ si $E[\,] \in \mathcal{SN}_\beta$.

Cas d'induction. Si $t \in \mathbb{SAT}_{i+1}(X)$, alors ou bien $t \in \mathbb{SAT}_i(X)$ et $t \in \mathcal{SN}_\beta$ par hypothèse d'induction, ou bien $t = E[(\lambda x.u)v]$ avec $v \in \mathcal{SN}_\beta$ et $E[u[v/x]] \in \mathbb{SAT}_i(X)$. On a alors $E[u[v/x]] \in \mathcal{SN}_\beta$ par hypothèse d'induction, d'où $E[(\lambda x.u)v] \in \mathcal{SN}_\beta$ par le lemme 9.1.10.

Vérifions maintenant que les conditions ($\mathbb{SAT}1$) et ($\mathbb{SAT}2$) sont satisfaites par $\mathbb{SAT}(X)$.

($\mathbb{SAT}1$) Si $E[\ ] \in \mathcal{SN}_\beta$, on a $E[x] \in \mathbb{SAT}_0 \subseteq \mathbb{SAT}(X)$.

($\mathbb{SAT}2$) Si $E[t[u/x]] \in \mathbb{SAT}(X)$, alors il existe $i \in \mathbb{N}$ tel que $E[t[u/x] \in \mathbb{SAT}_i(X)$. Si de plus $u \in \mathcal{SN}_\beta$, alors on a $E[(\lambda x.t)u] \in \mathbb{SAT}_{i+1}(X) \subseteq \mathbb{SAT}(X)$.

On montre maintenant que pour tout $S \in \mathbb{SAT}_\beta$ tel que $X \subseteq S$, on a $\mathbb{SAT}_i(X) \subseteq S$. On raisonne par induction sur i.

Case de base. Car $S \in \mathbb{SAT}_\beta$ et $X \subseteq S$.

Cas d'induction. Si $t \in \mathbb{SAT}_{i+1}(X)$, alors soit $t \in \mathbb{SAT}_i(X)$ et $t \in S$ par hypothèse d'induction, soit $t = E[(\lambda x.u)v]$ avec $v \in \mathcal{SN}_\beta$ et $E[u[v/x]] \in \mathbb{SAT}_i(X)$. Par hypothèse d'induction $E[u[v/x]] \in S$, donc $t \in S$ car vérifie ($\mathbb{SAT}2$). □

Il s'en suit que

$$\mathbb{SAT}_\beta \quad = \quad \{\mathbb{SAT}(X) \mid X \subseteq \mathcal{SN}_\beta\}\ .$$

Montrons maintenant que $\mathbb{SAT} : \mathcal{P}(\mathcal{SN}_\beta) \to \mathcal{P}(\mathcal{SN}_\beta)$ est un opérateur de clôture.

Lemme 9.3.7 *$\mathbb{SAT} : \mathcal{P}(\mathcal{SN}_\beta) \to \mathcal{P}(\mathcal{SN}_\beta)$ est un opérateur de clôture.*

Preuve.

Idempotence. Par le lemme 9.3.6.

Extensivité. Soit $X \in \mathcal{P}(\mathcal{SN}_\beta)$. Par induction sur i, on a $X \subseteq \mathbb{SAT}_i(X)$ pour tout $i \in \mathbb{N}$, donc $X \subseteq \mathbb{SAT}(X) = \bigcup_{i\in\mathbb{N}} \mathbb{SAT}_i(X)$.

Monotonie. Soient $X \subseteq Y$ dans $\mathcal{P}(\mathcal{SN}_\beta)$. Par induction sur $i \in \mathbb{N}$, on a $\mathbb{SAT}_i(X) \subseteq \mathbb{SAT}_i(Y)$ pour tout i. On en déduit que $\mathbb{SAT}(X)_i \subseteq \bigcup_{j\in\mathbb{N}} \mathbb{SAT}_j(Y)$ pour tout i, donc que $\bigcup_{i\in\mathbb{N}} \mathbb{SAT}_i(X) \subseteq \bigcup_{j\in\mathbb{N}} \mathbb{SAT}_j(Y)$. □

Comme $\mathbb{SAT}_\beta$ est défini par l'opérateur de clôture $\mathbb{SAT} : \mathcal{P}(\mathcal{SN}_\beta) \to \mathcal{P}(\mathcal{SN}_\beta)$, selon le lemme 9.3.4 c'est un treillis complet dont l'élément maximal est $\mathcal{SN}_\beta$ et dont les plus grandes bornes inférieures sont données par les intersections. Il s'en suit que $\mathbb{SAT}_\beta$ satisfait la condition (9.18).

D'autre part, on a montré à la proposition 9.1.13 que $\mathbb{SAT}_\beta$ vérifie (9.17). De plus, la condition 9.2.7.(i) suit directement de la clause ($\mathbb{SAT}2$). Par le lemme 9.2.7, l'interprétation ($\mathbb{SAT}_\beta, B \in \mathcal{B}_0 \mapsto \mathcal{SN}_\beta$) est donc adéquate. Sa validité, quand à elle, est conséquence de la clause ($\mathbb{SAT}1$).

On a donc, par le théorème 9.2.5, la normalisation forte du système F, ce qui constitue une preuve du théorème 3.3.16. Les résultats énoncés ici se généralisent sans problèmes au λ-calcul avec produits.

Théorème 9.3.8 *Si t est un terme typable dans le système $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$, alors $t \in \mathcal{SN}_{\beta\pi}$.*

9.3.3 Ensembles saturés et réécriture

Passons maintenant au cas de la réécriture, et voyons comment prendre en compte la clause 9.2.7.(ii) dans les ensembles saturés.

Au lemme 9.1.35, on a formulé une condition suffisante sur un système de réécriture $\mathcal{R}$ pour la normalisation forte des termes typés dans $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$. En particulier, nous supposions que le type des symboles était de la forme $T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H$, où H est généré par la grammaire

$$T, U \in \mathcal{T}_\times(\mathcal{B}_0) \quad ::= \quad B \quad | \quad T \times U \ ,$$

avec $B \in \mathcal{B}_0$.

D'autre part, nous avons vu que cette condition repose sur deux propriétés de l'interprétation des types :

$$(t \in [\![T]\!] \quad \wedge \quad t \to_{\beta\pi\mathcal{R}} u) \quad \Longrightarrow \quad u \in [\![T]\!] \tag{9.19}$$

$$(\forall v. \quad f t_1 \dots t_n \to_{\beta\pi\mathcal{R}} v \quad \Longrightarrow \quad v \in [\![T]\!]) \quad \Longrightarrow \quad f t_1 \dots t_n \in [\![T]\!] \ . \tag{9.20}$$

Nous avons montré à la proposition 9.1.33 que (9.19) est satisfaite sur $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$. De plus, d'après la proposition 9.1.34, la propriété (9.20) est vérifiée par les types $T \in \mathcal{T}_\times(\mathcal{B}_0)$.

Ainsi, les ensembles saturés prennent en compte la réécriture de manière satisfaisante si on fait certaines hypothèses sur le type des symboles. Nous utiliserons cette propriété au chapitre 11.

Cependant, il est important de comprendre quelles sont les conditions de clôture qu'une famille de réductibilité doit vérifier pour pouvoir prendre en compte la réécriture de manière générale.

Pour mettre en valeur ces propriétés, il est intéressant de voir comment prendre en compte des systèmes de réécriture *polymorphes*, en particulier ceux comportant des symboles dont le type de sortie est une variable de type.

Pour cela, nous considérons des systèmes de réécriture typés dans $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$ ayant une forme particulière. Notons que tout $T \in \mathcal{T}_{2\times}(\mathcal{B}_0)$ s'écrit d'une manière unique sous la forme

$$T \quad = \quad \forall X_1 \dots X_p.\ T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H \ ,$$

où H n'est pas un type flèche et $p, n \geq 0$.

Définition 9.3.9 (Système de réécriture universellement typé) *Soit (Σ_0, τ) une signature typée telle que τ est une fonction de Σ_0 dans $\mathcal{T}_{2\times}(\mathcal{B}_0)$.*

(i) Pour tout $f \in \Sigma_0$ avec

$$\tau(f) \quad = \quad \forall X_1 \dots X_p.\ T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H \ ,$$

où H n'est pas un type flèche, le type de tête de f *est* $H_f =_{def} H$, *et* l'arité applicative de f *est* $a_f =_{def} n$.

(ii) Un système de réécriture $\mathcal{R}$ typé dans $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$ est universellement typé *si pour tout $l \mapsto_{\mathcal{R}} r$, on a $l = f l_1 \dots l_k$ avec $f \in \Sigma_0$ et $k \leq a_f$.*

Notons que tout système de réécriture typé dans une signature basée sur les produits est universellement typé.

Exemple 9.3.10 *Par exemple, en typant le symbole* id *par* $\tau(\mathsf{id}) = \forall X.X \Rightarrow X$*, la règle de réécriture*

$$\mathsf{id}\ x \quad \mapsto_{\mathsf{id}} \quad x$$

de l'exemple 3.1.5 constitue un système de réécriture universellement typé.

Remarque 9.3.11 (Notations) *L'hypothèse que les membres gauches* l *sont de la forme* $l = f l_1 \dots l_k$ *avec*

$$\tau(f) \quad = \quad \forall \vec{X}.T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H_f \quad \textit{et} \quad k \leq n$$

peut s'énoncer

$$l \quad = \quad f l_1 \dots l_k \qquad \textit{avec} \qquad \tau(f) \quad = \quad \forall \vec{X}.T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow H\ ,$$

où H *n'est pas nécessairement* H_f*. Notons que l'on a toujours* $k \leq a_f$*.*

On a besoin de la généralisation suivante des propriétés (9.19) et (9.20) : pour tout ensemble saturé S,

$$(t \in S \quad \wedge \quad t \to_{\beta\pi\mathcal{R}} u) \quad \Longrightarrow \quad u \in S \tag{9.21}$$

$$(\forall v.\ \ f t_1 \dots t_{a_f} \to_{\beta\pi\mathcal{R}} v \quad \Longrightarrow \quad v \in S) \quad \Longrightarrow \quad f t_1 \dots t_{a_f} \in S\ . \tag{9.22}$$

Ces deux propriétés ne sont en général pas des propriétés des ensembles saturés. Rappelons que même dans le cas de la β-réduction seule, les ensembles saturés ne sont en général pas stables par réduction, (voir par exemple [Wer94]).

Exemple 9.3.12 *Considérons les termes* $t =_{def} \lambda x.(\lambda y.y)x$ *et* $u =_{def} \lambda x.x$*. On a* $t \to_\beta u$*, mais* $u \notin \mathcal{SAT}(\{t\})$ *et* $t \notin \mathcal{SAT}(\{u\})$*.*

Preuve. Il suffit de remarquer que pour tout $X \subseteq \mathcal{SN}_\beta$, tous les termes de la forme $\lambda x.v$ de $\mathcal{SAT}(X)$ sont déjà dans X. Cela se montre aisément par induction sur la définition de $\mathcal{SAT}$ donnée en 9.3.5. □

Une manière de faire pourrait être de rajouter ces clauses à la définition des ensembles saturés. On aurait alors $S \in \mathcal{SAT}_{\beta\pi\mathcal{R}}$ si et seulement si $S \subseteq \mathcal{SN}_{\beta\pi\mathcal{R}}$ et

($\mathcal{SAT}0$) si $t \in S$ et $t \to_{\beta\pi\mathcal{R}} u$ alors $u \in S$,

($\mathcal{SAT}1$) si $E[\ ] \in \mathcal{SN}_{\beta\pi\mathcal{R}}$ et $x \in \mathcal{X}$ alors $E[x] \in S$,

($\mathcal{SAT}2_\beta$) si $E[t[u/x]] \in S$ et $u \in \mathcal{SN}_{\beta\pi\mathcal{R}}$ alors $E[(\lambda x.t)u] \in S$,

($\mathcal{SAT}2_{\times i}$) si $t_{3-i} \in \mathcal{SN}_{\beta\pi\mathcal{R}}$ et $E[t_i] \in S$ alors $E[\pi_i(t_1, t_2)] \in S$,

($\mathcal{SAT}2_\mathcal{R}$) si $\forall v.\ E[f t_1 \dots t_{a_f}] \to_{\beta\pi\mathcal{R}} v \implies v \in S$ alors $E[f t_1 \dots t_{a_f}] \in S$.

Ce type d'ensembles saturés fonctionne bien, mais leur formulation n'est pas très uniforme.

9.3.4 Candidats de réductibilité

Nous allons maintenant voir une famille de réductibilité qui permet de prendre en compte la propriété (9.22) de manière directe. Il s'agit des candidats de réductibilité de Girard [Gir72]. Historiquement, c'est avec cette famille de réductibilité que la normalisation forte du système F a été montrée pour la première fois.

Pour essayer de comprendre comment les candidats de Girard sont définis, nous allons partir des propriétés vues à la section 9.3.3 au sujet de la réécriture. Considérons donc un ensemble $C \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ tel que

($\mathfrak{CR}0$) si $t \in C$ et $t \to_{\beta\mathcal{R}} u$ alors $u \in C$,

($\mathfrak{CR}1_{\mathcal{R}}$) si $\forall v.\ E[ft_1 \dots t_{a_f}] \to_{\beta\mathcal{R}} v \implies v \in C$ alors $E[ft_1 \dots t_{a_f}] \in C$.

Essayons de voir quelle pourrait être la clause qui permette de prendre en compte les variables (comme ($\mathfrak{SAT}1$)) et la β-réduction (comme ($\mathfrak{SAT}2$)).

Pour cela, on se base sur la propriété qu'ont certains termes de ne pas interagir avec les contextes d'élimination. Cette propriété a été utilisée avec les ensembles saturés à la section 9.1 :

$$\begin{aligned}
\forall v.\ E[x] \to_{\beta\pi\mathcal{R}} v &\implies \big(v = E'[x] \text{ avec } E[\,] \to_{\beta\pi\mathcal{R}} E'[\,]\big) \qquad (9.23)\\
\forall v.\ E[(\lambda x.t)u] \to_{\beta\pi\mathcal{R}} v &\implies \big(v = E'[t'] \text{ avec } (E[\,], (\lambda x.t)u) \to_{\beta\pi\mathcal{R}} (E'[\,], t')\big)\\
\forall v.\ E[\pi_i(t_1, t_2)] \to_{\beta\pi\mathcal{R}} v &\implies \big(v = E'[t'] \text{ avec } (E[\,], \pi_i(t_1, t_2)) \to_{\beta\pi\mathcal{R}} (E'[\,], t')\big)\ .
\end{aligned}$$

Notons que dans le cas des systèmes de réécriture universellement typés, ceci est aussi vérifié pour les symboles qui ont autant d'arguments que leur arité applicative :

$$\forall v.\ E[ft_1 \dots t_{a_f}] \to_{\beta\pi\mathcal{R}} v \implies \big(v = E'[t'] \text{ avec } (E[\,], ft_1 \dots t_{a_f}) \to_{\beta\pi\mathcal{R}} (E'[\,], t')\big)\ .$$

Les termes qui n'interagissent pas avec les contextes d'élimination sont dits *neutres*. Les candidats de réductibilité reposent sur le fait que comme les termes neutres n'interagissent pas avec les contextes d'élimination, leurs propriétés de réductibilité peuvent être montrées par induction sur $\mathcal{SN}$.

D'autre part, pour avoir une famille de réductibilité valide au sens de la définition 9.2.4, il faut que chacun de ses éléments contienne les variables. Pour les candidats de réductibilité, cela provient de la propriété (9.23), qui assure que les variables sont des termes neutres.

Notre but est ici d'avoir une notion de candidats de réductibilité la plus générale possible, qui ne dépende que d'une relation de réécriture $\to_R$ et d'un ensemble de « contextes d'élimination » $\mathcal{E}$. Nous voulons donc laisser le maximum de liberté possible pour le choix des ces contextes, tout en assurant la propriété (9.23). Notre notion de terme neutre est plus générale que celle des définitions usuelles [GLT89, Gal89].

Définition 9.3.13 (Termes neutres) *Soit une variable* $[\,]$ *et un ensemble* $\mathcal{E}$ *de termes* $E[\,] \in \Lambda(\Sigma)$ *qui est clos par composition :*

$$E[\,], F[\,] \in \mathcal{E} \implies E[F[\,]] \in \mathcal{E}\ ,$$

où $E[t] =_{def} E[\,][t/[\,]]$.

— *Un terme* $t \in \Lambda(\Sigma)$ *est* pré-neutre *dans* $\mathcal{E}$ *pour une relation de réécriture* $\to_R$ *si pour tout* $E[\,] \in \mathcal{E}$,

$$\forall v.\quad E[t] \to_R v \quad\Longrightarrow\quad \big(v = E'[t'] \quad \textit{avec} \quad E'[\,] \in \mathcal{E} \ \textit{ et } \ (E[\,], t) \to_R (E'[\,], t')\big)\ .$$

— $\mathcal{E}$ *est un ensemble de* contextes d'élimination *sur* $\Lambda(\Sigma)$ *pour une relation de réécriture* $\to_R$ *sur* $\Lambda(\Sigma)$ *si*

(i) $\mathcal{E}$ *est stable par* R*-réduction :* $E[\,] \in \mathcal{E}$ *et* $E[\,] \to_R F[\,]$ *implique* $F[\,] \in \mathcal{E}$,

(ii) toutes les variables sont pré-neutres,

(iii) pour tout terme t *pré-neutre et pour tout* $E[\,] \in \mathcal{E}$, $E[t]$ *est pré-neutre dans* $\mathcal{E}$ *pour* $\to_R$.

— *Si* $\mathcal{E}$ *est un ensemble de contextes d'élimination pour* $\to_R$, *alors les termes pré-neutres dans* $\mathcal{E}$ *pour* $\to_R$ *sont dits* neutres *dans* $\mathcal{E}$ *pour* $\to_R$. *On désigne l'ensemble des termes neutre pour* $\to_R$ *dans* $\mathcal{E}$ *par* $\mathcal{N}_{R\mathcal{E}}$.

Remarque 9.3.14 *Les contextes d'élimination ne sont donc pas des contextes au sens de la définition 1.2.18. En effet, même si le « trou »* $[\,]$ *apparaît sous un lieur dans* $E[\,]$, *alors ce lieur ne lie aucune variable libre de* t *dans* $E[t]$ *car* $E[t] = E[\,][t/[\,]]$.

Ceci permet de voir les contextes d'élimination comme des termes de $\Lambda(\Sigma)$, *et en particulier de dire s'ils sont fortement normalisant ou non.*

Dans le cadre de la réductibilité pour la normalisation forte, nous ne connaissons pas de cas dans lequel on ait besoin de contextes d'élimination pouvant lier des variables.

Ils nous semble d'ailleurs, mais ce n'est qu'une intuition, que des contextes d'élimination pouvant lier des variables ne seraient peut-être pas utilisables dans des preuves de normalisation forte. *Cette intuition nous provient de la remarque 10.7.7.*

Lorsque le contexte le permet, on désigne $\mathcal{N}_{R\mathcal{E}}$ par $\mathcal{N}_R$ ou même par $\mathcal{N}$.

Remarque 9.3.15 *Dans le cas du* λ*-calcul pur et du* λ*-calcul avec produits, en prenant comme contextes d'élimination respectivement* $\mathcal{E}_\Rightarrow$ *et* $\mathcal{E}_{\Rightarrow\times}$, *notre définition correspond à la définition usuelle (voir par exemple [GLT89]), et pour la réécriture, notre notion est très proche de celle de [Bla05b]. En effet, nos termes neutres sont exactement les termes qui ne sont pas de la forme*

— $\lambda x.t$, *car* $[(\lambda x.t)]u \mapsto_\beta t[u/x]$,

— (t_1, t_2), *car* $\pi_i[(t_1, t_2)] \mapsto_\pi t_i$,

— $f\vec{t}$ *tels qu'il existe une règle* $f\vec{l} \mapsto_{\mathcal{R}} r$, *une substitution* σ, *et des termes* $\vec{u}$ *avec* $|\vec{u}| > 0$ *tels que* $\vec{t}\vec{u} = \vec{l}\sigma$, *car* $[f\vec{t}]\vec{u} \mapsto_{\mathcal{R}} r\sigma$.

Dans [Bla05b], un terme de la forme $f\vec{t}$ *n'est pas neutre s'il existe une règle* $f\vec{l}$ *telle que* $|\vec{t}| < |\vec{l}|$, *et ce même s'il n'existe pas de termes* $\vec{u}$ *ni de substitution* σ *tels que* $\vec{t}\vec{u} = \vec{l}\sigma$.

Notons que dans le cas de $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$, *la propriété (9.23) assure que* $\mathcal{E}_{\Rightarrow\times}$ *est un ensemble de contextes d'élimination pour* $\to_{\beta\pi\mathcal{R}}$.

Définition 9.3.16 (Candidats de réductibilité) *Étant donnés une relation de réécriture* $\to_R$ *sur* $\Lambda(\Sigma)$ *et un ensemble de contextes d'élimination* $\mathcal{E}$ *pour* $\to_R$, *l'ensemble* $\mathfrak{CR}_{R\mathcal{E}}$ *des* candidats de réductibilité pour R dans $\mathcal{E}$ *est l'ensemble des* $C \subseteq \mathcal{SN}_R$ *tels que*

($\mathcal{CR}0$) *si* $t \in C$ *et* $t \to_R u$ *alors* $u \in C$,

($\mathcal{CR}1$) *si* $t \in \mathcal{N}_{R\mathcal{E}}$ *et* $\forall u.\ t \to_R u \implies u \in C$ *alors* $t \in C$.

Lorsque que le contexte le permet, on désigne $\mathcal{CR}_{R\mathcal{E}}$ par $\mathcal{CR}_R$ ou même par $\mathcal{CR}$.

Exemple 9.3.17 *Soit un ensemble de types de base* $\mathcal{B}$ *et une signature* Σ.

(i) Les candidats de réductibilité de $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B})$*, notés* $\mathcal{CR}_\beta$*, sont les candidats de réductibilité pour* $\to_\beta$ *dans*

$$E[\,] \in \mathcal{E}_{\Rightarrow} \quad ::= \quad [\,] \quad | \quad E[\,]\,t\ ,$$

où $t \in \Lambda(\Sigma)$.

(ii) Les candidats de réductibilité de $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B})$*, notés* $\mathcal{CR}_{\beta\pi}$*, sont les candidats de réductibilité pour* $\to_{\beta\pi}$ *dans*

$$E[\,] \in \mathcal{E}_{\Rightarrow\times} \quad ::= \quad [\,] \quad | \quad E[\,]\,t \quad | \quad \pi_1\,E[\,] \quad | \quad \pi_2\,E[\,]\ ,$$

où $t \in \Lambda(\Sigma)$.

(iii) Soit $\mathcal{R}$ *un TRS universellement typé dans* $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$*. On désigne par* $\mathcal{CR}_{\beta\pi\mathcal{R}}$ *l'ensemble des candidats de réductibilité pour* $\to_{\beta\pi\mathcal{R}}$ *dans* $\mathcal{E}_{\Rightarrow\times}$.
Notons que les termes neutres de $\mathcal{CR}_{\beta\pi\mathcal{R}}$ *sont ceux donnés à la remarque 9.3.15.*
De plus, les candidats $C \in \mathcal{CR}_{\beta\pi\mathcal{R}}$ *vérifient* ($\mathcal{CR}1_{\mathcal{R}}$)*. En effet, les membres gauches de règles définissant un symbole* $f \in \Sigma_0$ *sont de la forme* $f l_1 \dots l_k$ *avec* $k \leq a_f$*. Il s'en suit que tout terme de la forme* $f t_1 \dots t_{a_f}$ *est neutre, d'où, par* ($\mathcal{CR}1$)*,* $f t_1 \dots t_{a_f} \in C$ *si* $(f t_1 \dots t_{a_f})_{\beta\pi\mathcal{R}} \subseteq C$.

Vérifions maintenant que les candidats de réductibilité $C \in \mathcal{CR}_{\beta\pi\mathcal{R}}$ satisfont les clauses ($\mathcal{SAT}1$), ($\mathcal{SAT}2_\beta$) et ($\mathcal{SAT}2_{\pi i}$). Les preuves sont similaires à celles que l'on a vues à la section 9.1 pour la propriété $\mathcal{SN} \in \mathcal{SAT}$. Il est intéressant, pour la suite, de le montrer dans un cadre un peu plus général.

Lemme 9.3.18 *Soit* $\to_R$ *une relation de réécriture sur* $\Lambda(\Sigma)$.

($\mathcal{SAT}1$) *Soit* $\mathcal{E}$ *un ensemble de contextes d'élimination pour* $\to_R$*. Pour tout* $C \in \mathcal{CR}_{R\mathcal{E}}$*, si* $E[\,] \in \mathcal{E} \cap \mathcal{SN}_R$ *et* $x \in \mathcal{X}$ *alors* $E[x] \in C$.

($\mathcal{SAT}2_\beta$) *Soit* $\mathcal{E}$ *un ensemble de contextes d'élimination pour* $\to_{\beta R}$ *tel que* $(\lambda x.t)u \in \mathcal{N}_{\beta R\mathcal{E}}$ *pour tout* t, u*. Pour tout* $C \in \mathcal{CR}_{\beta R\mathcal{E}}$ *et tout* $E[\,] \in \mathcal{E}$*, si* $E[t[u/x]] \in C$ *et* $u \in \mathcal{SN}_{\beta R}$ *alors* $E[(\lambda x.t)u] \in C$,

($\mathcal{SAT}2_{\pi i}$) *Soit* $\mathcal{E}$ *un ensemble de contextes d'élimination pour* $\to_{\pi R}$ *tel que pour tout* $i \in \{1,2\}$ *et tout* t_1, t_2*, on ait* $\pi_i(t_1, t_2) \in \mathcal{N}_{\pi R\mathcal{E}}$*. Pour tout* $C \in \mathcal{CR}_{\pi R\mathcal{E}}$ *et tout* $E[\,] \in \mathcal{E}$*, si* $E[t_i] \in C$ *et* $t_{3-i} \in \mathcal{SN}_{\pi R}$ *alors* $E[\pi_i(t_1, t_2)] \in C$,

Preuve.

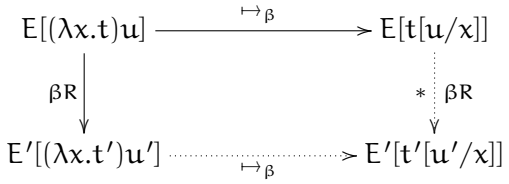
($\mathcal{SAT}1$) Par définition, les termes de la forme $E[x]$ avec $E[\,] \in \mathcal{E}$ vérifient

$$\forall v.\quad E[x] \to_R v \quad \implies \quad \big(v = E'[x] \quad \text{avec} \quad E'[\,] \in \mathcal{E} \quad \text{et} \quad E[\,] \to_R E'[\,]\big)\ .$$

Ce sont donc des termes neutres dont tous les réduits sont neutres, et on a le résultat recherché en raisonnant par induction sur $E[\,] \in \mathcal{SN}_R$.

($\mathbb{SAT}2_\beta$) On doit montrer que si $E[t[u/x]] \in C$ avec $u \in \mathcal{SN}_{\beta R}$ alors $E[(\lambda x.t)u] \in C$.

La preuve est la même qu'aux lemmes 9.1.10 et 9.1.24, en utilisant la standardisation faible, qui est un conséquence du fait que $(\lambda x.t)u \in \mathcal{N}_{\beta R\mathcal{E}}$:



Notons que $E[\ ], t, u \in \mathcal{SN}_{\beta R}$. On raisonne par induction sur les triplets $(E[\ ], t, u)$ ordonnés par l'extension produit de $\to_{\beta R}$. Comme $E[(\lambda x.t)u] \in \mathcal{N}_{\beta R\mathcal{E}}$, d'après ($\mathbb{CR}1$) il suffit de montrer que $E[(\lambda x.t)u] \to_{\beta R} v$ implique $v \in C$.

Selon la standardisation faible, il y a deux possibilités :

— $v = E[t[u/x]] \in C$ par hypothèse,

— $v = E'[(\lambda x.t')u']$ avec $E'[\ ] \in \mathcal{E}$, $(E[\ ], t, u) \to_{\beta R} (E'[\ ], t', u')$ et $E[t[u/x]] \to^*_{\beta R} E'[t'[u'/x]]$. On a $E'[t'[u'/x]] \in C$ car C est stable par réduction (c'est la clause ($\mathbb{CR}0$)). De plus, on a $(\lambda x.t')u' \mapsto_\beta t'[u/x]$ et $v \in \mathcal{N}_{\beta R\mathcal{E}}$, donc $v \in C$ par hypothèse d'induction.

($\mathbb{SAT}2_{\pi i}$) Similaire. □

Remarque 9.3.19 *Nous avons vu à la section 9.1 qu'un des rôles importants des familles de réductibilité est d'assurer la préservation de certaines propriétés par expansion faible de tête dans les contextes d'élimination.*

Les ensembles saturés sont minimaux par rapport à cette requête, car ils n'assurent que l'expansion de tête dans les contextes d'élimination.

D'un autre coté, lorsque que l'on considère un système de réécriture arbitraire, cette notion d'expansion de tête est trop restrictive et on a besoin d'expansion générale dans les contextes d'élimination. C'est ce que font les candidats de Girard et ce pourquoi ils sont plus faciles à manier en présence de réécriture.

Comme avec les ensembles saturés, vérifier que $\mathbb{CR}$ n'est pas vide revient à vérifier que les candidats de réductibilité induisent des interprétations valides. Pour le montrer, il est intéressant d'utiliser le fait que $\mathbb{CR}$ peut être défini par un opérateur de clôture sur $\mathcal{P}(\mathcal{SN})$, ce qui nous donnera aussi la stabilité par intersection.

À la différence de $\mathbb{SAT}(\_)$, l'opérateur de clôture des candidats de réductibilité n'est pas toujours définissable par induction sur $\mathbb{N}$. Rappelons que par la définition 1.1.1, les signatures que nous considérons sont dénombrables. Il s'en suit que $\Lambda(\Sigma)$ est un ensemble dénombrable, donc que l'opérateur de clôture peut être défini par induction sur les ordinaux dénombrables. On considère donc un ensemble ordonné $(\mathfrak{O}, \leq)$ vérifiant les axiomes des ordinaux dénombrables (voir [Gal91]).

Définition 9.3.20 *Soient une relation de réécriture $\to_R$ sur $\Lambda(\Sigma)$ et un ensemble de contextes d'élimination $\mathcal{E}$ pour $\to_R$.*

— *On définit la fonction* $\mathsf{CR} : \mathcal{P}(\mathcal{SN}_\mathsf{R}) \to \mathcal{P}(\mathcal{SN}_\mathsf{R})$ *comme suit :*

$$\mathsf{CR}(X) \quad =_{def} \quad X \cup \{t \in \mathcal{N}_{\mathsf{R}\mathcal{E}} \mid \forall u.\ \ t \to_\mathsf{R} u \implies u \in X\}\ .$$

— *On définit les fonctions* $\mathcal{CR}_\mathfrak{a} : \mathcal{P}(\mathcal{SN}_\mathsf{R}) \to \mathcal{P}(\mathcal{SN}_\mathsf{R})$ *par induction sur* $\mathfrak{a} \in \mathfrak{O}$ *comme suit :*

$$\begin{array}{lll} \mathcal{CR}_0(X) & =_{def} & (X)^*_\mathsf{R} \\ \mathcal{CR}_{\mathfrak{a}+1}(X) & =_{def} & \mathsf{CR}(\mathcal{CR}_\mathfrak{a}(X)) \\ \mathcal{CR}_\lambda(X) & =_{def} & \bigcup_{\mathfrak{a}<\lambda} \mathcal{CR}_\mathfrak{a}(X) \quad \textit{si } \lambda \textit{ est un ordinal limite}\ . \end{array}$$

Il est clair que CR est une fonction monotone sur $\mathcal{P}(\mathcal{SN}_\mathsf{R})$. Pour tout $X \in \mathcal{P}(\mathcal{SN}_\mathsf{R})$, elle est donc monotone sur le treillis complet $\{(Y)^*_\mathsf{R} \mid X \subseteq Y \subseteq \mathcal{SN}_\mathsf{R}\}$, sur lequel elle a donc un plus petit point fixe. Vérifions que ce plus petit point fixe est $\mathcal{CR}_{\mathfrak{p}_X}(X)$ pour un ordinal dénombrable $\mathfrak{p}_X \in \mathfrak{O}$.

Proposition 9.3.21 *Pour tout* $X \subseteq \mathcal{SN}_\mathsf{R}$*, il existe* $\mathfrak{p}_X \in \mathfrak{O}$ *tel que* $\mathcal{CR}_{\mathfrak{p}_X}(X)$ *soit le plus petit point fixe de* CR *dans* $\{(Y)^*_\mathsf{R} \mid X \subseteq Y \subseteq \mathcal{SN}_\mathsf{R}\}$.

Preuve. Supposons que pour tout $\mathfrak{a}, \mathfrak{b} \in \mathfrak{O}$, on ait $\mathcal{CR}_\mathfrak{a}(X) \subsetneq \mathcal{CR}_\mathfrak{b}(X)$ si $\mathfrak{a} < \mathfrak{b}$. Comme $\mathfrak{O}$ n'est pas dénombrable, on aurait un ensemble non dénombrable $\{t_\mathfrak{b} \mid \mathfrak{b} \in \mathfrak{O}\} \subseteq \Lambda(\Sigma)$ d'éléments deux à deux distincts, ce qui est impossible car $\Lambda(\Sigma)$ est lui-même un ensemble dénombrable.

Il s'en suit que le plus petit point fixe de CR dans $\{(Y)^*_\mathsf{R} \mid X \subseteq Y \subseteq \mathcal{SN}_\mathsf{R}\}$ est atteint pour un ordinal de $\mathfrak{O}$. □

D'autre part, si $t \in \mathcal{CR}_\mathfrak{a}(X)$ avec $\mathfrak{a}$ aussi petit que possible, alors $\mathfrak{a}$ est soit 0, soit de la forme $\mathfrak{b} + 1$. En effet, si $t \in \mathcal{CR}_\lambda(X)$ où λ est un ordinal limite, alors par définition de $\mathcal{CR}_\lambda(X)$ il existe $\mathfrak{a} < \lambda$ tel que $t \in \mathcal{CR}_\mathfrak{a}(X)$.

Définition 9.3.22 *Soient une relation de réécriture* $\to_\mathsf{R}$ *sur* $\Lambda(\Sigma)$ *et un ensemble de contextes d'élimination* $\mathcal{E}$ *pour* $\to_\mathsf{R}$*. On définit la fonction* $\mathcal{CR} : \mathcal{P}(\mathcal{SN}_\mathsf{R}) \to \mathcal{P}(\mathcal{SN}_\mathsf{R})$ *par* $\mathcal{CR}(X) =_{def} \mathcal{CR}_{\mathfrak{p}_X}(X)$*, où* $\mathcal{CR}_{\mathfrak{p}_X}(X)$ *est le plus petit point fixe de* CR *dans* $\{(Y)^*_\mathsf{R} \mid X \subseteq Y \subseteq \mathcal{SN}_\mathsf{R}\}$.

Montrons que $\mathcal{CR}$ est l'opérateur de clôture qui définit les candidats de réductibilité.

Lemme 9.3.23 *Si* $X \subseteq \mathcal{SN}_\mathsf{R}$ *alors* $\mathcal{CR}(X)$ *est le plus petit candidat de réductibilité pour* $\to_\mathsf{R}$ *dans* $\mathcal{E}$ *contenant* X.

Preuve. Commençons par montrer que $\mathcal{CR}(X) \in \mathcal{CR}_{\mathsf{R}\mathcal{E}}$ pour tout $X \subseteq \mathcal{SN}_\mathsf{R}$. Tout d'abord, comme $\mathsf{CR} : \mathcal{P}(\mathcal{SN}_\mathsf{R}) \to \mathcal{P}(\mathcal{SN}_\mathsf{R})$, il est clair que $\mathcal{CR}(X) \subseteq \mathcal{SN}_\mathsf{R}$.

Vérifions maintenant que les conditions ($\mathcal{CR}0$) et ($\mathcal{CR}1$) sont satisfaites par $\mathcal{CR}(X)$.

($\mathcal{CR}0$) Soit $t \to_\mathsf{R} u$ et $t \in \mathcal{CR}_\mathfrak{a}(X)$ avec $\mathfrak{a}$ aussi petit que possible. Si $\mathfrak{a} = 0$ alors on a $u \in (X)^*_\mathsf{R} = \mathcal{CR}_0$. Sinon, $\mathfrak{a} = \mathfrak{b} + 1$, $t \in \mathcal{N}_{\mathsf{R}\mathcal{E}}$ et $u \in \mathcal{CR}_\mathfrak{b}(X)$ par définition.

($\mathcal{CR}1$) Soit $t \in \mathcal{N}_{\mathsf{R}\mathcal{E}}$ avec $(t)_\mathsf{R} \subseteq \mathcal{CR}(X)$. Alors on a $(t)_\mathsf{R} \subseteq \mathcal{CR}_\mathfrak{a}(X)$ pour un $\mathfrak{a} < \mathfrak{p}_X$, d'où $t \in \mathcal{CR}_{\mathfrak{a}+1}(X)$.

On montre maintenant par induction sur $\mathfrak{a}$ que $\mathcal{CR}_\mathfrak{a}(X) \subseteq C$ pour tout $C \in \mathcal{CR}_{R\mathcal{E}}$ contenant X. Par ($\mathcal{CR}0$), on a $\mathcal{CR}_0(X) = (X)^*_R \subseteq C$, et si $t \in \mathcal{CR}_{\mathfrak{a}+1}(X) \setminus \mathcal{CR}_\mathfrak{a}(X)$ alors $t \in \mathcal{N}_{R\mathcal{E}}$ et par hypothèse d'induction $(t)_R \subseteq \mathcal{CR}_\mathfrak{a}(X) \subseteq C$, donc $t \in C$ par ($\mathcal{CR}1$). De plus, si λ est un ordinal limite, alors par hypothèse d'induction $\mathcal{CR}_\mathfrak{a}(X) \subseteq C$ pour tout $\mathfrak{a} < \lambda$, donc $\mathcal{CR}_\lambda(X) \subseteq C$. □

Il s'en suit que $\mathcal{CR}_{R\mathcal{E}} = \{\mathcal{CR}(X) \mid X \subseteq \mathcal{SN}_R\}$. Montrons maintenant que $\mathcal{CR}(\_)$ est un opérateur de clôture.

Lemme 9.3.24 *$\mathcal{CR} : \mathcal{P}(\mathcal{SN}_R) \to \mathcal{P}(\mathcal{SN}_R)$ est un opérateur de clôture.*

Preuve. L'idempotence suit du lemme 9.3.23, l'extensivité et la monotonie se montrent comme au lemme 9.3.7. □

Comme $\mathcal{CR}_{R\mathcal{E}}$ est défini par l'opérateur de clôture $\mathcal{CR} : \mathcal{P}(\mathcal{SN}_R) \to \mathcal{P}(\mathcal{SN}_R)$, selon le lemme 9.3.4 c'est un treillis complet dont l'élément maximal est $\mathcal{SN}_R$ et dont les plus grandes bornes inférieures sont données par les intersections. Il s'en suit que $\mathcal{CR}_{R\mathcal{E}}$ satisfait la condition (9.18).

L'élément minimal de $\mathcal{CR}_{R\mathcal{E}}$ a une forme intéressante.

Définition 9.3.25 (Termes héréditairement neutres) *On définit $\mathcal{HN}_{R\mathcal{E}}$,* l'ensemble des termes héréditairement neutres *pour $\to_R$ dans $\mathcal{E}$ comme étant le plus petit ensemble tel que*

$$\forall t \in \mathcal{N}_{R\mathcal{E}}.\quad (\forall u.\ t \to_R u \implies u \in \mathcal{HN}_{R\mathcal{E}}) \implies t \in \mathcal{HN}_{R\mathcal{E}}\ .$$

L'ensemble $\mathcal{HN}_{R\mathcal{E}}$ n'est pas vide : on a par exemple $\mathcal{X} \subseteq \mathcal{HN}_{R\mathcal{E}}$.

Remarque 9.3.26 *Ainsi, un terme est héréditairement neutre si et seulement si c'est un terme neutre fortement normalisant dont tous les réduits sont neutres.*

Exemple 9.3.27 *Dans le cas du λ-calcul pur, les termes héréditairement neutres sont exactement les termes de la forme $E[x]$ avec $E[\ ] \in \mathcal{E}_\Rightarrow \cap \mathcal{SN}_\beta$ et $x \in \mathcal{X}$.*

Proposition 9.3.28 *$\mathcal{HN}_{R\mathcal{E}}$ est le plus petit élément de $\mathcal{CR}_{R\mathcal{E}}$.*

Preuve. Par le lemme 9.3.4, le plus petit élément de $\mathcal{CR}_{R\mathcal{E}}$ est la clôture de $\emptyset \in \mathcal{P}(\mathcal{SN}_R)$. Or, on a $\mathcal{HN}_{R\mathcal{E}} = \mathcal{CR}(\emptyset)$ car $\mathcal{CR}(\emptyset)$ est le plus petit élément de $\mathcal{P}(\mathcal{SN}_R)$ tel que

$$\mathcal{CR}(\emptyset) \quad = \quad \mathcal{CR}(\emptyset) \cup \{t \in \mathcal{N}_{R\mathcal{E}} \mid \forall u.\quad t \to_R u \quad \implies \quad u \in \mathcal{CR}(\emptyset)\}\ .$$

□

Remarque 9.3.29 *Dans le cas du λ-calcul pur, il suit de l'exemple 9.3.27 que $\mathcal{HN}_\beta$ est aussi le plus petit élément de $\mathbb{SAT}_\beta$.*

Enfin, les termes héréditairement neutres sont stables par contextes d'élimination fortement normalisants.

Proposition 9.3.30 *Si* $t \in \mathcal{HN}_{R\mathcal{E}}$ *et* $E[\,] \in \mathcal{E} \cap \mathcal{SN}_R$ *alors* $E[t] \in \mathcal{HN}_{R\mathcal{E}}$.

PREUVE. On raisonne par induction sur les paires $(E[\,], t)$ ordonnées par l'extension produit de $\to_R$.

Soit donc $t \in \mathcal{HN}_{R\mathcal{E}}$ et $E[\,] \in \mathcal{E} \cap \mathcal{SN}_R$. Comme $E[t] \in \mathcal{N}_{R\mathcal{E}}$, on a $E[t] \in \mathcal{HN}_{R\mathcal{E}}$ si $(E[t])_R \subseteq \mathcal{HN}_{R\mathcal{E}}$. Soit v tel que $E[t] \to_R v$. Alors comme t est neutre, on a $v = E'[t']$ avec $(E[\,], t) \to_R (E'[\,], t')$. De plus, $t' \in \mathcal{HN}_{R\mathcal{E}}$ car $t \in \mathcal{HN}_{R\mathcal{E}}$, donc $v \in \mathcal{HN}_{R\mathcal{E}}$ par hypothèse d'induction. □

Voyons maintenant comment les candidats de réductibilité se comportent par rapport aux interprétations de la flèche et du produit.

Proposition 9.3.31 *Soit* $\mathcal{E}$ *un ensemble de contextes d'élimination pour* $\to_R$ *une relation de réécriture finiment branchante sur* $\Lambda(\Sigma)$.

(i) Si $[\,]t \in \mathcal{E}$ *pour tout* $t \in \Lambda(\Sigma)$ *alors,*

$$C_1, C_2 \in \mathfrak{CR}_{R\mathcal{E}} \quad \Longrightarrow \quad C_2 \Rightarrow C_1 \in \mathfrak{CR}_{R\mathcal{E}} \ .$$

(ii) Si $\pi_i[\,] \in \mathcal{E}$ *pour tout* $i \in \{1,2\}$ *alors,*

$$C_1, C_2 \in \mathfrak{CR}_{R\mathcal{E}} \quad \Longrightarrow \quad C_1 \times C_2 \in \mathfrak{CR}_{R\mathcal{E}} \ .$$

PREUVE. Tout d'abord, on a $C_2 \Rightarrow C_1, C_1 \times C_2 \subseteq \mathcal{SN}_R$ exactement pour les mêmes raisons qu'à la proposition 9.1.25, en utilisant le fait que $\mathcal{X} \subseteq \mathcal{HN}_{R\mathcal{E}} \in \mathfrak{CR}_{R\mathcal{E}}$. Vérifions maintenant que $C_2 \Rightarrow C_1$ et $C_1 \times C_2$ satisfont ($\mathfrak{CR}0$) et ($\mathfrak{CR}1$).

On ne traite que le cas de $\_ \Rightarrow \_$, celui de $\_ \times \_$ étant similaire et plus simple.

($\mathfrak{CR}0$) Si $t \in C_2 \Rightarrow C_1$ et $t \to_R u$, alors $tv \in C_1$ pour tout $v \in C_2$, donc $uv \in C_1$ par ($\mathfrak{CR}0$). Il s'en suit que $u \in C_2 \Rightarrow C_1$.

($\mathfrak{CR}1$) Supposons que $t \in \mathcal{N}_{R\mathcal{E}}$ soit tel que $(t)_R \subseteq C_2 \Rightarrow C_1$. Soit $v \in C_2$ et montrons que $tv \in C_1$. Comme $tv \in \mathcal{N}_{R\mathcal{E}}$, par ($\mathfrak{CR}1$) il suffit de montrer que $(tv)_R \subseteq C_2$. On raisonne par induction sur $v \in \mathcal{SN}_R$. Soit w tel que $tv \to_R w$. Comme t est neutre et $[\,]v \in \mathcal{E}$, on a $w = t'v'$ avec $(t, v) \to_R (t', v')$.

— Si $t = t'$, alors $v \to_R v'$. Par ($\mathfrak{CR}0$) on a $v' \in C_2$ et par hypothèse d'induction $tv' \in C_1$.

— Sinon, $t \to_R t'$ et $v = v'$, mais alors $t'v \in C_1$ car $(t)_R \subseteq C_2 \Rightarrow C_1$. □

On a maintenant tout ce qu'il faut pour avoir l'adéquation des interprétations basées sur $\mathfrak{CR}_{\beta\pi\mathcal{R}}$.

— La proposition 9.3.31 implique que $\_ \Rightarrow \_, \_ \times \_ \in \mathfrak{CR}^2_{\beta\pi\mathcal{R}} \to \mathfrak{CR}_{\beta\pi\mathcal{R}}$, et il suit du lemme 9.3.24 que $\bigcap \mathcal{R} \in \mathfrak{CR}_{\beta\pi\mathcal{R}}$ pour tout $\emptyset \neq \mathcal{R} \subseteq \mathfrak{CR}_{\beta\pi\mathcal{R}}$. Les candidats de réductibilité forment donc une famille de réductibilité au sens de la définition 9.2.4.

— Le lemme 9.3.18 assure que l'interprétation $(\mathfrak{CR}_{\beta\pi\mathcal{R}}, (B \in \mathcal{B}_0 \mapsto \mathcal{SN}_{\beta\pi\mathcal{R}})\cup\_ \times \_)$ est valide, ainsi que la condition 9.2.7.(i) et la condition 9.2.7.(ii) pour les produits : pour tout T_1, T_2 et tout $\rho : \mathcal{V} \to \mathfrak{CR}_{\beta\pi\mathcal{R}}$,

$$\begin{array}{rcl} (t_1 \in [\![T_1]\!]\rho \quad \wedge \quad t_2 \in [\![T_2]\!]\rho) & \Longrightarrow & (t_1, t_2) \in [\![T_1]\!]\rho \times [\![T_2]\!]\rho \\ \forall i \in \{1,2\}. \quad t \in [\![T_1]\!]\rho \times [\![T_2]\!]\rho & \Longrightarrow & \pi_i t \in [\![T_i]\!]\rho \ . \end{array}$$

Considérons maintenant le cas de la réécriture. On suppose que $\mathcal{R}$ est universellement typé dans $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$, au sens de la définition 9.3.9 : les membres gauches sont de la forme $f l_1 \dots l_k$ avec $\tau(f) = \forall \vec{X}.T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow H$.

L'essentiel est d'adapter à ces systèmes la condition formulée au lemme 9.1.35. Une condition similaire à celle que l'on obtient est utilisée dans [Bla05b].

Lemme 9.3.32 (Calculabilité des règles de réécriture) *Soit $\mathcal{R}$ un système de réécriture universellement typé dans $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$. Supposons de plus que $\to_{\beta\pi\mathcal{R}}$ est finiment branchante et que $(\mathcal{CR}_{\beta\pi\mathcal{R}}, [\![\_]\!])$ est une interprétation telle que $[\![\_ \times \_]\!] = \_ \times \_$.*

Si pour toute règle $f l_1 \dots l_k \mapsto_{\mathcal{R}} r$ avec $\tau(f) = \forall \vec{X}.T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow H$ et toute valuation (σ, ρ) on a

$$(\forall i \in \{1, \dots, k\}. \quad l_i\sigma \in [\![T_i]\!]\rho) \quad \Longrightarrow \quad r\sigma \in [\![H]\!]\rho \ ,$$

alors les termes typables de $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0, \Sigma_0, \tau)$ sont fortement $\beta\pi\mathcal{R}$-normalisants.

Preuve. On applique le théorème 9.2.5 avec l'interprétation $(\mathcal{CR}_{\beta\pi\mathcal{R}}, [\![\_]\!])$. Il reste à monter que c'est une interprétation adéquate, et pour cela à voir que la condition 9.2.7.(ii) est satisfaite pour les symboles de Σ_0.

Soit $f \in \Sigma_0$ avec $\tau(f) = \forall \vec{X}.T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H_f$. Soient $t_1, \dots, t_n$ avec $t_i \in [\![T_i]\!]\rho$ pour tout $i \in \{1, \dots, n\}$. On doit montrer que $f t_1 \dots t_n \in [\![H_f]\!]\rho$. Comme toutes les règles définissant f ont au plus n arguments, $f t_1 \dots t_n$ est un terme neutre. Par $(\mathcal{CR}1)$, il suffit donc d'avoir $(f t_1 \dots t_n)_{\beta\pi\mathcal{R}} \subseteq [\![H_f]\!]\rho$. Le reste de la preuve suit celle du lemme 9.1.35. □

9.3.5 Biorthogonaux

Nous allons maintenant aborder une troisième famille de réductibilité. Cette famille est définie par une relation *d'orthogonalité* entre termes et contextes d'élimination. L'idée de ce type d'interprétation provient de la logique linéaire de Girard [Gir87]. En ce qui concerne la réductibilité, les biorthogonaux ont été introduits pour prendre en compte la logique classique : il sont utilisés par Parigot [Par97] pour montrer la normalisation forte du λμ-calcul du second ordre, et sont à la base de travaux de Krivine sur la réalisabilité en logique classique [DK00, Kri04].

Il s'avère que les biorthogonaux sont aussi très utiles pour capturer des propriétés calculatoires assez fines des termes typables dans certains systèmes, voir par exemple [Pit00, VM04, Vou04, MV05, Rib07b].

Essayons d'en présenter l'idée de base dans le cas du λ-calcul simplement typé. Reprenons l'interprétation des types simples donnée à la définition 9.1.5 :

$$\begin{array}{lcll} [\![B]\!] & = & \mathcal{SN}_\beta & \text{si } B \in \mathcal{B}_0 \\ [\![U \Rightarrow T]\!] & = & [\![U]\!] \Rightarrow [\![T]\!] \ . & \end{array}$$

Comme un type simple $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$ peut s'écrire sous la forme $T = T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow B$, son interprétation s'écrit $[\![T_1]\!] \Rightarrow \dots [\![T_n]\!] \Rightarrow \mathcal{SN}_\beta$, c'est à dire

$$[\![T]\!] \quad = \quad \{t \mid \forall u_1 \in [\![T_1]\!], \dots, u_n \in [\![T_n]\!]. \quad t u_1 \dots u_n \in \mathcal{SN}_\beta\} \ .$$

Il s'en suit qu'en associant à T l'ensemble de contextes d'élimination

$$[\![T]\!]^{\perp\!\!\!\perp} \quad =_{\text{def}} \quad \{[\ ]u_1 \dots u_n \mid u_1 \in [\![T_1]\!], \dots, u_n \in [\![T_n]\!]\}\ ,$$

on a

$$[\![T]\!] \quad = \quad \{t \mid \forall E[\ ] \in [\![T]\!]^{\perp\!\!\!\perp}.\ \ E[t] \in \mathcal{SN}_\beta\}\ . \tag{9.24}$$

Cette construction est une forme *d'orthogonalité*.

Définition 9.3.33 (Orthogonalité) *Soient deux ensembles $\mathcal{A}$ et Π, et une relation $\perp\!\!\!\perp \subseteq \mathcal{A} \times \Pi$.*

— *Pour tout $A \subseteq \mathcal{A}$,* l'orthogonal de A pour $\perp\!\!\!\perp$ *est*

$$A^{\perp\!\!\!\perp} \quad =_{def} \quad \{\pi \in \Pi \mid \forall a.\ \ a \in A \quad \Longrightarrow \quad a \perp\!\!\!\perp \pi\}\ .$$

— *Symétriquement,* l'orthogonal de $P \subseteq \Pi$ pour $\perp\!\!\!\perp$ *est*

$$P^{\perp\!\!\!\perp} \quad =_{def} \quad \{a \in \mathcal{A} \mid \forall \pi.\ \ \pi \in P \quad \Longrightarrow \quad a \perp\!\!\!\perp \pi\}\ .$$

Les opérateurs d'orthogonalité $(\_)^{\perp\!\!\!\perp} : \mathcal{P}(\mathcal{A}) \to \mathcal{P}(\Pi)$ et $(\_)^{\perp\!\!\!\perp} : \mathcal{P}(\Pi) \to \mathcal{P}(\mathcal{A})$ forment une connexion de Gallois entre les treillis complets $(\mathcal{P}(\mathcal{A}), \subseteq)$ et $(\mathcal{P}(\Pi), \supseteq)$.

Proposition 9.3.34 *Pour tout $A \subseteq \mathcal{A}$ et tout $P \subseteq \Pi$, on a*

$$A \subseteq P^{\perp\!\!\!\perp} \quad \textit{si et seulement si} \quad P \subseteq A^{\perp\!\!\!\perp}\ .$$

Preuve. Il suffit de montrer une direction, l'autre étant symétrique. Supposons que $A \subseteq P^{\perp\!\!\!\perp}$ et montrons que $P \subseteq A^{\perp\!\!\!\perp}$. Par hypothèse, pour tout $a \in A$, on a $a \perp\!\!\!\perp \pi$ pour tout $\pi \in P$, donc pour tout $\pi \in P$, on a $a \perp\!\!\!\perp \pi$ pour tout $a \in A$, d'où $P \subseteq A^{\perp\!\!\!\perp}$. □

Les connections de Gallois induisent des opérateurs de clôture.

Proposition 9.3.35 *Les fonctions $(\_)^{\perp\!\!\!\perp\perp\!\!\!\perp} : \mathcal{P}(\mathcal{A}) \to \mathcal{P}(\mathcal{A})$ et $(\_)^{\perp\!\!\!\perp\perp\!\!\!\perp} : \mathcal{P}(\Pi) \to \mathcal{P}(\Pi)$ sont des opérateurs de clôture.*

Preuve. On ne traite que le cas de $(\_)^{\perp\!\!\!\perp\perp\!\!\!\perp} : \mathcal{P}(\mathcal{A}) \to \mathcal{P}(\mathcal{A})$, l'autre étant symétrique.

Extensivité. Pour tout $X \subseteq \mathcal{A}$ on a $X^{\perp\!\!\!\perp} \subseteq X^{\perp\!\!\!\perp}$ donc $X \subseteq X^{\perp\!\!\!\perp\perp\!\!\!\perp}$ par la proposition 9.3.34.

On déduit de cette propriété que $(\_)^{\perp\!\!\!\perp}$ est anti-monotone :

$$X \subseteq Y \quad \Longrightarrow \quad Y^{\perp\!\!\!\perp} \subseteq X^{\perp\!\!\!\perp}\ . \tag{9.25}$$

En effet, si $X \subseteq Y$ alors par extensivité on a $X \subseteq Y^{\perp\!\!\!\perp\perp\!\!\!\perp}$, donc $Y^{\perp\!\!\!\perp} \subseteq X^{\perp\!\!\!\perp}$ par la proposition 9.3.34.

Monotonie. Si $X \subseteq Y$ alors par (9.25) on a $Y^{\perp\!\!\!\perp} \subseteq X^{\perp\!\!\!\perp}$, d'où $X^{\perp\!\!\!\perp\perp\!\!\!\perp} \subseteq Y^{\perp\!\!\!\perp\perp\!\!\!\perp}$ en ré-appliquant (9.25).

Idempotence. Car

$$X^{\perp\!\!\!\perp} \quad = \quad X^{\perp\!\!\!\perp\perp\!\!\!\perp\perp\!\!\!\perp} \,. \tag{9.26}$$

En effet, on a $X^{\perp\!\!\!\perp} \subseteq X^{\perp\!\!\!\perp\perp\!\!\!\perp\perp\!\!\!\perp}$ par extensivité et $X^{\perp\!\!\!\perp\perp\!\!\!\perp\perp\!\!\!\perp} \subseteq X^{\perp\!\!\!\perp}$ en appliquant la proposition 9.3.34 à $X \subseteq X^{\perp\!\!\!\perp\perp\!\!\!\perp}$, obtenue par deux applications de l'extensivité. □

Définition 9.3.36 *On dit que* $A \subseteq \mathcal{A}$ *(*$P \subseteq \Pi$*) est* un biorthogonal *si* $A = A^{\perp\!\!\!\perp\perp\!\!\!\perp}$.

On dénote par $\mathcal{P}(\mathcal{A})^{\perp\!\!\!\perp\perp\!\!\!\perp}$ *(resp. par* $\mathcal{P}(\Pi)^{\perp\!\!\!\perp\perp\!\!\!\perp}$*) l'ensemble des biorthogonaux de* $\mathcal{P}(\mathcal{A})$ *(resp. de* $\mathcal{P}(\Pi)$*), et par* $\mathcal{P}^\star(\mathcal{A})^{\perp\!\!\!\perp\perp\!\!\!\perp}$ *(resp. par* $\mathcal{P}^\star(\Pi)^{\perp\!\!\!\perp\perp\!\!\!\perp}$*) l'ensemble des biorthogonaux de sous-ensembles non-vides de* $\mathcal{P}(\mathcal{A})$ *(resp. de* $\mathcal{P}(\Pi)$*).*

Proposition 9.3.37 $A \subseteq \mathcal{A}$ *(resp.* $P \subseteq \Pi$*) est un biorthogonal si et seulement s'il existe* $X \subseteq \Pi$ *(resp.* $X \subseteq \mathcal{A}$*) tel que* $A = X^{\perp\!\!\!\perp}$ *(resp.* $P = X^{\perp\!\!\!\perp}$*).*

Preuve. Si A est un biorthogonal alors par définition $A = (A^{\perp\!\!\!\perp})^{\perp\!\!\!\perp}$. Inversement, si $A = Y^{\perp\!\!\!\perp}$ alors $A^{\perp\!\!\!\perp\perp\!\!\!\perp} = Y^{\perp\!\!\!\perp\perp\!\!\!\perp\perp\!\!\!\perp}$, donc $A^{\perp\!\!\!\perp\perp\!\!\!\perp} = Y^{\perp\!\!\!\perp} = A$ par (9.26). □

Voyons maintenant comment appliquer ces idées à la réductibilité. Nous définissons une interprétation des types par biorthogonalité en se basant sur la relation suggéré en (9.24). Par soucis de généralité, nous nous plaçons dans le même cadre que pour les candidats de réductibilité à la section 9.3.4. On considère donc une relation de réécriture $\to_R$ sur $\Lambda(\Sigma)$ et un ensemble $\mathcal{E}$ de contextes d'élimination pour $\to_R$, au sens de la définition 9.3.13.

Définition 9.3.38 *On définit* $\perp\!\!\!\perp \subseteq \Lambda(\Sigma) \times \mathcal{E}$ *par*

$$t \perp\!\!\!\perp E[\,] \qquad \textit{si et seulement si} \qquad E[t] \in \mathcal{SN}_R \,.$$

Les sous-ensembles non-vides de $\mathcal{SN}_R$ qui sont des biorthogonaux pour $\perp\!\!\!\perp$ sont des candidats de réductibilité au sens de la définition 9.3.16.

Lemme 9.3.39 $\mathcal{P}^\star(\mathcal{SN}_R)^{\perp\!\!\!\perp\perp\!\!\!\perp} \subseteq \mathcal{CR}_{R\mathcal{E}}$.

Preuve. Soit $C \in \mathcal{P}^\star(\mathcal{SN}_R)^{\perp\!\!\!\perp\perp\!\!\!\perp}$. On vérifie que C satisfait les clauses ($\mathcal{CR}0$) et ($\mathcal{CR}1$).

($\mathcal{CR}0$) Si $t \in C$ et $t \to_R u$, alors pour tout contexte $E[\,] \in C^{\perp\!\!\!\perp}$ on a $E[t] \to_R E[u]$, donc $E[u] \in \mathcal{SN}_R$.

($\mathcal{CR}1$) Soit $t \in \mathcal{N}_{R\mathcal{E}}$ tel que $(t)_R \subseteq C$. On doit montrer que $E[t] \in \mathcal{SN}_R$ pour tout $E[\,] \in C^{\perp\!\!\!\perp}$. Comme C est un sous-ensemble non vide de $\mathcal{SN}_R$, on a $C^{\perp\!\!\!\perp} \subseteq \mathcal{SN}_R$. On raisonne par induction sur $E[\,] \in \mathcal{SN}_R$. Soit v tel que $E[t] \to_R v$. Comme t est neutre, par définition on a $v = E'[t']$ avec $(E[\,], t) \to_R (E'[\,], t')$. On a alors deux cas :

— $v = E[u]$ avec $t \to_R u$. Dans ce cas on a $E[u]$ car $u \in C$ et $E[\,] \in C^{\perp\!\!\!\perp}$.

— $v = E'[t]$ avec $E[\,] \to_R t'$. On a alors $E'[\,] \in C^{\perp\!\!\!\perp}$ car pour tout $u \in C$, $E[u] \to_R E'[u]$ et $E[u] \in \mathcal{SN}_R$, donc $E'[u] \in \mathcal{SN}_R$. Il s'en suit que $E'[t] \in \mathcal{SN}_R$ par hypothèse d'induction. □

Ainsi, si on se place dans le λ-calcul simplement typé avec produits $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$, comme $\mathcal{E}_{\Rightarrow\times}$ est un ensemble de contextes d'élimination pour $\to_{\beta\pi}$ (remarque 9.3.15), par le lemme 9.3.18, on a $\mathcal{P}^\star(\mathcal{SN}_{\beta\pi})^{\perp\!\!\!\perp\perp\!\!\!\perp} \subseteq \mathcal{SAT}_{\beta\pi}$.

Afin d'avoir une interprétation valide et adéquate, il reste donc à montrer que la flèche préserve les biorthogonaux. Pour cela, on observe comment l'interprétation (9.24) traverse la structure des types simples. Notons que dans le cas de base, on a $[\![T]\!] = \mathcal{SN}_\beta = \{[\ ]\}^{\perp\!\!\!\perp}$. Considérons maintenant un type de la forme $U \Rightarrow T$. L'idée principale est que pour tout $t, u \in \Lambda(\Sigma)$ et tout $E[\ ] \in \mathcal{E}_\Rightarrow$,

$$tu \perp\!\!\!\perp E[\ ] \quad \text{si et seulement si} \quad t \perp\!\!\!\perp E[[\ ]u]\ .$$

On a donc

$$\begin{aligned}
[\![U]\!] \Rightarrow [\![T]\!] &= \{t \mid \forall u \in [\![U]\!],\ \forall E[\ ] \in [\![T]\!]^{\perp\!\!\!\perp}.\quad tu \perp\!\!\!\perp E[\ ]\} \\
&= \{t \mid \forall u \in [\![U]\!],\ \forall E[\ ] \in [\![T]\!]^{\perp\!\!\!\perp}.\quad t \perp\!\!\!\perp E[[\ ]u]\} \\
&= \{E[[\ ]u] \mid u \in [\![U]\!] \quad \wedge \quad E[\ ] \in [\![T]\!]^{\perp\!\!\!\perp}\}^{\perp\!\!\!\perp}\ .
\end{aligned}$$

Définition 9.3.40 *Soit $\mathcal{E}$ un ensemble de contextes d'élimination pour une relation de réécriture $\to_R$ sur $\Lambda(\Sigma)$. Si $[\ ]t \in \mathcal{E}$ pour tout $t \in \Lambda(\Sigma)$, alors étant donnés $A \subseteq \Lambda(\Sigma)$ et $P \subseteq \mathcal{E}$, on pose*

$$A \cdot P \quad =_{def} \quad \{E[[\ ]u] \mid u \in A \quad \wedge \quad E[\ ] \in P\}\ .$$

Proposition 9.3.41 *Soit $\mathcal{E}$ un ensemble de contextes d'élimination pour une relation de réécriture $\to_R$ sur $\Lambda(\Sigma)$.*

(i) Si $[\ ]t \in \mathcal{E}$ pour tout $t \in \Lambda(\Sigma)$, alors pour tout $A, B \subseteq \mathcal{SN}_R$ on a

$$A \Rightarrow B^{\perp\!\!\!\perp\perp\!\!\!\perp} \quad = \quad (A \cdot B^{\perp\!\!\!\perp})^{\perp\!\!\!\perp}\ .$$

(ii) Si $\pi_i[\ ] \in \mathcal{E}$ pour tout $i \in \{1,2\}$, alors pour tout $A_1, A_2 \subseteq \mathcal{SN}_R$ on a

$$A_1^{\perp\!\!\!\perp\perp\!\!\!\perp} \times A_2^{\perp\!\!\!\perp\perp\!\!\!\perp} \quad = \quad \{E[\pi_1[\ ]] \mid E[\ ] \in A_1^{\perp\!\!\!\perp}\}^{\perp\!\!\!\perp} \quad \cap \quad \{E[\pi_2[\ ]] \mid E[\ ] \in A_2^{\perp\!\!\!\perp}\}^{\perp\!\!\!\perp}\ .$$

Preuve.

(i) On a $t \in A \Rightarrow B^{\perp\!\!\!\perp\perp\!\!\!\perp}$ si et seulement si $E[tu] \in \mathcal{SN}_R$ pour tout $u \in A$ et tout $E[\ ] \in B^{\perp\!\!\!\perp}$.

(ii) On a $t \in A_1^{\perp\!\!\!\perp\perp\!\!\!\perp} \times A_2^{\perp\!\!\!\perp\perp\!\!\!\perp}$ si et seulement si $E[\pi_i t] \in \mathcal{SN}_R$ pour tout $i \in \{1,2\}$ et tout $E[\ ] \in A_i^{\perp\!\!\!\perp}$. □

D'après les propositions 9.3.37 et 9.3.35, il s'en suit que $A^{\perp\!\!\!\perp\perp\!\!\!\perp} \Rightarrow B^{\perp\!\!\!\perp\perp\!\!\!\perp}$ et $A^{\perp\!\!\!\perp\perp\!\!\!\perp} \times B^{\perp\!\!\!\perp\perp\!\!\!\perp}$ sont des biorthogonaux pour tout $A, B \subseteq \mathcal{SN}_{\beta\pi}$. De plus, si A et B sont non-vides, alors $A^{\perp\!\!\!\perp\perp\!\!\!\perp}$ et $B^{\perp\!\!\!\perp\perp\!\!\!\perp}$ sont des ensembles saturés, donc $A^{\perp\!\!\!\perp\perp\!\!\!\perp} \Rightarrow B^{\perp\!\!\!\perp\perp\!\!\!\perp}, A^{\perp\!\!\!\perp\perp\!\!\!\perp} \times B^{\perp\!\!\!\perp\perp\!\!\!\perp} \in \mathcal{SAT}_{\beta\pi}$ par la proposition 9.1.25.

On a donc, avec les biorthogonaux, une interprétation valide et adéquate pour le système F avec produits $\boldsymbol{\lambda}_{2\times}(\mathcal{B}_0)$.

Interprétation de la logique classique. Dans le cadre de la réductibilité, les biorthogonaux peuvent être utilisés pour l'interprétation de la logique classique. L'idée est de voir $(\_)^{\perp\!\!\perp}$ comme une sorte de négation (non homogène !) intuitionniste. Par ailleurs, un analogue des lois de De Morgan intuitionnistes pour l'union et l'intersection sont satisfaites (nous reviendrons sur ce point à la section 10.4.3 :

$$\begin{array}{rcl} X^{\perp\!\!\perp} \cap Y^{\perp\!\!\perp} & = & (X \cup Y)^{\perp\!\!\perp} \\ X^{\perp\!\!\perp} \cup Y^{\perp\!\!\perp} & \subseteq & (X \cap Y)^{\perp\!\!\perp} \ . \end{array}$$

De plus, le fait que $(\_)^{\perp\!\!\perp\perp\!\!\perp}$ soit un opérateur de clôture lui donne un comportement similaire à celui des non-non traductions de Gödel.

9.4 Réécriture du premier ordre

Nous avons vu dans ce chapitre ce qui nous semble être les points importants des preuves de normalisation forte de lambda-calculs typés en présence de réécriture.

Nous terminons le chapitre par une courte discussion sur la préservation de la normalisation forte pour la combinaison de la réécriture du premier ordre (resp. algébrique) et du système F. C'est un résultat connu, qui peut s'énoncer de la manière suivante :

Théorème 9.4.1 *Soit $\mathcal{R}$ un TRS algébriquement typé dans $\lambda_2(\mathcal{B}_0, \Sigma, \tau)$. Si $\to_{\mathcal{R}}$ est fortement normalisant sur $\Lambda(\Sigma)$, alors $\to_{\beta\mathcal{R}}$ est fortement normalisant sur les termes typés de $\lambda_2(\mathcal{B}_0, \Sigma, \tau)$.*

Dans le cas du système F *à la Church*, il a été montré dans [BTG89, Oka89]. Notons que les deux résultats ne sont pas exactement les mêmes :

— Okada montre dans [Oka89] que si $\mathcal{R}$ est un TRS sur $\mathcal{T}er(\Sigma, \mathcal{X})$ fortement normalisant et algébriquement typé dans $\lambda_2^{Ch}(\mathcal{B}_0, \Sigma, \tau)$, alors $\to_{\beta\mathcal{R}}$ est fortement normalisant sur les termes typés de $\lambda_2^{Ch}(\mathcal{B}_0, \Sigma, \tau)$: la réécriture n'est pas curryfiée.

— Breazu-Tannen et Gallier montrent dans [BTG89] que si $\mathcal{R}$ est un système algébriquement typé dans $\lambda_2^{Ch}(\mathcal{B}_0, \Sigma_0, \tau)$ tel que $\to_{\mathcal{R}}$ est fortement normalisante, alors $\to_{\beta\mathcal{R}}$ fortement normalisant que les termes typés de $\lambda_2^{Ch}(\mathcal{B}_0, \Sigma_0, \tau)$: la réécriture est curryfiée.

Remarque 9.4.2 *Notons que le résultat de [BTG89] est légèrement plus général que celui de [Oka89] car d'après la remarque 3.6.6, les termes algébriquement typés de $\lambda_{2\Lambda}(\mathcal{B}_0, \lambda_0, \tau)$ forment une Σ_τ-algèbre libre sur $\mathcal{X}$.*

D'autre part, nous allons voir que si $\mathcal{R}$ est un TRS algébriquement typé dans le système $\lambda_2(\mathcal{B}_0, \Sigma_0, \tau)$, alors $\to_{\mathcal{R}}$ normalise fortement sur $\Lambda(\Sigma_0)$ si et seulement si elle normalise fortement sur les termes algébriquement typés de $\lambda_2(\mathcal{B}_0, \Sigma_0, \tau)$.

Le théorème 9.4.1 est adapté aux types intersection dans [BF96], résultat qui est utilisé dans [BFG97] pour obtenir son extension au calcul des constructions. Enfin, une preuve directe dans le calcul des constructions est proposée dans [Bla05b].

Toutes ces preuves reposent sur une méthode similaire : le découpage des termes en « cap » et « aliens », les premier représentant les têtes algébriques des termes et les premier leurs sous-termes non algébriques maximaux.

Des termes du premier ordre aux lambda-termes

Le théorème 9.4.1 utilise le fait que la normalisation forte est préservée lorsque qu'un TRS du premier ordre est curryfié et quotienté par α-conversion. Nous prouvons cette propriété dans cette section en combinant le résultat de [KKSV96] sur la préservation la normalisation forte par curryfication, celui de Middeldorp [Mid89] qui implique la préservation de la normalisation forte par extension de la signature et enfin nos résultats de la section 4.3 sur les liens entre réécriture du premier ordre et réécriture modulo α-conversion.

Tout d'abord, rappelons que la normalisation forte est préservée par curryfication. Pour plus de détails sur la curryfication, voir le chapitre 8.

Théorème 9.4.3 ([KKSV96]) *Si le TRS $(\mathcal{T}er(\Sigma,\mathcal{X}),\mathcal{R})$ est fortement normalisant, alors le TRS $(\mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X}),\mathcal{R}_{\mathcal{C}})$ l'est aussi.*

Le résultat de Middeldorp sur le modularité de la normalisation forte est le suivant :

Théorème 9.4.4 ([Mid89]) *Soient $(\mathcal{T}er(\Sigma_1,\mathcal{X}),\mathcal{R}_1)$ et $(\mathcal{T}er(\Sigma_2,\mathcal{X}),\mathcal{R}_2)$ deux TRS fortement normalisants tels que $\Sigma_1\cap\Sigma_2=\emptyset$. S'il existe $i\in\{1,2\}$ tel que $\mathcal{R}_i$ ne comporte ni règles effondrantes ni règles dupliquantes, alors le TRS $(\mathcal{T}er(\Sigma_1\cup\Sigma_2,\mathcal{X}),\mathcal{R}_1\cup\mathcal{R}_2)$ est fortement normalisant.*

De même qu'à la section 5.3, on a la préservation de la normalisation forte par extension de la signature.

Corollaire 9.4.5 *Si $(\mathcal{T}er(\Sigma,\mathcal{X}),\mathcal{R})$ est un TRS fortement normalisant, alors le TRS $(\mathcal{T}er(\Sigma',\mathcal{X}),\mathcal{R})$ est fortement normalisant pour tout $\Sigma'\supseteq\Sigma$.*

On en déduit que si $\mathcal{R}$ est un système de réécriture sur $\mathcal{T}er(\Sigma,\mathcal{X})$ qui est fortement normalisant, alors $\to_{\mathcal{R}_{\mathcal{C}}}$ est fortement normalisant sur $\Lambda(\Sigma_{\mathcal{C}App})$. Rappelons que $\mathcal{L}(\Sigma)$ désigne les λ-termes bruts sur Σ, c'est-à-dire non quotientés par α-conversion.

Théorème 9.4.6 *Si $(\mathcal{T}er(\Sigma,\mathcal{X}),\mathcal{R})$ est un TRS fortement normalisant, alors la relation de réécriture $\to_{\mathcal{R}_{\mathcal{C}}}$ est fortement normalisante sur $\Lambda(\Sigma_{\mathcal{C}}^{App})$.*

Preuve. Soit $(\mathcal{T}er(\Sigma,\mathcal{X}),\mathcal{R})$ un TRS fortement normalisant. Par le théorème 9.4.3, le TRS $(\mathcal{T}er(\Sigma_{\mathcal{C}App},\mathcal{X}),\mathcal{R}_{\mathcal{C}})$ l'est aussi. Il suit alors du corollaire 9.4.5 que $\to_{\mathcal{R}_{\mathcal{C}}}$ est fortement normalisant sur $\mathcal{L}(\Sigma_{\mathcal{C}})$. Comme $\mathcal{R}_{\mathcal{C}}$ est applicatif, on a la normalisation forte de $\to_{\mathcal{R}_{\mathcal{C}}}$ sur $\Lambda(\Sigma_{\mathcal{C}})$ par le théorème 4.3.8. □

Nous pouvons maintenant prouver ce que nous avons dit à la remarque 9.4.2. Si $\to_{\mathcal{R}}$ est fortement normalisant sur les termes algébriquement typés de $\lambda_2(\mathcal{B}_0,\Sigma_0,\tau)$ alors en prolongeant la remarque 3.6.6, on peut définir un TRS décurryfié $\mathcal{R}_\tau$ tel que $\to_{\mathcal{R}_\tau}$ soit fortement normalisant sur $\mathcal{T}er(\Sigma_\tau,\mathcal{X})$. Comme $\Sigma_{\tau\mathcal{C}}=\Sigma_0$, il suit alors du théorème 9.4.6 que $\to_{\mathcal{R}}$ est fortement normalisant sur $\Lambda(\Sigma_0)$.

Chapitre 10

Unions de familles de réductibilité

Dans ce chapitre, nous étudions la possibilité d'avoir des familles de réductibilités valides, adéquates et stables par union.

Nous commençons par étudier la possibilité d'existence de telles familles de réductibilité. Pour cela, on considère le λ-calcul auquel on ajoute un système de réécriture $\mathcal{R}$ dit « simple » dont les membres gauches sont de la forme $f(x_1, \dots, x_n)$ où $f \in \Sigma_n$ et $x_1, \dots, x_n$ sont des variables distinctes. À la section 10.1, nous présentons un système de types intersection et union désigné par $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$, et nous montrons à la section 10.2 qu'il permet de typer tous les termes fortement $\beta\mathcal{R}$-normalisants et seulement ceux-ci.

Nous verrons à la section 10.3 que le système $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ obtenu en ajoutant à $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ la règle d'élimination des types union permet, pour certains systèmes de réécriture simples, de typer des termes non fortement normalisants. Ainsi, il existe des systèmes de réécriture qui ne peuvent être pris en compte par une famille de réductibilité valide, adéquate et stable par union. Il est intéressant de noter que de tels systèmes peuvent être confluents.

La section 10.4 est consacrée à l'étude de la clôture par union de familles de réductibilité. Nous commençons par aborder la question de manière générale au niveau des opérateurs de clôture. Ensuite, nous rappelons que les ensembles saturés pour le λ-calcul avec produits sont stables par union, puis nous dérivons de notre étude des opérateurs de clôture une condition nécessaire et suffisante pour que les candidats de réductibilité soient stables par union (l'existence de *réduits principaux forts*), et une condition nécessaire et suffisante pour que la clôture par union de biorthogonaux forme un ensemble de candidats de réductibilité (l'existence de *réduits principaux*)[1]. Nous montrons aussi que les réduits principaux forts sont des réduits principaux.

Ensuite, à la section 10.5, nous appliquons ces résultats à la réécriture simple, et nous montrons que l'existence de réduit principaux est une condition strictement plus générale que l'existence de réduit principaux forts. Ainsi, la clôture par union de biorthogonaux aboutit à une condition suffisante pour la normalisation forte dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ strictement plus générale que celle issue de la stabilité par union des candidats de réductibilité.

D'autre part, nous comparons à la section 10.6 les candidats de réductibilité et les ensembles saturés. Nous montrons, dans le cas de $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$, qu'une forme d'ensembles saturés stables par réduction correspond exactement aux candidats de réductibilité. Cela nous donne la stabilité par union des candidats de réductibilité pour ce calcul. Dans

[1] Les « réduits principaux forts » et les « réduits principaux » sont respectivement les « principal reducts » et les « weak principal reducts » de [Rib07a, Rib07b].

le cas de la réécriture simple, nous définissons des ensembles saturés qui permettent de prendre en compte les systèmes de réécriture qui ont des réduits principaux. Cela constitue une ébauche de réponse à la question soulevée à la section 9.3.3 sur l'intégration de la réécriture aux ensembles saturés.

Notons que notre condition pour la stabilité par union des candidats de Girard nous renseigne de manière intéressante sur leur structure « algébrique » : ils sont stables par union exactement lorsque ce sont les ensembles non vides de termes fortement normalisants clos par le bas pour un ordre d'observation faible.

Enfin, nous abordons à la section 10.7 une question complémentaire : existe-t-il des interprétations non stables par union mais qui soient valides et adéquates pour $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}$? Nous montrons que c'est le cas pour une interprétation utilisant une certaine forme de biorthogonaux, basés sur des contextes d'évaluation autorisant « l'appel par nom » et « l'appel par valeur ». Nous montrons que cette interprétation est complète par rapport à la normalisation forte dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$: pour tout système de réécriture simple $\mathcal{R}$, les termes typables dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ sont fortement $\beta\mathcal{R}$-normalisants si et seulement si cette interprétation est valide et adéquate pour $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$.

Nous verrons aussi dans cette section qu'étant donné un système de réécriture simple $\mathcal{R}$, la $\beta\mathcal{R}$-normalisation forte des termes typables dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ est équivalente au fait que la règle $(\sqcup\,\mathrm{E})$ spécifie la propriété opérationnelle suivante : si tous les réduits de tête d'un terme $f(\vec{t})$ peuvent être substitués de manière « sûre » dans un terme v, alors le terme $f(\vec{t})$ peut être lui-même substitué de manière « sûre » dans v. Cette propriété dit que si chaque réduit de tête de $f(\vec{t})$ interagit correctement avec des copies de *lui-même* dans v, alors les copies *des différents* réduits de tête de $f(\vec{t})$ interagissent bien *entre elles* dans v (voir les remarques 10.3.7 et 10.7.4).

Nous avons publié une grande partie du contenu de ce chapitre dans [Rib07a, Rib07b].

10.1 Réécriture simple avec types union et intersection

On commence par présenter un système avec types intersection et union. Ce système comporte en outre une règle provenant de [CS06] qui lui permet, pour des systèmes de réécriture $\mathcal{R}$ dits « simples », de typer tous les termes fortement $\beta\mathcal{R}$-normalisants et seulement ceux-ci.

L'intérêt de ce système est de mettre en évidence des problèmes liées à la stabilité par union des familles de réductibilité. En effet, lorsqu'on lui ajoute la règle d'élimination des types union, le système obtenu permet aussi, pour certains systèmes de réécriture simples $\mathcal{R}$, de typer des termes non $\beta\mathcal{R}$-normalisants. Or, les familles de réductibilités stables par union et prenant en compte la réécriture sont adéquates pour cette règle. Il s'en suit qu'il existe des systèmes de réécriture pour lesquels il n'existe pas de famille de réductibilité stable par union.

Les types intersection pour le λ-calcul ont été introduits dans [CD78]. Voir [Kri90, Gal98, AC98] pour des études plus récentes. Les types union, quand à eux, apparaissent probablement pour la première fois dans [MPS86], et ont été étudiés plus en profondeur dans [BDCL95].

Le système que l'on présente ici utilise pour le typage des symboles une règle introduite dans [CS06]. Cette règle type les symboles $f \in \Sigma_n$ de manière un peu inhabituelle. En effet, elle n'assigne pas de type a priori à f, mais, étant donnés des termes typables $t_1, \dots, t_n$, le terme $f(t_1, \dots, t_n)$ a pour type T si tous les $r[\vec{t}/\vec{x}]$ pour $f(\vec{t}) \mapsto_{\mathcal{R}} r$ sont typables par T. Nous revenons sur ce point à la remarque 10.1.6. Ainsi, le typage des symboles dans le système $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ est assez différent de celui des systèmes de types intersection étudiés dans [BF96, BBF96, BF97].

On considère des λ-termes avec symboles d'arité quelconque :

$$t, u \in \Lambda(\Sigma) \quad ::= \quad x \quad | \quad f(t_1, \dots, t_n) \quad | \quad t\,u \quad | \quad \lambda x.t$$

où $f \in \Sigma_n$ et $x \in \mathcal{X}$.

Définition 10.1.1 *Un système de réécriture $\mathcal{R}$ sur $\Lambda(\Sigma)$ est dit* simple *si toutes ses règles sont de la forme*

$$f(x_1, \dots, x_n) \quad \mapsto_{\mathcal{R}} \quad r$$

où $f \in \Sigma_n$ et $x_1, \dots, x_n$ sont des variables distinctes, et si pour tout $f \in \Sigma$, l'ensemble $\mathcal{R}(f) =_{def} \{r \mid f(\vec{x}) \mapsto_{\mathcal{R}} r\}$ est fini. On pose $\mathcal{F} =_{def} \{f \in \Sigma \mid \mathcal{R}(f) \neq \emptyset\}$ et $\mathcal{C} =_{def} \Sigma \setminus \mathcal{F}$.

Remarque 10.1.2 *Dans une règle $f(\vec{x}) \mapsto_{\mathcal{R}} r$ d'un système de réécriture simple $\mathcal{R}$, il n'y a pas a priori de condition sur les symboles pouvant apparaître dans r.*

Notons toutefois que si $\to_{\mathcal{R}}$ est fortement normalisante alors le symbole f n'apparaît pas dans r.

On considère des types « union » et « intersection » construits à partir du type de base o. Ce sont donc des éléments de $\mathcal{T}_{\Rightarrow}(\{o, \_ \sqcap \_, \_ \sqcup \_\})$.

Définition 10.1.3 *On désigne par $\mathcal{T}_{\sqcap\sqcup}$ l'ensemble des types union et intersection avec type de base o :*

$$T, U \in \mathcal{T}_{\sqcap\sqcup} \quad ::= \quad o \quad | \quad U \Rightarrow T \quad | \quad T \sqcap U \quad | \quad T \sqcup U\ .$$

Les types sont munis de la relation de sous-typage $\sqsubseteq$ décrite à la figure 10.1. Ainsi, $(\mathcal{T}_{\sqcap\sqcup}, \sqsubseteq, \sqcup, \sqcap)$ est un préordre ayant toutes les plus grandes bornes inférieures finies et toutes les plus petites bornes supérieures finies.

Définition 10.1.4 *Étant donné un système de réécriture $\mathcal{R}$ simple sur $\Lambda(\Sigma)$, on désigne par $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ le système dont*

- *les termes sont ceux de $\Lambda(\Sigma)$,*
- *les types sont ceux de $\mathcal{T}_{\sqcap\sqcup}$,*
- *la relation de typage $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ est engendrée par les règles de la figure 10.2,*
- *la relation de réduction est $\mapsto_{\beta\mathcal{R}}$.*

L'opération $\sqcap$ sur les types peut être étendue aux contextes de typage.

Définition 10.1.5 *Étant donnés deux contextes Γ_0 et Γ_1, le contexte $\Gamma_0 \sqcap \Gamma_1$ est défini comme suit :*

$$\Gamma_0 \sqcap \Gamma_1(x) =_{def} \begin{cases} \Gamma_0(x) \sqcap \Gamma_1(x) & \text{si } x \in \Gamma_0 \cap \Gamma_1\ , \\ \Gamma_i(x) & \text{si } x \in \Gamma_i \setminus \Gamma_{1-i}\ . \end{cases}$$

$$\frac{}{T \sqsubseteq T} \qquad \frac{U_2 \sqsubseteq U_1 \quad T_1 \sqsubseteq T_2}{U_1 \Rightarrow T_1 \sqsubseteq U_2 \Rightarrow T_2} \qquad \frac{T \sqsubseteq U \quad U \sqsubseteq V}{T \sqsubseteq V}$$

$$\frac{}{(T \Rightarrow U_1) \sqcap (T \Rightarrow U_2) \sqsubseteq T \Rightarrow (U_1 \sqcap U_2)}$$

$$\frac{T_1, T_2 \sqsubseteq U}{T_1 \sqcup T_2 \sqsubseteq U} \qquad \frac{}{T_1, T_2 \sqsubseteq T_1 \sqcup T_2}$$

$$\frac{}{T_1 \sqcap T_2 \sqsubseteq T_1, T_2} \qquad \frac{T \sqsubseteq U_1, U_2}{T \sqsubseteq U_1 \sqcap U_2}$$

FIG. 10.1: Règles de sous-typage de $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$

$$(\text{Ax})\ \frac{}{\Gamma, x : T \vdash x : T} \qquad (\text{Fun})\ \frac{\Gamma \vdash \vec{t} : \vec{T} \qquad \forall f(\vec{x}) \mapsto_{\mathcal{R}} r, \quad \Gamma, \vec{x} : \vec{T} \vdash r : U}{\Gamma \vdash f(\vec{t}) : U}$$

$$(\Rightarrow \text{I})\ \frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \Rightarrow T} \qquad (\Rightarrow \text{E})\ \frac{\Gamma \vdash t : U \Rightarrow T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T}$$

$$(\sqcap \text{I})\ \frac{\Gamma \vdash t : T_1 \qquad \Gamma \vdash t : T_2}{\Gamma \vdash t : T_1 \sqcap T_2} \qquad (\text{Sub})\ \frac{\Gamma \vdash t : T \qquad T \sqsubseteq U}{\Gamma \vdash t : U}$$

FIG. 10.2: Règles de typage de $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$

Notons que si $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ alors $\Gamma \sqcap \Gamma' \vdash_{\sqcap\sqcup\mathcal{R}} t : T \sqcup T'$ pour tout Γ' et tout T'.

Remarque 10.1.6 *La règle de typage*

$$(\textsc{Fun})\ \frac{\Gamma \vdash \vec{t} : \vec{T} \qquad \forall f(\vec{x}) \mapsto_{\mathcal{R}} r,\quad \Gamma, \vec{x} : \vec{T} \vdash r : U}{\Gamma \vdash f(\vec{t}) : U}$$

est un peu inhabituelle. Pour voir comment elle fonctionne, considérons un symbole $f \in \Sigma$*, défini par les règles* $(f(x_1, \ldots, x_n) \mapsto_{\mathcal{R}} r_j)_{j\in\{1,\ldots,p\}}$ *avec* $p \geq 0$.

Soient des termes $t_1, \ldots, t_n$ *et des types* $T_1, \ldots, T_n$ *tels que* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_i : T_i$ *pour tout* $i \in \{1, \ldots, n\}$*. Supposons de plus que pour tout* $j \in \{1, \ldots, p\}$*, on ait un type* U_j *tel que*

$$\Gamma, x_1 : T_1, \ldots, x_n : T_n \vdash_{\sqcap\sqcup\mathcal{R}} r_j : U_j\ .$$

Alors, en utilisant le sous-typage on obtient

$$\Gamma, x_1 : T_1, \ldots, x_n : T_n \vdash_{\sqcap\sqcup\mathcal{R}} r_j : \bigsqcup_{j\in\{1,\ldots,p\}} U_j\ .$$

Donc avec la règle (Fun)*, il vient*

$$\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(t_1, \ldots, t_n) : \bigsqcup_{j\in\{1,\ldots,p\}} U_j\ .$$

Notons que si $p = 0$*, c'est-à-dire s'il n'y a pas de règle définissant* f*, alors pour tout* $T \in \mathcal{T}_{\sqcap\sqcup}$*, on a* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(t_1, \ldots, t_n) : T$.

La règle (Fun) *est une simplification d'une règle de typage utilisée dans [CS06]. L'intuition de cette règle est que pour typer un terme de la forme* $f(\vec{t})$*, on doit d'abord typer tous les termes de la forme* $r\sigma$ *tels que* $f(\vec{l}) \mapsto_{\mathcal{R}} r$ *et* $\vec{t} = \vec{l}\sigma$*. Du fait qu'on considère des arbres de typage finis, un terme de la forme* $f(\vec{t})$ *ne peut être typé que s'il est fortement* $\mathcal{R}$*-normalisant. En quelque sorte, on internalise dans le système de type les conditions présentées aux lemmes 9.1.35 et 9.3.32.*

On dénote par $\boldsymbol{\lambda}_{\sqcap}$ le système des *types intersection* du λ-calcul pur, dont les termes sont ceux de Λ, dont les types sont les types de $\mathcal{T}_{\sqcap\sqcup}$ qui ne contiennent pas le symbole $\sqcup$, et dont les règles de typage sont celles de $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ auxquelles on enlève (Fun), et où la relation de sous-typage est celle de $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ à laquelle on enlève toutes les règles contenant le symbole $\sqcup$.

10.2 Adéquation et complétude pour la normalisation forte

Dans cette section on montre que pour tout système de réécriture simple $\mathcal{R}$ sur $\Lambda(\Sigma)$, un terme $t \in \Lambda(\Sigma)$ est fortement $\beta\mathcal{R}$-normalisant si et seulement s'il est typable dans $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$.

Pour le sens « si », on montre l'adéquation des candidats de réductibilité pour $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$. Le sens « seulement si » étend un résultat connu pour le λ-calcul pur : un terme $t \in \Lambda$ est fortement β-normalisable si et seulement s'il est typable dans le système $\boldsymbol{\lambda}_{\sqcap}$. Ce résultat est originellement dû à Pottinger [Pot80], sa preuve peut être trouvée dans [Kri90, Gal98], voir aussi [AC98]. Nous l'adaptons au cas de la règle (Fun).

10.2.1 Adéquation

Dans cette section on montre que tous les termes typables dans $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ sont fortement $\beta\mathcal{R}$-normalisants. Pour cela, nous interprétons les types par des candidats de réductibilité, construits à partir des contextes d'élimination de la flèche définis en 9.1.6 :

$$\mathsf{E}[\ ] \in \mathcal{E}_{\Rightarrow} \quad ::= \quad [\ ] \quad | \quad \mathsf{E}[\ ]\,\mathsf{t}\ ,$$

où $\mathsf{t} \in \Lambda(\Sigma)$.

Définition 10.2.1 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. On désigne par $\mathfrak{CR}_{\sqcap\sqcup\mathcal{R}}$ l'ensemble des candidats de réductibilité pour $\to_{\beta\mathcal{R}}$ dans les contextes d'élimination $\mathcal{E}_{\Rightarrow}$, au sens de la définition 9.3.16.*

On désigne par $\mathcal{N}_{\sqcap\sqcup\mathcal{R}}$ (resp. $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}}$) l'ensemble des termes neutres correspondant (resp. héréditairement neutres).

Remarque 10.2.2 *Notons que $\mathcal{N}_{\sqcap\sqcup\mathcal{R}}$ est exactement l'ensemble des termes qui ne sont pas de la forme $\lambda\mathsf{x}.\mathsf{t}$.*

On interprète les types de manière standard. Rappelons que par les lemmes 9.3.24 et 9.3.4, les candidats de réductibilité forment un treillis complet

$$\mathfrak{CR} \quad =_{\text{def}} \quad (\mathfrak{CR}_{\sqcap\sqcup\mathcal{R}}, \subseteq, \bigvee, \bigcap, \mathcal{HN}_{\sqcap\sqcup\mathcal{R}}, \mathcal{SN}_{\beta\mathcal{R}})\ ,$$

où $\bigvee \mathcal{C} = \mathfrak{CR}(\bigcup \mathcal{C})$ pour tout $\mathcal{C} \subseteq \mathfrak{CR}_{\sqcap\sqcup\mathcal{R}}$. On peut donc interpréter $\sqcap$ par $\cap$ et $\sqcup$ par $\vee$. Ainsi $[\![\mathsf{T} \sqcup \mathsf{U}]\!] = \mathfrak{CR}([\![\mathsf{T}]\!] \cup [\![\mathsf{U}]\!])$ est le plus petit candidat de réductibilité contenant $[\![\mathsf{T}]\!] \cup [\![\mathsf{U}]\!]$. On reviendra sur la stabilité par union des candidats de réductibilité à la section 10.4.

Définition 10.2.3 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. On définit l'interprétation $[\![\mathsf{T}]\!] \in \mathfrak{CR}_{\sqcap\sqcup\mathcal{R}}$ des types $\mathsf{T} \in \mathcal{T}_{\sqcap\sqcup}$ par induction sur T comme suit :*

$$\begin{array}{lll}
[\![\mathsf{o}]\!] & =_{def} & \mathcal{SN}_{\beta\mathcal{R}}\ , \\
[\![\mathsf{U} \Rightarrow \mathsf{T}]\!] & =_{def} & [\![\mathsf{U}]\!] \Rightarrow [\![\mathsf{T}]\!]\ , \\
[\![\mathsf{T} \sqcap \mathsf{U}]\!] & =_{def} & [\![\mathsf{T}]\!] \cap [\![\mathsf{U}]\!]\ , \\
[\![\mathsf{T} \sqcup \mathsf{U}]\!] & =_{def} & \mathfrak{CR}([\![\mathsf{T}]\!] \cup [\![\mathsf{U}]\!])\ .
\end{array}$$

Commençons par voir que cette interprétation respecte la relation de sous-typage $\sqsubseteq$.

Proposition 10.2.4 *Soient $\mathsf{X}, \mathsf{Y}, \mathsf{Z} \subseteq \Lambda(\Sigma)$. Si $\mathsf{X} \subseteq \mathsf{Y}$ alors*

(i) $\mathsf{Z} \Rightarrow \mathsf{X} \quad \subseteq \quad \mathsf{Z} \Rightarrow \mathsf{Y}$,

(ii) $\mathsf{Y} \Rightarrow \mathsf{Z} \quad \subseteq \quad \mathsf{X} \Rightarrow \mathsf{Z}$.

Preuve. Rappelons que $\mathsf{A} \Rightarrow \mathsf{B} \ = \ \{\mathsf{t} \mid \forall \mathsf{u}. \quad \mathsf{u} \in \mathsf{A} \Longrightarrow \mathsf{tu} \in \mathsf{B}\}$.

(i) Si $\mathsf{t} \in \mathsf{Z} \Rightarrow \mathsf{X}$ alors $\mathsf{tu} \in \mathsf{X} \subseteq \mathsf{Y}$ pour tout $\mathsf{u} \in \mathsf{Z}$, donc $\mathsf{t} \in \mathsf{Z} \Rightarrow \mathsf{Y}$.

(ii) Si $\mathsf{t} \in \mathsf{Y} \Rightarrow \mathsf{Z}$, alors $\mathsf{tu} \in \mathsf{Z}$ pour tout $\mathsf{u} \in \mathsf{X} \subseteq \mathsf{Y}$, donc $\mathsf{t} \in \mathsf{X} \Rightarrow \mathsf{Z}$. □

Proposition 10.2.5 *Si* $T \sqsubseteq U$ *alors* $[\![T]\!] \subseteq [\![U]\!]$.

PREUVE. Par induction sur $T \sqsubseteq U$.

$$\frac{}{T \sqsubseteq T}$$

Car $[\![T]\!] = [\![T]\!]$.

$$\frac{U_2 \sqsubseteq U_1 \quad T_1 \sqsubseteq T_2}{U_1 \Rightarrow T_1 \sqsubseteq U_2 \Rightarrow T_2}$$

Par hypothèse d'induction on a $[\![U_2]\!] \subseteq [\![U_1]\!]$ et $[\![T_1]\!] \subseteq [\![T_2]\!]$. On conclut par la proposition 10.2.4.

$$\frac{T \sqsubseteq U \quad U \sqsubseteq V}{T \sqsubseteq V}$$

Par hypothèse d'induction on a $[\![T]\!] \subseteq [\![U]\!]$ et $[\![U]\!] \subseteq [\![V]\!]$, d'où $[\![T]\!] \subseteq [\![V]\!]$.

$$\frac{}{(T \Rightarrow U_1) \sqcap (T \Rightarrow U_2) \sqsubseteq T \Rightarrow (U_1 \sqcap U_2)}$$

Soit $t \in ([\![T]\!] \Rightarrow [\![U_1]\!]) \cap ([\![T]\!] \Rightarrow [\![U_2]\!])$ et $u \in [\![T]\!]$. Alors $tu \in [\![U_1]\!]$ et $tu \in [\![U_2]\!]$, d'où $tu \in [\![U_1]\!] \cap [\![U_2]\!]$.

$$\frac{T_1, T_2 \sqsubseteq U}{T_1 \sqcup T_2 \sqsubseteq U}$$

Par hypothèse d'induction on a $[\![T_1]\!], [\![T_2]\!] \subseteq [\![U]\!]$. Il s'en suit que $\mathcal{CR}([\![T_1]\!] \cup [\![T_2]\!]) \subseteq [\![U]\!]$ car $\mathcal{CR}([\![T_1]\!] \cup [\![T_2]\!])$ est la plus petite borne supérieure de $[\![T_1]\!]$ et $[\![T_2]\!]$ dans le treillis $\mathcal{CR}$.

$$\frac{}{T_1, T_2 \sqsubseteq T_1 \sqcup T_2}$$

Car $\mathcal{CR}([\![T_1]\!] \cup [\![T_2]\!])$ est la plus petite borne supérieure de $[\![T_1]\!]$ et $[\![T_2]\!]$ dans le treillis complet $\mathcal{CR}$.

$$\frac{}{T_1 \sqcap T_2 \sqsubseteq T_1, T_2}$$

Car $[\![T_1]\!] \cap [\![T_2]\!]$ est la plus grande borne inférieure de $[\![T_1]\!]$ et $[\![T_2]\!]$ dans le treillis $\mathcal{CR}$.

$$\frac{T \sqsubseteq U_1, U_2}{T \sqsubseteq U_1 \sqcap U_2}$$

Par hypothèse d'induction on a $[\![T]\!] \subseteq [\![U_1]\!], [\![U_2]\!]$. Il s'en suit que $[\![T]\!] \subseteq [\![U_1]\!] \cap [\![U_2]\!]$ car $[\![U_1]\!] \cap [\![U_2]\!]$ est la plus grande borne inférieure de $[\![T_1]\!]$ et $[\![T_2]\!]$ dans le treillis $\mathcal{CR}$. □

On peut maintenant montrer que les termes typables dans $\lambda_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ sont fortement $\beta\mathcal{R}$-normalisants. Rappelons qu'étant donnés une substitution σ et un contexte Γ, $\sigma \models \Gamma$ signifie que $\sigma(x) \in [\![\Gamma(x)]\!]$ pour tout $x \in \mathcal{D}om(\Gamma)$.

Théorème 10.2.6 (Adéquation) *Si* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ *alors* $t \in \mathcal{SN}_{\beta\mathcal{R}}$.

PREUVE. Comme au lemme 9.1.15 : on montre que si $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ et $\sigma \models \Gamma$, alors $t\sigma \in [\![T]\!]$.

On raisonne par induction sur $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$. Le cas de la règle (AX) est trivial. Par le lemme 9.3.18, tout candidat de réductibilité $C \in \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ satisfait ($\mathcal{SAT}1$), et comme les termes de la forme $(\lambda x.u)v$ sont neutres, C satisfait aussi ($\mathcal{SAT}2_\beta$). Il s'en suit que les règles ($\Rightarrow$I) et ($\Rightarrow$E) sont traitées comme au lemme 9.1.15.

(FUN) Soit $\sigma \models \Gamma$. Par hypothèse d'induction, on a $\vec{t}\sigma \in [\![\vec{T}]\!]$. De plus, pour tout $\vec{u} \in [\![\vec{T}]\!]$ et toute règle $f(\vec{x}) \mapsto_{\mathcal{R}} r$, on a $r\sigma[\vec{u}/\vec{x}] \in [\![U]\!]$, soit $r[\vec{u}/\vec{x}] \in [\![U]\!]$ car $FV(r) \subseteq \vec{x}$.
On doit montrer que $f(\vec{t}\sigma) \in [\![U]\!]$. Pour cela, on montre par induction sur $(\vec{u})$ que pour tout $\vec{u} \in [\![\vec{T}]\!]$, on a $f(\vec{u}) \in [\![U]\!]$. Comme $\vec{t}\sigma \in [\![\vec{T}]\!]$, il s'en suivra que $f(\vec{t}\sigma) \in [\![U]\!]$.
Soit donc $\vec{u} \in [\![\vec{T}]\!]$. Comme $f(\vec{u})$ est un terme neutre, par ($\mathcal{CR}1$), il suffit de montrer que $(f(\vec{u}))_{\beta\mathcal{R}} \subseteq [\![U]\!]$. Soit v tel que $f(\vec{u}) \to_{\beta\mathcal{R}} v$. Il y a deux cas :
— $v = f(\vec{u}')$ avec $(\vec{u}) \to_{\beta\mathcal{R}} (\vec{u}')$. Alors $\vec{u}' \in [\![\vec{T}]\!]$ par ($\mathcal{CR}0$), donc $v = f(\vec{u}') \in [\![U]\!]$ par hypothèse d'induction.
— $v = r[\vec{u}/\vec{x}]$ avec $f(\vec{x}) \mapsto_{\mathcal{R}} r$. Alors comme $\vec{u} \in [\![\vec{T}]\!]$, on a $v = r[\vec{u}/\vec{x}] \in [\![U]\!]$.

($\sqcap$I) Soit $\sigma \models \Gamma$. Par hypothèse d'induction, on a $t\sigma \in [\![T_i]\!]$ pour tout $i \in \{1, 2\}$, d'où $t\sigma \in [\![T_1]\!] \cap [\![T_2]\!]$.

(SUB) Par la proposition 10.2.5. □

10.2.2 Complétude

On montre maintenant que tous les termes fortement $\beta\mathcal{R}$-normalisants sont typables dans $\lambda_{\sqcap\sqcup\mathcal{R}}(\Sigma)$. La preuve est standard.

On commence par des propriétés classiques des systèmes de types. La première est l'inversion, qui est très similaire à la propriété d'inversion de [DCLP96, DCLP98], à la différence que dans notre cas, $(\mathcal{T}_{\sqcap\sqcup}, \sqsubseteq, \sqcup, \sqcap)$ n'est pas un treillis distributif.

Proposition 10.2.7 (Inversion)

(i) $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} x : T$ *si et seulement si* $x \in \mathcal{D}om(\Gamma)$ *et* $\Gamma(x) \sqsubseteq T$.

(ii) $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} tu : T$ *si et seulement s'il existe un type* U *tel que* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : U \Rightarrow T$ *et* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$.

(iii) $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \lambda x.t : T$ *si et seulement s'il existe un entier* $n \geq 1$*, et des types* $U_1, \ldots, U_n$ *et* $V_1, \ldots, V_n$ *tels que* $\bigsqcap_{i\in\{1,\ldots,n\}}(U_i \Rightarrow V_i) \sqsubseteq T$ *et* $\Gamma, x : U_i \vdash_{\sqcap\sqcup\mathcal{R}} t : V_i$ *pour tout* $i \in \{1, \ldots, n\}$.

(iv) $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t}) : V$ *si et seulement s'il existe des types* $\vec{T}$ *tels que* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T}$ *et* $\Gamma, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r : V$ *pour tout* $f(\vec{x}) \mapsto_{\mathcal{R}} r$.

PREUVE. Les conditions sont bien évidemment suffisantes. Pour voir qu'elles sont nécessaires, on raisonne par induction sur les dérivations de typage.

(i) La dernière règle utilisée ne peut être que (AX), ($\sqcap$I) ou (SUB). Le cas de (AX) est trivial, celui de (SUB) suit directement de l'hypothèse d'induction.
Quand à ($\sqcap$I), par hypothèse d'induction on a $\Gamma(x) \sqsubseteq T_1, T_2$, d'où $\Gamma(x) \sqsubseteq T_1 \sqcap T_2$.

(ii) La dernière règle utilisée ne peut être que ($\Rightarrow$E), ($\sqcap$I) ou (Sub). Le cas de ($\Rightarrow$E) est trivial, celui de (Sub) suit directement de l'hypothèse d'induction et du fait que $V \sqsubseteq T$ implique $U \Rightarrow V \sqsubseteq U \Rightarrow T$.

Quand à ($\sqcap$I), on a $T = T_1 \sqcap T_2$ et par hypothèse d'induction il existe U_1, U_2 tels que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : U_i \Rightarrow T_i$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U_i$ pour tout $i \in \{1,2\}$. En posant $U =_{\text{def}} U_1 \sqcap U_2$, on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$, et de plus, pour tout $i \in \{1,2\}$, $U_i \Rightarrow T_i \sqsubseteq U \Rightarrow T_i$ donc $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : U \Rightarrow T_i$, soit $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : U \Rightarrow T$ car $(U \Rightarrow T_1) \sqcap (U \Rightarrow T_2) \sqsubseteq U \Rightarrow T$.

(iii) La dernière règle utilisée ne peut être que ($\Rightarrow$I), ($\sqcap$I) ou (Sub). Le cas de ($\Rightarrow$I) est trivial et celui de (Sub) suit directement de l'hypothèse d'induction.

Quand à ($\sqcap$I), on a $T = T_1 \sqcap T_2$ et par hypothèse d'induction, pour tout $i \in \{1,2\}$ il existe $U^i_1, \dots, U^i_{n_i}$ et $V^i_1, \dots, V^i_{n_i}$ tels que $\bigsqcap_{j\in\{1,\dots,n_i\}}(U^i_j \Rightarrow V^i_j) \sqsubseteq T_i$ avec de plus $\Gamma, x : U^i_j \vdash_{\sqcap\sqcup\mathcal{R}} t : V^i_j$ pour tout $j \in \{1, \dots, n_i\}$.

Il s'en suit que

$$\bigsqcap_{i\in\{1,2\}} \bigsqcap_{j\in\{1,\dots,n_i\}} (U^i_j \Rightarrow V^i_j) \quad \sqsubseteq \quad T_1 \sqcap T_2 \ .$$

(iv) La dernière règle utilisée ne peut être que (Fun), ($\sqcap$I) ou (Sub). Le cas de (Fun) est trivial et celui de (Sub) suit directement de l'hypothèse d'induction.

Quand à ($\sqcap$I), on a $T = T_1 \sqcap T_2$ et par hypothèse d'induction, pour tout $i \in \{1,2\}$ il existe $\vec{V}^i$ tels que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{V}^i$ et $\Gamma, \vec{x} : \vec{V}^i \vdash_{\sqcap\sqcup\mathcal{R}} r : T_i$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$.

Il s'en suit que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{V}^1 \sqcap \vec{V}^2$ et que $\Gamma, \vec{x} : \vec{V}^1 \sqcap \vec{V}^2 \vdash_{\sqcap\sqcup\mathcal{R}} r : T_1 \sqcap T_2$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$. □

Le lemme de substitution est une propriété de « tout » système de type. Elle est essentielle pour que la β-réduction corresponde à la réduction des coupures de la flèche.

Lemme 10.2.8 (Substitution) *Si* $\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ *et* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$ *alors on a* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T$.

Preuve. Par induction sur $\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T$.

(Ax) Dans ce cas $t = y \in \mathcal{X}$. Il y a deux possibilités :
— si $t = x$, alors $T = U$, $t[u/x] = u$ et par hypothèse on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$,
— sinon $t[u/x] = t = y$, et comme $\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} y : T$ est obtenu par la règle (Ax), on a $\Gamma(y) = T$ donc $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ par (Ax).

($\Rightarrow$I) Dans ce cas, $t = \lambda y.t_1$ et $T = T_2 \Rightarrow T_1$. Par le lemme 1.2.10, on peut supposer que $y \notin FV(u) \cup \{x\}$. On a donc

$$\frac{\Gamma, y : T_2, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t_1 : T_1}{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} \lambda y.t_1 : T_2 \Rightarrow T_1}$$

et par hypothèse d'induction $\Gamma, y : T_2 \vdash_{\sqcap\sqcup\mathcal{R}} t_1[u/x] : T_1$ donc $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \lambda y.(t_1[u/x]) : T_2 \Rightarrow T_1$. Or, comme $y \notin FV(u) \cup \{x\}$, on a $\lambda y.(t_1[u/x]) = (\lambda y.t_1)[u/x]$ par le lemme 1.3.9.(i).

($\Rightarrow$E) Dans ce cas, $t = t_1 t_2$, et il existe $V \in \mathcal{T}_{\sqcap\sqcup\mathcal{R}}$ tel que

$$\frac{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t_1 : V \Rightarrow T \qquad \Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t_2 : V}{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t_1 t_2 : T}$$

Par hypothèse d'induction, on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_1[u/x] : V \Rightarrow T$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_2[u/x] : V$, d'où $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} (t_1 t_2)[u/x] : T$.

(Fun) Dans ce cas, $t = f(\vec{t})$ et

$$\frac{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T} \qquad \forall f(\vec{x}) \mapsto_{\mathcal{R}} r, \quad \Gamma, x : U, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r : T}{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t}) : T}$$

Par hypothèse d'induction, on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t}[u/x] : \vec{T}$ et pour toute règle $f(\vec{x}) \mapsto_{\mathcal{R}} r$, $\Gamma, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r[u/x] : T$ donc $\Gamma, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r : T$ car $FV(r) \subseteq \vec{x}$. Il s'en suit que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t})[u/x] : T$.

($\sqcap$I) Dans ce cas, $T = T_1 \sqcap T_2$ et

$$\frac{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T_1 \qquad \Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T_2}{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T_1 \sqcap T_2}$$

Par hypothèse d'induction on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T_i$ pour tout $i \in \{1, 2\}$, d'où $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T_1 \sqcap T_2$.

(Sub) Dans ce cas

$$\frac{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : V \qquad V \sqsubseteq T}{\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T}$$

Par hypothèse d'induction on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : V$ d'où $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T$. □

En particulier, si $\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ avec $x \notin FV(t)$, alors on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$. Cette propriété est appelée *contraction*.

On passe maintenant aux propriétés clefs pour la complétude des types intersection et union. Notre preuve suit une démarche similaire à celle de [Gal98].

Proposition 10.2.9 (Interpolation) *Soit Γ et x tels que $x \notin \mathcal{D}om(\Gamma)$. Supposons que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$. Alors il existe V tel que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : V$ et $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} t : T$.*

Preuve. On raisonne par induction sur t.

$t \in \mathcal{X}$. Il y a deux cas :
- si $t = x$ alors $t[u/x] = u$. On prend $V =_{\text{def}} T$ et comme $x \notin \mathcal{D}om(\Gamma)$ on a $\Gamma, x : T \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : T$,
- sinon $t = y \neq x$ et $t[u/x] = y$. On prend $V =_{\text{def}} U$ et comme $x \notin \mathcal{D}om(\Gamma)$ on a $\Gamma, x : U \vdash_{\sqcap\sqcup\mathcal{R}} t : T$.

$t = t_1 t_2$. Par la proposition 10.2.7.(ii), comme $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_1[u/x] t_2[u/x] : T$ il existe W tel que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_1[u/x] : W \Rightarrow T$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_2[u/x] : W$.
Par hypothèse d'induction, il existe V_1, V_2 tels que $\Gamma, x : V_1 \vdash_{\sqcap\sqcup\mathcal{R}} t_1 : W \Rightarrow T$ et $\Gamma, x : V_2 \vdash_{\sqcap\sqcup\mathcal{R}} t_2 : W$, avec $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : V_1 \sqcap V_2$.
En posant $V =_{\text{def}} V_1 \sqcap V_2$, on a $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} t_1 : W \Rightarrow T$ et $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} t_2 : W$, d'où $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} t_1 t_2 : T$.

$t = \lambda y.t_1$. D'après le lemme 1.2.10, on peut supposer $x \neq y$. Par la proposition 10.2.7.(iii) il existe $U_1, \dots, U_n$ et $W_1, \dots, W_n$ tels que $\bigsqcap_{i\in\{1,\dots,n\}}(U_i \Rightarrow W_i) \sqsubseteq T$ ainsi que $\Gamma, y : U_i \vdash_{\sqcap\sqcup\mathcal{R}} t_1[u/x] : W_i$ pour tout $i \in \{1, \dots, n\}$.

Par hypothèse d'induction il existe V_i tel que $\Gamma, x : V_i, y : U_i \vdash_{\sqcap\sqcup\mathcal{R}} t_1 : W_i$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : V_i$ pour tout $i \in \{1, \dots, n\}$. Avec $V =_{\text{def}} \bigsqcap_{i\in\{1,\dots,n\}} V_i$ on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : V$ et $\Gamma, x : V, y : U_i \vdash_{\sqcap\sqcup\mathcal{R}} t_1 : W_i$ pour tout $i \in \{1, \dots, n\}$, d'où $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} \lambda y.t_1 : T$ par la proposition 10.2.7.(iii).

$t = f(t_1, \dots, t_n)$. Par la proposition 10.2.7.(iv) il existe des types $T_1, \dots, T_n$ tels que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t_i[u/x] : T_i$ pour tout $i \in \{1, \dots, n\}$, et $\Gamma, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r[u/x] : T$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$. Or, d'après la proposition 2.4.4, on peut supposer que pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$ on a $x \notin FV(r)$ donc $\Gamma, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r : T$.

Par hypothèse d'induction, pour tout $i \in \{1, \dots, n\}$ il existe un type V_i tel que $\Gamma, x : V_i \vdash_{\sqcap\sqcup\mathcal{R}} t_i : T_i$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : V_i$.

Avec $V =_{\text{def}} \bigsqcap_{i\in\{1,\dots,n\}}$ on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : V$, $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} t_i : T_i$, et comme $x \notin FV(r)$, $\Gamma, x : V, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} r : T$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$. D'où $\Gamma, x : V \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t}) : T$. □

Notons que si $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$ avec $y \notin \mathcal{D}om(\Gamma)$ alors on a $\Gamma[y/x] \vdash_{\sqcap\sqcup\mathcal{R}} t[y/x] : T$.

Lemme 10.2.10 (Expansion faible de tête)

(i) *Si* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$ *et* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x]\vec{v} : T$ *alors* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} (\lambda x.t)u\vec{v} : T$.

(ii) *Si* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T}$ *et* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} r[\vec{t}/\vec{x}]\vec{v} : T$ *pour tout* $f(\vec{x}) \mapsto_{\mathcal{R}} r$, *alors* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t})\vec{v} : T$.

Preuve. On ne détaille que le cas (ii) car le cas (i) est similaire et plus simple.

On traite tout d'abord le cas où $\vec{x} \cap \mathcal{D}om(\Gamma) = \emptyset$. On raisonne par induction sur $|\vec{v}|$.

Cas de base $|\vec{v}| = 0$. Dans ce cas on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T}$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} r[\vec{t}/\vec{x}] : T$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$.

Par la proposition 10.2.9, pour chaque $f(\vec{x}) \mapsto_{\mathcal{R}} r$ il existe $\vec{T}^r$ tels que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T}^r$ et $\Gamma, \vec{x} : \vec{T}^r \vdash_{\sqcap\sqcup\mathcal{R}} r : T$. Comme $\mathcal{R}$ est fini, en prenant $\vec{T}' =_{\text{def}} \bigsqcap_r \vec{T}^r$, on obtient $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T}'$ et $\Gamma, \vec{x} : \vec{T}' \vdash_{\sqcap\sqcup\mathcal{R}} r : T$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$, soit $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t}) : T$.

Cas d'induction. Soient $\vec{v}$ et v tels que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \vec{t} : \vec{T}$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} r[\vec{t}/\vec{x}]\vec{v}v : T$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$. Par la proposition 10.2.7.(ii), pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$ il existe V_r tel que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} r[\vec{t}/\vec{x}]\vec{v} : V_r \Rightarrow T$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} v : V_r$.

Comme $\mathcal{R}(f)$ est fini, en prenant $V =_{\text{def}} \bigsqcap_r V_r$ on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} v : V$ et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} r[\vec{t}/\vec{x}]\vec{v} : V \Rightarrow T$ pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$. Par hypothèse d'induction on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t})\vec{v} : V \Rightarrow T$, d'où $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t})\vec{v}v : T$.

Considérons maintenant le cas où $\vec{x}$ et $\mathcal{D}om(\Gamma)$ ne sont pas disjoints. Posons $\vec{x}' =_{\text{def}} \{x_i \mid i \in \{1, \dots, |\vec{x}|\} \wedge x_i \in \mathcal{D}om(\Gamma)\}$. De ce fait, Γ est de la forme $\Gamma', \vec{x}' : \vec{U}$. Soient $\vec{y}'$ disjoints de $\mathcal{D}om(\Gamma)$ tels que $|\vec{y}'| = |\vec{x}'|$, et posons $\vec{t}' =_{\text{def}} \vec{t}[\vec{y}'/\vec{x}']$ et $\vec{v}' =_{\text{def}} \vec{v}[\vec{y}'/\vec{x}']$. On a alors $\Gamma', \vec{y}' : \vec{U}' \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t}')\vec{v}' : T$. On en déduit que $\Gamma', \vec{x}' : \vec{U}' \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t})\vec{v} : T$ en utilisant le lemme 10.2.8. □

Définition 10.2.11 *On définit* $\succ_{\beta\mathcal{R}}$ *comme étant l'union de* $\rightarrow_{\beta\mathcal{R}}$ *et de la relation sous-terme définie en 1.2.20.*

La relation $\succ_{\beta\mathcal{R}}$ est bien fondée sur $\mathcal{SN}_{\beta\mathcal{R}}$, voir la preuve du lemme 7.1.11.

Théorème 10.2.12 (Complétude) *Pour tout* $t \in \Lambda(\Sigma)$*, si* $t \in \mathcal{SN}_{\beta\mathcal{R}}$ *alors il existe* Γ *et* T *tels que* $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t : T$.

Preuve. On raisonne par induction sur $\prec_{\beta\mathcal{R}}$. Rappelons que par le lemme 3.2.6, tout terme $t \in \Lambda(\Sigma)$ s'écrit sous la forme $\lambda\vec{x}.h\vec{v}$ où h est soit une variable, un β-rédex, ou un terme de la forme $f(\vec{t})$.

Dans le cas $|\vec{x}| \neq 0$, par hypothèse d'induction il existe Γ, $\vec{T}$ et T tels que $\Gamma, \vec{x} : \vec{T} \vdash_{\sqcap\sqcup\mathcal{R}} h\vec{v} : T$, d'où $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \lambda\vec{x}.h\vec{v} : \vec{T} \Rightarrow T$.

On suppose maintenant que $|\vec{x}| = 0$ et on raisonne par cas sur h.

- $h = x \in \mathcal{X}$. Soit $p =_{\text{def}} |\vec{v}|$. Comme $\vec{v} \prec_{\beta\mathcal{R}} t$, par hypothèse d'induction il existe Γ_i, et T_i tels que $\Gamma_i \vdash_{\sqcap\sqcup\mathcal{R}} v_i : T_i$ pour tout $i \in \{1, \dots, p\}$. Il s'en suit que pour tout $T \in \mathcal{T}_{\sqcap\sqcup}$ on a
$$\Gamma_1 \sqcap \cdots \sqcap \Gamma_p \sqcap (x : \vec{T} \Rightarrow T) \vdash_{\sqcap\sqcup\mathcal{R}} x\vec{v} : T \ .$$
- $h = (\lambda x.u)v$. Comme $v, u[v/x]\vec{v} \prec_{\beta\mathcal{R}} t$, par hypothèse d'induction il existe Γ_1, Γ_2, T et V tels que $\Gamma_1 \vdash_{\sqcap\sqcup\mathcal{R}} u[v/x]\vec{v} : T$ et $\Gamma_2 \vdash_{\sqcap\sqcup\mathcal{R}} v : V$. En utilisant le lemme 10.2.10.(i) on en déduit que $\Gamma_1 \sqcap \Gamma_2 \vdash_{\sqcap\sqcup\mathcal{R}} (\lambda x.u)v\vec{v} : T$.
- $h = f(t_1, \dots, t_n)$. Comme $\vec{t} \prec_{\beta\mathcal{R}} f(\vec{t})\vec{v}$, par hypothèse d'induction, il existe un contexte Γ_i et un type T_i tels que $\Gamma_i \vdash_{\sqcap\sqcup\mathcal{R}} t_i : T_i$ pour tout $i \in \{1, \dots, n\}$.
  D'autre part, pour tout $f(\vec{x}) \mapsto_{\mathcal{R}} r$, comme $r[\vec{t}/\vec{x}]\vec{v} \prec_{\beta\mathcal{R}} f(\vec{t})\vec{v}$, par hypothèse d'induction il existe un contexte Γ_r et un type V_r tels que $\Gamma_r \vdash_{\sqcap\sqcup\mathcal{R}} r[\vec{t}/\vec{x}]\vec{v} : V_r$.
  Du fait que $\mathcal{R}(f)$ est fini, en posant
$$\Gamma \quad =_{\text{def}} \quad \Big(\bigsqcap_i \Gamma_i\Big) \sqcap \Big(\bigsqcap_r \Gamma_r\Big) \qquad \text{et} \qquad V \quad =_{\text{def}} \quad \bigsqcup_r V_r \ ,$$
  et en appliquant le lemme 10.2.10.(ii), on obtient $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t})\vec{v} : V$. □

10.3 Élimination de l'union et normalisation forte

Dans cette section, nous montrons que lorsqu'on ajoute à $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ la règle d'élimination de l'union

$$(\sqcup\,\mathrm{E}) \ \frac{\Gamma \vdash t : T_1 \sqcup T_2 \qquad \Gamma, x : T_1 \vdash c : C \qquad \Gamma, x : T_2 \vdash c : C}{\Gamma \vdash c[t/x] : C}$$

alors on obtient un système qui permet, pour certains systèmes de réécriture simples $\mathcal{R}$, de typer des termes qui ne sont pas $\beta\mathcal{R}$-normalisants.

Cette règle peut être vue comme une version implicite de l'élimination des co-produits (voir section 3.7.3). Elle semble avoir été introduite dans [MPS86], et elle est étudiée plus en profondeur dans [BDCL95].

Définition 10.3.1 *On désigne par* $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ *le système dont les termes, les types et la relation de réduction sont ceux de* $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}$*, et dont la relation de typage* $\Gamma \vdash_{(\sqcup\,\mathrm{E})} t : T$ *est engendrée par les règles de la figure 10.2 auxquelles on ajoute la règle* $(\sqcup\,\mathrm{E})$.

Remarque 10.3.2 (Préservation du type par réduction) *Il est connu que la règle* $(\sqcup\,\mathrm{E})$ *fait perdre la préservation du typage par réduction : il se peut que* $\Gamma \vdash_{(\sqcup\,\mathrm{E})\mathcal{R}} t : T$ *avec* $t \to_{\beta\mathcal{R}} u$ *mais que* $\Gamma \vdash_{(\sqcup\,\mathrm{E})\mathcal{R}} u : T$ *ne soit pas dérivable.*

Un exemple pour le λ*-calcul pur est donné dans [BDCL95]. Posons*

$$\Gamma \quad =_{def} \quad x : (U_1 \Rightarrow U_2 \Rightarrow T) \sqcap (U_2 \Rightarrow U_2 \Rightarrow T),\ y : V \Rightarrow (U_1 \sqcup U_2),\ z : V\ .$$

Rappelons que le λ*-terme identité, défini à l'exemple 3.2.5, est le terme* $Id =_{def} \lambda x.x$*. On dérive facilement*

$$\Gamma \vdash y\,z : U_1 \sqcup U_2 \qquad et \qquad \Gamma \vdash Id\,y\,z : U_1 \sqcup U_2\ .$$

D'autre part, pour tout $i \in \{1,2\}$*, on a* $\Gamma, w : U_i \vdash xww : T$*. En appliquant la règle* $(\sqcup\,\mathrm{E})$*, on a donc*

$$(\sqcup\,\mathrm{E})\ \frac{\Gamma \vdash y\,z : U_1 \sqcup U_2 \qquad \Gamma, w : U_1 \vdash xww : T \qquad \Gamma, w : U_2 \vdash xww : T}{\Gamma \vdash x\,(y\,z)(y\,z) : T}$$

ainsi que

$$(\sqcup\,\mathrm{E})\ \frac{\Gamma \vdash Id\,y\,z : U_1 \sqcup U_2 \qquad \Gamma, w : U_1 \vdash xww : T \qquad \Gamma, w : U_2 \vdash xww : T}{\Gamma \vdash x\,(Id\,y\,z)(Id\,y\,z) : T}$$

Cependant, on a

$$x\,(Id\,y\,z)(y\,z) \quad \leftarrow_\beta \quad x\,(Id\,y\,z)(Id\,y\,z) \quad \to_\beta \quad x\,(y\,z)(Id\,y\,z)\ ,$$

alors que les termes $x(yz)(Id\,yz)$ *et* $x(Id\,yz)(yz)$ *ne peuvent avoir le type* T *dans le contexte* Γ*.*

L'intuition de cet exemple est que la règle $(\sqcup\,\mathrm{E})$ *permet d'exprimer une forme de* partage d'information*. Le terme* $x\,(y\,z)(y\,z)$ *(resp.* $x\,(Id\,y\,z)(Id\,y\,z)$*) est typé en éliminant l'union du type de* $y\,z$ *(resp.* $Id\,y\,z$*), et en le faisant* en même temps pour ses deux occurrences *dans* $x\,(y\,z)(y\,z)$ *(resp.* $x\,(Id\,y\,z)(Id\,y\,z)$*), mais ceci n'est pas possible avec les termes* $x(y\,z)(Id\,yz)$ *et* $x(Id\,yz)(yz)$*. Cette notion de partage d'information se retrouve aussi avec l'élimination des types existentiels implicites ; nous reviendrons sur ce point à la section 11.3.2.*

Notons que dans le cas du λ*-calcul pur, il est montré dans [BDCL95] que si* $\Gamma \vdash_{(\sqcup\,\mathrm{E})} t : T$ *avec* $t \to_\beta u$ *alors il existe un terme* v *tel que* $u \to_\beta^* v$ *et* $\Gamma \vdash_{(\sqcup\,\mathrm{E})} v : T$*.*

Nous allons maintenant voir deux exemples de systèmes de réécriture simples $\mathcal{R}$ tels que le typage dans $\lambda_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ n'entraîne pas la $\beta\mathcal{R}$-normalisation forte. On commence par le choix non déterministe « démoniaque », défini par les règles

$$x + y \quad \mapsto_{\mathcal{R}} \quad x \qquad \text{et} \qquad x + y \quad \mapsto_{\mathcal{R}} \quad y\ .$$

Nous verrons à l'exemple 10.3.4 que cela peut aussi se produire pour des systèmes confluents.

Exemple 10.3.3 *Soit* $t_1 =_{def} \lambda z.zy\omega$ *et* $t_2 =_{def} \lambda z.\omega$, *où* ω *est le terme* $\lambda x.xx$ *défini en 3.2.10. Les termes* t_1t_1 *et* t_2t_2 *sont fortement* β*-normalisants mais le terme* t_1t_2 *ne l'est pas*[2]. *En effet,*

$$t_1t_2 \quad \to_\beta \quad (\lambda z.\omega)y\omega \quad \to_\beta \quad \omega\omega \notin \mathcal{SN}_\beta \ .$$

Considérons maintenant le système de réécriture simple constitué des règles

$$x_1 + x_2 \quad \mapsto_{\mathcal{R}} \quad x_1 \qquad et \qquad x_1 + x_2 \quad \mapsto_{\mathcal{R}} \quad x_2 \ .$$

Par le théorème 10.2.12 et la proposition 10.2.9, pour tout $i \in \{1,2\}$ *il existe des types* T_i, U_i *et* V_i *tels que*

$$y : V_i \vdash_{\sqcap\sqcup} t_i : T_i \qquad et \qquad y : V_i, x : T_i \vdash_{\sqcap\sqcup} xx : U_i \ .$$

Ainsi, avec $V =_{def} V_1 \sqcap V_2$, *on a*

$$(\textsc{Fun}) \ \frac{y : V \vdash_{\sqcap\sqcup} t_1 : T_1 \qquad y : V \vdash_{\sqcap\sqcup} t_2 : T_2 \qquad \begin{array}{c} y : V, x_1 : T_1, x_2 : T_2 \vdash_{\sqcap\sqcup} x_1 : T_1 \sqcup T_2 \\ y : V, x_1 : T_1, x_2 : T_2 \vdash_{\sqcap\sqcup} x_2 : T_1 \sqcup T_2 \end{array}}{y : V \vdash_{\sqcap\sqcup\mathcal{R}} t_1 + t_2 : T_1 \sqcup T_2}$$

En appliquant la règle $(\sqcup\,\mathrm{E})$, *on en déduit que*

$$(\sqcup\,\mathrm{E}) \ \frac{y : V \vdash_{\sqcap\sqcup\mathcal{R}} t_1 + t_2 : T_1 \sqcup T_2 \qquad \begin{array}{c} y : V, x : T_1 \vdash_{\sqcap\sqcup\mathcal{R}} xx : U_1 \sqcup U_2 \\ y : V, x : T_2 \vdash_{\sqcap\sqcup\mathcal{R}} xx : U_1 \sqcup U_2 \end{array}}{y : V \vdash_{(\sqcup\,\mathrm{E})\mathcal{R}} (t_1 + t_2)(t_1 + t_2) : U_1 \sqcup U_2}$$

Or

$$(t_1 + t_2)(t_1 + t_2) \quad \to^+_{\beta\mathcal{R}} \quad t_1t_2 \quad \to^+_{\beta\mathcal{R}} \quad \omega\omega \notin \mathcal{SN}_{\beta\mathcal{R}} \ .$$

Il est intéressant de noter qu'il existe aussi des systèmes de réécriture simples *confluents* $\mathcal{R}$ avec lesquels on peut typer dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ des termes non fortement $\beta\mathcal{R}$-normalisants.

Exemple 10.3.4 *Considérons le système de réécriture simple suivant*

$$p \quad \mapsto_{\mathcal{R}} \quad \lambda xy.g(xa\omega) \qquad p \quad \mapsto_{\mathcal{R}} \quad \lambda xy.g(yy) \qquad g(x) \quad \mapsto_{\mathcal{R}} \quad a \ .$$

Notons que le système $\mapsto_{\mathcal{R}}$ *est confluent sur* $\Lambda(\Sigma)$, *donc, comme il est linéaire à gauche,* $\to_{\beta\mathcal{R}}$ *est confluente sur* $\Lambda(\Sigma)$ *par le théorème 6.1.4.*

Posons $u_1 =_{def} \lambda xy.g(xa\omega)$ *et* $u_2 =_{def} \lambda xy.g(yy)$. *Comme* $u_1u_1, u_2u_2 \in \mathcal{SN}_{\beta\mathcal{R}}$, *par le théorème 10.2.12 et la proposition 10.2.9, il existe des types* T_1, T_2 *et* U *tels que*

$$(\sqcup\,\mathrm{E}) \ \frac{\vdash_{\sqcap\sqcup\mathcal{R}} p : T_1 \sqcup T_2 \qquad x : T_1 \vdash_{\sqcap\sqcup\mathcal{R}} xx : U \qquad x : T_2 \vdash_{\sqcap\sqcup\mathcal{R}} xx : U}{\vdash_{(\sqcup\,\mathrm{E})\mathcal{R}} pp : U}$$

alors que

$$pp \quad \to_{\beta\mathcal{R}} \quad u_1u_2 \quad \to_{\beta\mathcal{R}} \quad \lambda y.g((\lambda x'y'.g(y'y'))a\omega) \quad \to^+_{\beta\mathcal{R}} \quad \lambda y.g(g(\omega\omega)) \notin \mathcal{SN}_{\beta\mathcal{R}} \ .$$

[2]Nous devons cet exemple à Philippe de Groote.

Les exemples 10.3.3 et 10.3.4 dépendent fortement de la façon dont la règle (Fun) permet de typer les symboles $f \in \Sigma$. Ils pourraient être évités si on assignait aux symboles un type fixé a priori, comme cela se fait la plupart du temps.

Mais ce qui nous intéresse ici est d'essayer de comprendre quels sont les difficultés à avoir des familles de réductibilité stables par union, et de voir s'il existe des cas où cela est impossible. C'est notre principale motivation dans l'étude des systèmes $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ et $\boldsymbol{\lambda}_{(\sqcup\,E)\mathcal{R}}(\Sigma)$.

Explicitons maintenant les raisons pour lesquelles les exemples 10.3.3 et 10.3.4 empêchent d'avoir, dans certains cas, des familles de réductibilité stables par union.

Définition 10.3.5 *Une famille de réductibilité* $\mathcal{R}\mathrm{ed}$ *au sens de la définition 9.2.4 est* stable par union *si*

$$\emptyset \neq \mathcal{R} \subseteq \mathcal{R}\mathrm{ed} \quad \Longrightarrow \quad \bigcup \mathcal{R} \in \mathcal{R}\mathrm{ed}\ .$$

Lorsqu'on dispose d'une famille de réductibilité stable par union qui permet de prendre en compte un système de réécriture simple $\mathcal{R}$, alors tous les termes typables de $\boldsymbol{\lambda}_{(\sqcup\,E)\mathcal{R}}(\Sigma)$ sont fortement $\beta\mathcal{R}$-normalisants.

Lemme 10.3.6 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. S'il existe une famille de réductibilité $\mathcal{U}$ stable par union ainsi qu'une interprétation $[\![\_]\!]_{\mathcal{U}} : \mathcal{T}_{\sqcap\sqcup} \to \mathcal{U}$ valide et adéquate pour $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ et telle que*

$$[\![T \sqcup U]\!]_{\mathcal{U}} \quad = \quad [\![T]\!]_{\mathcal{U}} \cup [\![U]\!]_{\mathcal{U}}\ ,$$

alors tout terme typable dans $\boldsymbol{\lambda}_{(\sqcup\,E)\mathcal{R}}(\Sigma)$ est fortement $\beta\mathcal{R}$-normalisable.

Preuve. On montre que si $\Gamma \vdash_{(\sqcup\,E)\mathcal{R}} t : T$ et $\sigma \models_{[\![\_]\!]_{\mathcal{U}}} \Gamma$ alors $t\sigma \in [\![T]\!]_{\mathcal{U}}$. On procède par induction sur $\Gamma \vdash_{(\sqcup\,E)\mathcal{R}} t : T$. Les règles de $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ sont prises en compte par hypothèse. Il reste le cas de $(\sqcup\,E)$.

$$(\sqcup\,E)\ \frac{\Gamma \vdash t : T_1 \sqcup T_2 \qquad \Gamma, x : T_1 \vdash c : C \qquad \Gamma, x : T_2 \vdash c : C}{\Gamma \vdash c[t/x] : C}$$

Soit $\sigma \models_{[\![\_]\!]_{\mathcal{U}}} \Gamma$. Par hypothèse d'induction on a $t\sigma \in [\![T_1]\!]_{\mathcal{U}} \cup [\![T_2]\!]_{\mathcal{U}}$ et $c(\sigma[u/x]) \in [\![C]\!]_{\mathcal{U}}$ pour tout $u \in [\![T_1]\!]_{\mathcal{U}} \cup [\![T_2]\!]_{\mathcal{U}}$, d'où $c(\sigma[t\sigma/x]) \in [\![C]\!]_{\mathcal{U}}$.

D'autre part, par le lemme 1.2.10, on peut supposer que $x \notin \mathrm{FV}(\sigma) \cup \mathcal{D}\mathrm{om}(\sigma)$. On a donc $c(\sigma[t\sigma/x]) = (c\sigma)[t\sigma/x]$ par la proposition 1.3.5. De plus, on a $(c[t/x])\sigma = (c\sigma)(\sigma \circ [t/x])$ par le lemme 1.3.9.(ii), soit $(c[t/x])\sigma = (c\sigma)[t\sigma/x]$. Il s'en suit que $(c[t/x])\sigma \in [\![C]\!]_{\mathcal{U}}$. □

En vertu du lemme 10.3.6 et de l'exemple 10.3.4, il existe donc un système de réécriture simple et confluent qui n'admet pas de famille de réductibilité $\mathcal{U}$ stable par union ni d'interprétation $[\![\_]\!]_{\mathcal{U}} : \mathcal{T}_{\sqcap\sqcup} \to \mathcal{U}$ adéquate pour $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ et telle que $[\![T \sqcup U]\!]_{\mathcal{U}} = [\![T]\!]_{\mathcal{U}} \cup [\![U]\!]_{\mathcal{U}}$.

Notons que dans le lemme 10.3.6, nous avons supposé que la famille de réductibilité $\mathcal{U}$ donne lieu à une interprétation valide et adéquate pour $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$. Cette condition nous semble être naturellement satisfaite par n'importe quelle famille de réductibilité permettant de prendre en compte la réécriture et la β-réduction. En effet, si $\mathcal{U}$ est une famille de réductibilité stable par union et satisfaisant, pour tout $C \in \mathcal{U}$,

($\mathcal{SAT}0$) si $t \in C$ et $t \to_{\beta\mathcal{R}} u$ alors $u \in C$,
($\mathcal{SAT}1$) si $[\,]\vec{t} \in \mathcal{SN}_{\beta\mathcal{R}}$ et $x \in \mathcal{X}$ alors $x\vec{t} \in C$,
($\mathcal{SAT}2_\beta$) si $t[u/x]\vec{t} \in C$ et $u \in \mathcal{SN}_{\beta\mathcal{R}}$ alors $(\lambda x.t)u\vec{t} \in C$,
($\mathcal{SAT}2_\mathcal{R}$) si $\forall v.\, f(t_1,\dots,t_n)\vec{t} \to_{\beta\mathcal{R}} v \implies v \in C$, alors $f(t_1,\dots,t_n)\vec{t} \in C$,
alors toute interprétation $[\![\_]\!] : \mathcal{T}_{\sqcap\sqcup} \to \mathcal{U}$ telle que

$$\begin{array}{lcl} [\![U \Rightarrow T]\!]_u & = & [\![U]\!]_u \Rightarrow [\![T]\!]_u\ , \\ [\![T \sqcap U]\!]_u & = & [\![T]\!]_u \cap [\![U]\!]_u\ , \\ [\![T \sqcup U]\!]_u & = & [\![T]\!]_u \cup [\![U]\!]_u\ , \end{array}$$

est adéquate pour $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$.

Remarque 10.3.7 *La perte de normalisation forte dans les exemples 10.3.3 et 10.3.4 peut être analysée de la manière suivante.*

Avec certains systèmes de réécriture, en particulier celui du choix non déterministe, on peut avoir des termes de type $T_1 \sqcup T_2$ qui ne sont ni de type T_1 ni de type T_2. C'est le cas pour le terme $t_1 + t_2$ de l'exemple 10.3.3, où t_1 est de type T_1 mais pas de type T_2, et inversement pour t_2. Dans ce cas, le type « union » $T_1 \sqcup T_2$ ne peut pas être vu comme une union des types T_1 et T_2.

Pourtant, la règle

$$(\sqcup\,\mathrm{E})\ \frac{\Gamma \vdash t : T_1 \sqcup T_2 \qquad \Gamma, x : T_1 \vdash c : C \qquad \Gamma, x : T_2 \vdash c : C}{\Gamma \vdash c[t/x] : C}$$

« force » le type $T_1 \sqcup T_2$ à se comporter comme une union : la substitution de t pour x dans c n'est justifiée que par le fait que dans c, la variable x doit être de type T_1 ou T_2. Il n'est donc pas surprenant que l'on ait un problème lorsqu'on substitue à x un terme $t_1 + t_2$ qui n'est ni de type T_1, ni de type T_2.

Dans le cas de $t_1 + t_2$, on peut pousser l'interprétation un peu plus loin. Supposons que l'on ait

$$\frac{\dfrac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash t_1 + t_2 : T_1 \sqcup T_2} \qquad \Gamma, x : T_1 \vdash c : C \qquad \Gamma, x : T_2 \vdash c : C}{\Gamma \vdash c[t_1 + t_2/x] : C}$$

Si on a $\Gamma, x : T_1 \vdash_{\sqcap\sqcup+} c : C$ et $\Gamma, x : T_2 \vdash_{\sqcap\sqcup+} c : C$, alors par le lemme de substitution 10.2.8, on a $\Gamma \vdash_{\sqcap\sqcup+} c[t_1/x] : C$ et $\Gamma \vdash_{\sqcap\sqcup+} c[t_2/x] : C$. Ainsi, par le théorème 10.2.6, on a $c[t_1/x] \in \mathcal{SN}_{\beta+}$ et $c[t_2/x] \in \mathcal{SN}_{\beta+}$.

Cela veut dire que les termes t_1 et t_2 peuvent être dupliqués de manière « sûre » dans le « contexte[3] » c, c'est-à-dire que les différentes occurrences de t_1 (resp. t_2) dans $c[t_1/x]$ (resp. $c[t_2/x]$) interagissent bien entre elles. *D'un autre côté, la substitution $c[t_1 + t_2/x]$ a pour effet de placer dans c* en même temps *des occurrences de t_1 et des occurrences de t_2, alors que rien n'assure qu'elles interagissent bien entre elles.*

Notons que si on évalue $t_1 + t_2$ avant de substituer dans c, alors on substitue dans c un terme t_i de type T_i, ce qui ne pose pas de problèmes. Ainsi, la règle $(\sqcup\,\mathrm{E})$ *se comporte bien en « appel par valeur », caractéristique exploitée par exemple dans [VM04].*

[3] Rappelons que c n'est pas un contexte au sens de la définition 1.2.18 car il ne capture pas de variables.

10.4 Stabilité par union de familles de réductibilité

Dans cette section, nous étudions des conditions suffisantes pour avoir une famille de réductibilité stable par union prenant en compte une relation de réécriture donnée. Rappelons que nous avons vu à la section 10.3 que ce n'est pas toujours possible, c'est-à-dire qu'il existe des relations de réécriture qui n'admettent pas de familles de réductibilité stables par union.

Pour essayer d'obtenir une famille de réductibilité stable par union, nous étudions sous quelles conditions la clôture par union d'une famille de réductibilité est encore une famille de réductibilité. Comme les familles de réductibilité que nous avons présentées à la section 9.3 peuvent être obtenues à l'aide d'opérateurs de clôture, nous commençons par voir, pour un opérateur de clôture $\overline{(\_)}$ sur $\mathcal{P}(D)$, quelle est la clôture par union de l'ensemble des éléments clos de $\mathcal{P}(D)$. On écrit $\overline{d}$ pour désigner $\overline{\{d\}}$ et $\overline{\mathcal{P}^\star(D)}$ pour désigner $\{\overline{X} \mid \emptyset \neq X \subseteq D\}$.

Définition 10.4.1 *Étant donné un opérateur de clôture $\overline{(\_)} : \mathcal{P}(D) \to \mathcal{P}(D)$, on désigne par $\overline{\Omega^\star}$ l'ensemble des $X \subseteq D$ non vides tels que*

$$X \quad = \quad \bigcup\{\overline{d} \mid d \in X\} \,.$$

Proposition 10.4.2 *Soit un opérateur de clôture $\overline{(\_)} : \mathcal{P}(D) \to \mathcal{P}(D)$. Alors $\overline{\Omega^\star}$ est le plus petit ensemble tel que*

$$\overline{\mathcal{P}^\star(D)} \subseteq \overline{\Omega^\star} \qquad \textit{et} \qquad \left(\emptyset \neq \mathcal{C} \subseteq \overline{\Omega^\star} \quad \Longrightarrow \quad \bigcup\mathcal{C}, \bigcap\mathcal{C} \in \overline{\Omega^\star}\right) \,.$$

Preuve. Pour tout $X \in \overline{\mathcal{P}^\star(X)}$, on a $X = \bigcup\{\overline{d} \mid d \in X\}$ car $\overline{d} \subseteq X$ et $d \in \overline{d}$ pour tout $d \in X$. D'où $\overline{\mathcal{P}^\star(D)} \subseteq \overline{\Omega^\star}$.

Si $\mathcal{C} \subseteq \overline{\Omega^\star}$, alors $C = \bigcup\{\overline{d} \mid d \in C\}$ pour tout $C \in \mathcal{C}$. Il s'en suit que

$$\bigcup\mathcal{C} \quad = \quad \bigcup\{\overline{d} \mid \exists C. \quad C \in \mathcal{C} \quad \wedge \quad d \in C\} \quad = \quad \bigcup\{\overline{d} \mid d \in \bigcup\mathcal{C}\} \,.$$

D'autre part, si $d \in \bigcap\mathcal{C}$, alors pour tout $C \in \mathcal{C}$, on a $d \in C$, donc $\overline{d} \subseteq C$, et inversement, si $\overline{d} \subseteq C$ pour tout $C \in \mathcal{C}$, alors $d \in \bigcap\mathcal{C}$. Ainsi,

$$\bigcap\mathcal{C} \quad = \quad \bigcup\{\overline{d} \mid \forall C. \quad C \in \mathcal{C} \quad \Longrightarrow \quad d \in C\} \quad = \quad \bigcup\{\overline{d} \mid d \in \bigcap\mathcal{C}\} \,.$$

Enfin, soit Ω un ensemble contenant $\overline{\mathcal{P}^\star(D)}$, stable par intersection et par union. Si $X \in \overline{\Omega^\star}$, alors $X = \bigcup\{\overline{d} \mid d \in X\}$. Or $\{\overline{d} \mid d \in X\} \subseteq \Omega$, donc $\bigcup\{\overline{d} \mid d \in X\} \in \Omega$. □

10.4.1 Ensembles saturés

On commence par appliquer la proposition 10.4.2 aux ensembles saturés. On considère les ensembles $\beta\pi$-saturés pour $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$, tels que présentés à la définition 9.1.20.

Comme on l'a vu à la section 9.3.2, les ensembles $\beta\pi$-saturés peuvent être définis via un opérateur de clôture. Étant donné $S \subseteq \mathcal{SN}_{\beta\pi}$, on désigne par $\mathcal{SAT}(S)$ le plus petit

ensemble $\beta\pi$-saturé contenant S. On étend la notation aux termes $t \in \mathcal{SN}_{\beta\pi}$ en posant $\mathbb{SAT}(t) =_{\text{def}} \mathbb{SAT}(\{t\})$. Ainsi, tout $S \in \mathbb{SAT}_{\beta\pi}$ s'écrit sous la forme

$$S \quad = \quad \bigcup\{\mathbb{SAT}(t) \mid t \in S\} \,. \tag{10.1}$$

Il suit de la proposition 10.4.2 que $\mathbb{SAT}_{\beta\pi}$ est stable par union si et seulement si tout ensemble $S \subseteq \mathcal{SN}_{\beta\pi}$ vérifiant (10.1) est $\beta\pi$-saturé. Cette condition est aisément vérifiée.

Théorème 10.4.3 *L'ensemble $\mathbb{SAT}_{\beta\pi}$ définit en 9.1.20 est stable par union.*

Preuve. Soit $S \subseteq \mathcal{SN}_{\beta\pi}$ vérifiant (10.1).

La clause ($\mathbb{SAT}1$) est évidente : si $E[\,] \in \mathcal{E}_{\Rightarrow\times} \cap \mathcal{SN}_{\beta\pi}$ et $x \in \mathcal{X}$ alors $E[x] \in \mathbb{SAT}(t)$ pour tout $t \in S$ donc $E[x] \in S$.

Les clauses ($\mathbb{SAT}2_\beta$) et ($\mathbb{SAT}2_{\times i}$) peuvent être traitées ensemble : on a $t \mapsto_{\beta\pi} u$ avec $t \in \mathcal{SN}_{\beta\pi}$ et $E[u] \in S$. Il s'en suit que $E[u] \in \mathbb{SAT}(v)$ pour un $v \in S$, et on en déduit que $E[t] \in \mathbb{SAT}(v) \subseteq S$. □

Remarque 10.4.4 *La stabilité par union des ensembles saturés est un fait connu, voir par exemple [Wer94], voir aussi [Abe06] pour un exemple d'utilisation de cette propriété.*

On passe au cas du λ-calcul combiné à un système de réécriture simple $\mathcal{R}$ sur $\Lambda(\Sigma)$. D'après ce que l'on a vu à la section 9.3.3, on doit considérer des ensembles saturés qui satisfont ces deux clauses supplémentaires :

($\mathbb{SAT}0$) si $t \in S$ et $t \to_{\beta\mathcal{R}} u$ alors $u \in S$,

($\mathbb{SAT}2_\mathcal{R}$) si $\forall v.\ E[f(t_1,\dots,t_n)] \to_{\beta\mathcal{R}} v \implies v \in S$ alors $E[f(t_1,\dots,t_n)] \in S$.

De même qu'à la section 9.3, ces deux clauses sont plus uniformément prises en compte par les candidats de réductibilité.

10.4.2 Candidats de réductibilité

Considérons maintenant le cas des candidats de réductibilité. Soient $\to_R$ une relation de réécriture sur $\Lambda(\Sigma)$ et $\mathcal{E}$ un ensemble de contextes d'élimination pour $\to_R$.

Pour tout $t \in \mathcal{SN}_R$, on désigne $\mathbb{CR}(\{t\})$ par $\mathbb{CR}(t)$. Il suit du lemme 9.3.23 que tout $C \in \mathbb{CR}_{R\mathcal{E}}$ s'écrit sous la forme

$$C \quad = \quad \bigcup\{\mathbb{CR}(t) \mid t \in C\} \,. \tag{10.2}$$

Comme on l'a vu à la proposition 10.4.2, $\mathbb{CR}_{R\mathcal{E}}$ est stable par union si et seulement si tout ensemble $C \subseteq \mathcal{SN}_R$ vérifiant (10.2) est un candidat de réductibilité.

Voyons ce qui peut assurer $C \in \mathbb{CR}_{R\mathcal{E}}$ pour un ensemble $C \subseteq \mathcal{SN}_R$ satisfaisant (10.2).

— La clause ($\mathbb{CR}0$) est aisément vérifiée : si $t \in C$ et $t \to_R u$ alors $u \in \mathbb{CR}(t)$ par ($\mathbb{CR}0$), donc $u \in C$ par (10.2).

— Le cas de ($\mathbb{CR}1$) est plus délicat. Soit $t \in \mathcal{N}_{R\mathcal{E}}$ tel que $\forall u.\ t \to_R u \implies u \in C$. Comme C vérifie (10.2), pour tout $u \in (t)_R$ il existe $t_u \in C$ tel que $u \in \mathbb{CR}(t_u)$. Mais on ne peut en conclure $t \in C$ que s'il existe $t_u \in C$ tel que $t \in \mathbb{CR}(t_u)$. Or, par ($\mathbb{CR}0$) appliqué à $\mathbb{CR}(t_u)$, cela est équivalent au fait que $(t)_R \subseteq \mathbb{CR}(t_u)$.

Ainsi, pour tout $C \subseteq \mathcal{SN}_R$ vérifiant (10.2) on a $C \in \mathcal{CR}_{R\mathcal{E}}$ si et seulement si

$$\forall t \in \mathcal{N}_{R\mathcal{E}}.\quad (t)_R \subseteq C \quad\Longrightarrow\quad \exists u \in C.\quad (t)_R \subseteq \mathcal{CR}(u)\ . \tag{10.3}$$

Cette propriété n'est pas satisfaite pour tout système de réécriture.

Exemple 10.4.5 *D'après le lemme 10.3.6, les systèmes de réécriture simples $\mathcal{R}$ présentés aux exemples 10.3.3 et 10.3.4 engendrent des relations de réécriture $\to_{\beta\mathcal{R}}$ pour lesquelles $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ n'est pas stable par union, donc pour lesquels la propriété (10.3) n'est en général pas vérifiée pour les ensembles C satisfaisant (10.2).*

Il est donc intéressant de voir quelles conditions sur $\to_R$ permettent d'assurer (10.3). Afin de comprendre cette propriété, essayons d'analyser sous quelles conditions, étant donnés $t \in \mathcal{SN}_R$ et $C \subseteq \mathcal{SN}_R$, on a $t \in \mathcal{CR}(C)$.

Proposition 10.4.6 *Soit $C \subseteq \mathcal{SN}_R$ et $t \in \mathcal{SN}_R$. On a $t \in \mathcal{CR}(C)$ si et seulement si $t \in (C)^*_R$ ou ($t \in \mathcal{N}_{R\mathcal{E}}$ et $(t)_R \subseteq \mathcal{CR}(C)$).*

Preuve. Le sens « si » suit directement de la définition de $\mathcal{CR}(\_)$ (définition 9.3.22).

Pour le sens « seulement si », soit $\mathfrak{a} \in \mathfrak{O}$ le plus petit ordinal tel que $t \in \mathcal{CR}_\mathfrak{a}(C)$. Alors ou bien $\mathfrak{a} = 0$ et $t \in (C)^*_R$, ou bien $\mathfrak{a} = \mathfrak{b} + 1$, et $t \in \mathcal{N}_{R\mathcal{E}}$ avec $(t)_R \subseteq \mathcal{CR}_\mathfrak{b}(C)$. □

Par rapport à (10.3), c'est le cas où $t \in \mathcal{N}_{R\mathcal{E}}$ qui nous intéresse dans la proposition 10.4.6.

Proposition 10.4.7 *Soit $C \subseteq \mathcal{SN}_R$ et $t \in \mathcal{N}_{R\mathcal{E}} \cap \mathcal{SN}_R$. On a $(t)_R \subseteq \mathcal{CR}(C)$ si et seulement si*

$$\forall v \notin \mathcal{N}_{R\mathcal{E}}.\quad t \to^*_R v \quad\Longrightarrow\quad v \in (C)^*_R\ .$$

Preuve. On raisonne par induction sur $t \in \mathcal{SN}_R$.

Supposons que $(t)_R \subseteq \mathcal{CR}(C)$ et soit $v \in (t)^*_R \backslash \mathcal{N}_{R\mathcal{E}}$. Comme $t \in \mathcal{N}_{R\mathcal{E}}$, il existe $u \in \mathcal{CR}(C)$ tel que $t \to_R u \to^*_R v$. Si $u \notin \mathcal{N}_{R\mathcal{E}}$, alors par la proposition 10.4.6, on a $u \in (C)^*_R$ donc $v \in (C)^*_R$. Sinon, comme $u \in \mathcal{CR}(C)$, on a $(u)_R \subseteq \mathcal{CR}(C)$ par ($\mathcal{CR}0$) d'où $v \in (C)^*_R$ par hypothèse d'induction.

Inversement, soit $t \in \mathcal{N}_{R\mathcal{E}}$ tel que $v \in (C)^*_R$ pour tout $v \in (t)^*_R \setminus \mathcal{N}_{R\mathcal{E}}$. Si $t \in (C)^*_R$ alors $(t)_R \subseteq (C)^*_R$. Sinon, pour tout $u \in (t)_R$, par hypothèse on a $u \in \mathcal{CR}(C)$ si $u \notin \mathcal{N}_{R\mathcal{E}}$, et si $u \in \mathcal{N}_{R\mathcal{E}}$, par hypothèse d'induction on a $(u)_R \subseteq \mathcal{CR}(C)$ car $(u)^*_R \setminus \mathcal{N}_{R\mathcal{E}} \subseteq (C)^*_R$, d'où $u \in \mathcal{CR}(C)$ par ($\mathcal{CR}1$). □

Ainsi, un terme fortement normalisant est dans un candidat de réductibilité si et seulement si tous ses réduits non neutres le sont. L'idée principale est qu'un terme non neutre est un terme qui interagit avec les contextes d'élimination. C'est donc un élément « observable », et il peut être vu comme une « valeur ». De ce fait, un terme fortement normalisant t est dans un candidat de réductibilité C si et seulement si toutes ses « valeurs » sont dans C.

D'autre part, étant donné $u \in \mathcal{SN}_\mathcal{R}$, d'après les propositions 10.4.6 et 10.4.7 toutes les valeurs de $\mathcal{CR}(u)$ sont des valeurs de u. Il s'en suit que $t \in \mathcal{CR}(u)$ si et seulement si toutes les valeurs de t sont des valeurs de u.

Cette idée peut être formulée à l'aide d'un préordre d'observation faible sur les termes.

Définition 10.4.8 *On pose* $t \lesssim_{\mathcal{N}} u$ *si et seulement si*

$$\forall v \notin \mathcal{N}_{R\mathcal{E}}.\quad t \to_R^* v \quad\Longrightarrow\quad u \to_R^* v\ .$$

Il est clair que $t \to_R u$ implique $u \lesssim_{\mathcal{N}} t$. De plus, la relation $\lesssim_{\mathcal{N}}$ est stable par contextes d'élimination.

Proposition 10.4.9 *Si* $t \lesssim_{\mathcal{N}} u$ *alors pour tout* $E[\ ] \in \mathcal{E}$ *on a* $E[t] \lesssim_{\mathcal{N}} E[u]$.

Preuve. On montre que pour tout $n \in \mathbb{N}$, tout $v \notin \mathcal{N}_{R\mathcal{E}}$, tous t, u tels que $t \lesssim_{\mathcal{N}} u$ et tout $E[\ ] \in \mathcal{E}$, si $E[t] \to_R^n v$ alors $E[u] \to_R^* v$. On raisonne par induction sur n.

Si t n'est pas neutre, alors $u \to_R^* t$ car $t \lesssim_{\mathcal{N}} u$ donc $E[u] \to_R^* E[t]$.

Si t est neutrel, alors on a $n \geq 1$. Supposons donc que $E[t] \to_R w \to_R^n v$. Alors comme t est neutre, on a $w = E'[t']$ avec $(E[\ ], t) \to_R (E[\ ], t')$.

— Si $E[\ ] \to_R E'[\ ]$ avec $t = t'$, alors comme $\mathcal{E}$ est stable par R-réduction, par hypothèse d'induction on a $E'[u] \to_R^* v$, donc $E[u] \to_R^* v$.

— Sinon, on a $t \to_R t'$ et $E'[\ ] = E[\ ]$. Si t' n'est pas neutre alors $u \to_R^* t'$ donc $E[u] \to_R^* v$, et dans le cas contraire, on a $t' \lesssim_{\mathcal{N}} u$, d'où $E[u] \to_R^* v$ par hypothèse d'induction. □

Ainsi, la proposition 10.4.9 implique que si $[\ ] \in \mathcal{E}$, alors pour tout $t, u \in \Lambda(\Sigma)$,

$$t \lesssim_{\mathcal{N}} u \qquad \text{si et seulement si} \qquad \forall E[\ ] \in \mathcal{E}.\quad E[t] \lesssim_{\mathcal{N}} E[u]\ .$$

Remarque 10.4.10 *Historiquement, les préordres d'observation ont été introduits pour caractériser l'équivalence observationnelle entre programmes : deux parties de programme sont observationnellement équivalentes si, lorsqu'elles sont placées dans le même contexte, les deux programmes tous les deux divergent ou donnent la même valeur.*

Les contextes utilisés sont en général des termes $C[\ ]$ avec un trou, éventuellement sous un lieur. Lorsque les parties de programmes à comparer sont des termes clos, le Lemme du Contexte de Milner [Mil77] dit que l'observation dans les contextes se ramène à l'observation dans les contextes applicatifs (ce sont les contextes d'élimination de la flèche). Cette propriété n'est bien évidement pas satisfaite pour les termes ouverts. Nous renvoyons le lecteur à [Mil77, JM96, HO00] pour une présentation et des références sur le sujet, voir aussi [AC98].

Dans le cas du λ-calcul pur, les termes non neutres sont les abstractions, et de ce fait les termes non neutres clos et en forme normale correspondent aux valeurs des langages fonctionnels. D'autre part, on a $t \lesssim_{\mathcal{N}} u$ si et seulement si $E[t] \lesssim_{\mathcal{N}} E[u]$ pour tout $E[\ ] \in \mathcal{E}_{\Rightarrow}$. De ce fait, avec $\lesssim_{\mathcal{N}}$ on observe la réduction vers des « valeurs » de termes ouverts placés dans des contextes d'évaluations, d'où la dénomination « ordre d'observation faible *».*

Pour tout $t \in \mathcal{SN}_R$, le plus petit candidat de réductibilité contenant t est un segment initial de $\mathcal{SN}_R$ pour $\lesssim_{\mathcal{N}}$.

Lemme 10.4.11 *Pour tout* $t \in \mathcal{SN}_R$, *on a* $\mathcal{CR}(t) = \{u \in \mathcal{SN}_R \mid u \lesssim_{\mathcal{N}} t\}$.

PREUVE. D'après la proposition 10.4.6, pour tout $t, u \in \mathcal{SN}_R$ on a $u \in \mathcal{CR}(t)$ si et seulement si $t \to_R^* u$ ou bien $u \in \mathcal{N}_{R\mathcal{E}}$ et $\forall v \notin \mathcal{N}_{R\mathcal{E}}.\ u \to_R^* v \implies t \to_R^* v$, c'est-à-dire si et seulement si $u \lesssim_{\mathcal{N}} t$. □

Voyons maintenant ce que la caractérisation du lemme 10.4.11 nous dit sur (10.2) et (10.3). Étant donné un ensemble $C \subseteq \mathcal{SN}_R$ vérifiant (10.2), on a vu que $C \in \mathcal{CR}_{R\mathcal{E}}$ si et seulement si C satisfait (10.3). D'après le lemme 10.4.11, cela est équivalent au fait que

$$\forall t \in \mathcal{N}_{R\mathcal{E}}.\quad (t)_R \subseteq C \quad\implies\quad \exists u \in C.\quad t \lesssim_{\mathcal{N}} u\ .$$

En particulier, soit $t \in \mathcal{N}_{R\mathcal{E}} \cap \mathcal{SN}_R$ et $C =_{\text{def}} \bigcup\{\mathcal{CR}(u) \mid u \in (t)_R\}$. L'ensemble C vérifie (10.2) car si $v \in \bigcup\{\mathcal{CR}(u) \mid u \in C\}$ alors il existe u, w tels que $v \in \mathcal{CR}(u)$, $u \in \mathcal{CR}(w)$ et $w \in (t_R)$, et comme $\mathcal{CR}(u) \subseteq \mathcal{CR}(w)$, on a bien $v \in C$. Comme $(t)_R \subseteq C$ on doit donc avoir $u \in (t)_R$ tel que $t \lesssim_{\mathcal{N}} u$.

En d'autre termes, tout terme t neutre, fortement normalisant et non normal doit avoir un réduit u tel que toutes les valeurs de t soient des valeurs de u. Un tel u est un *réduit principal fort de* t.

Définition 10.4.12 (Réduit principal fort) *Étant donné* $t \in \mathcal{N}_{R\mathcal{E}} \cap \mathcal{SN}_R$ *tel que* $(t)_R \neq \emptyset$, *un terme* $u \in (t)_R$ *tel que* $t \lesssim_{\mathcal{N}} u$ *est* un réduit principal fort de t.

On désigne par $\mathcal{O}_{\mathcal{N}}$ *l'ensemble des sous ensembles non-vides de* $\mathcal{SN}_R$ *qui sont clos par le bas pour* $\lesssim_{\mathcal{N}}$.

Remarque 10.4.13 *La propriété 10.4.7 dit que l'appartenance d'un terme fortement normalisant à un candidat de réductibilité est caractérisée par l'appartenance de ses valeurs à ce candidat. Ainsi pour* $t \in \mathcal{N}_{R\mathcal{E}} \cap \mathcal{SN}_R$, *un terme* $u \in \mathcal{SN}_R$ *tel que* $t \lesssim_{\mathcal{N}} u$ *« résume » le comportement de* t *vis-à-vis des candidats de réductibilité : si* $u \in C$ *avec* $C \in \mathcal{CR}_{R\mathcal{E}}$ *alors* $t \in C$.

On peut maintenant caractériser la stabilité par union des candidats de réductibilité.

Théorème 10.4.14 *Soit une relation de réécriture* $\to_R$ *et un ensemble* $\mathcal{E}$ *de contextes d'élimination pour* $\to_R$. *Les trois points suivants sont équivalents :*

(i) $\mathcal{CR}_{R\mathcal{E}}$ *est stable par union.*

(ii) Tout terme non normal $t \in \mathcal{N}_{R\mathcal{E}} \cap \mathcal{SN}_R$ *a un réduit principal fort.*

(iii) $\mathcal{CR}_{R\mathcal{E}} = \mathcal{O}_{\mathcal{N}}$.

PREUVE.

(i) ⇒ (ii). Soit t un terme neutre non normal et fortement normalisant. Par hypothèse l'ensemble $C =_{\text{def}} \bigcup\{\mathcal{CR}(u) \mid u \in (t)_R\}$ est un candidat de réductibilité. On a donc $t \in C$ car $(t)_R \subseteq C$. Par le lemme 10.4.11, il existe donc $u \in (t)_R$ tel que $t \lesssim_{\mathcal{N}} u$.

(ii) ⇒ (iii). On a $\mathcal{CR}_{\mathcal{RE}} \subseteq \mathcal{O}_{\mathcal{N}}$ par le lemme 10.4.11. Soit $C \in \mathcal{O}_{\mathcal{N}}$ et vérifions que c'est un candidat de réductibilité.

($\mathcal{CR}0$) Si $t \in C$ et $t \to_R u$ alors $u \in \mathcal{SN}_R$ et $u \lesssim_{\mathcal{N}} t$ donc $u \in C$.

($\mathfrak{CR}1$) Soit $t \in \mathcal{N}_{R\mathcal{E}}$ tel que $(t)_R \subseteq C$. Notons que $t \in \mathcal{SN}_R$ car $C \subseteq \mathcal{SN}_R$.
Si $(t)_R = \emptyset$ alors $t \lesssim_{\mathcal{N}} u$ pour tout $u \in C$ donc $t \in C$ car C n'est pas vide.
Sinon, par hypothèse il existe $u \in (t)_R$ tel que $t \lesssim_{\mathcal{N}} u$, donc $t \in C$ car $(t)_R \subseteq C$.

(iii) $\Rightarrow$ (i). Évident. □

Remarque 10.4.15 *D'après la propriété 10.4.2 et le lemme 10.4.11, on a toujours $\mathfrak{CR}_{R\mathcal{E}} \subseteq \mathcal{O}_{\mathcal{N}}$. Ainsi, par le théorème 10.4.14, $\mathfrak{CR}_{R\mathcal{E}}$ est stable par union si et seulement si $\mathcal{O}_{\mathcal{N}} \subseteq \mathfrak{CR}_{R\mathcal{E}}$.*

Remarque 10.4.16 *Le théorème 10.4.14 nous renseigne de manière intéressante sur la structure « algébrique » de candidats de Girard : ils sont stables par union exactement lorsque ce sont les ensembles non vides de termes fortement normalisants clos par le bas pour $\lesssim_{\mathcal{N}}$.*

10.4.3 Clôture par union des biorthogonaux

Nous allons maintenant voir comment s'applique la proposition 10.4.2 à un opérateur de clôture construit par biorthogonalité, au sens de la section 9.3.5.

Soit donc deux ensembles $\mathcal{A}$ et Π et une relation $\perp\!\!\!\perp \subseteq \mathcal{A} \times \Pi$. Rappelons que par la proposition 9.3.35, $(\_)^{\perp\!\!\!\perp\perp\!\!\!\perp}$ est un opérateur de clôture, donc $(A \cap B)^{\perp\!\!\!\perp\perp\!\!\!\perp} = A \cap B$ pour tout $A, B \in \mathcal{P}(\mathcal{A})^{\perp\!\!\!\perp\perp\!\!\!\perp}$. Cependant, la biorthogonalité ne se comporte pas aussi bien avec l'union.

Proposition 10.4.17 *Soit deux ensembles $\mathcal{A}$ et Π et une relation $\perp\!\!\!\perp \subseteq \mathcal{A} \times \Pi$. Pour tout $A, B \in \mathcal{P}(\mathcal{A})$ (resp. $\mathcal{P}(\Pi)$), on a*

$$A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp} = (A \cup B)^{\perp\!\!\!\perp}, \tag{10.4}$$

$$A^{\perp\!\!\!\perp} \cup B^{\perp\!\!\!\perp} \subseteq (A \cap B)^{\perp\!\!\!\perp}. \tag{10.5}$$

Preuve. On commence par (10.4). Tout d'abord, comme $A, B \subseteq A \cup B$, on a $(A \cup B)^{\perp\!\!\!\perp} \subseteq A^{\perp\!\!\!\perp}, B^{\perp\!\!\!\perp}$ par la propriété (9.25), d'où $(A \cup B)^{\perp\!\!\!\perp} \subseteq A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp}$. Inversement, comme $A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp} \subseteq A^{\perp\!\!\!\perp}, B^{\perp\!\!\!\perp}$, en appliquant deux fois la proposition 9.3.34 on obtient $A, B \subseteq (A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp})^{\perp\!\!\!\perp}$, d'où $A \cup B \subseteq (A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp})^{\perp\!\!\!\perp}$ et $(A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp}) \subseteq (A \cup B)^{\perp\!\!\!\perp}$ en ré-appliquant la proposition 9.3.34.

En ce qui concerne (10.5), comme $A \cap B \subseteq A, B$, on a $A^{\perp\!\!\!\perp}, B^{\perp\!\!\!\perp} \subseteq (A \cap B)^{\perp\!\!\!\perp}$ par la propriété (9.25), d'où $A^{\perp\!\!\!\perp} \cup B^{\perp\!\!\!\perp} \subseteq (A \cap B)^{\perp\!\!\!\perp}$. □

Remarque 10.4.18 *Notons que l'inverse de (10.5) n'est en général pas vrai : si π est orthogonal à tout élément de $A \cap B$, il n'y a en général aucune raison pour que π soit orthogonal à tout élément de A ou à tout élément de B.*

Il suit de la propriété 10.4.17 que

$$A^{\perp\!\!\!\perp\perp\!\!\!\perp} \cup B^{\perp\!\!\!\perp\perp\!\!\!\perp} \subseteq (A^{\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp})^{\perp\!\!\!\perp} = (A \cup B)^{\perp\!\!\!\perp\perp\!\!\!\perp}, \tag{10.6}$$

mais en général $A^{\perp\!\!\!\perp\perp\!\!\!\perp} \cup B^{\perp\!\!\!\perp\perp\!\!\!\perp} \neq (A \cup B)^{\perp\!\!\!\perp\perp\!\!\!\perp}$. D'autre part, si A et B sont eux-mêmes des biorthogonaux, on a $(A \cap B)^{\perp\!\!\!\perp\perp\!\!\!\perp} = A^{\perp\!\!\!\perp\perp\!\!\!\perp} \cap B^{\perp\!\!\!\perp\perp\!\!\!\perp}$, et il suit de 10.4 que

$$(A \cap B)^{\perp\!\!\!\perp\perp\!\!\!\perp} \quad = \quad (A^{\perp\!\!\!\perp} \cup B^{\perp\!\!\!\perp})^{\perp\!\!\!\perp} \ . \tag{10.7}$$

Ainsi les biorthogonaux ne sont en général pas stables par union. Étant donné $a \in A$, on pose $a^{\perp\!\!\!\perp\perp\!\!\!\perp} =_{\text{def}} \{a\}^{\perp\!\!\!\perp\perp\!\!\!\perp}$. De même qu'avec les candidats de réductibilité à la section 10.4.2, pour comprendre l'effet de la proposition 10.4.2, il est intéressant de caractériser les ensembles de la forme $a^{\perp\!\!\!\perp\perp\!\!\!\perp}$ avec $a \in \mathcal{A}$.

De manière analogue aux candidats de réductibilité, ces ensembles sont caractérisés par un préordre, qui peut être vu comme un préordre d'observation si on voit les éléments de A comme des « programmes » et les éléments de Π comme des « contextes » (c.-à-d. des « co-programmes »).

Définition 10.4.19 *Étant donnés deux ensembles $\mathcal{A}$ et Π et une relation $\perp\!\!\!\perp \subseteq \mathcal{A} \times \Pi$, on définit la relation $\lesssim_{\perp\!\!\!\perp} \subseteq \mathcal{A} \times \mathcal{A}$ par*

$$a \lesssim_{\perp\!\!\!\perp} b \qquad \textit{si et seulement si} \qquad a^{\perp\!\!\!\perp} \subseteq b^{\perp\!\!\!\perp} \ .$$

On a donc $a \lesssim_{\perp\!\!\!\perp} b$ si et seulement si

$$\forall \pi \in \Pi. \quad a \perp\!\!\!\perp \pi \quad \Longrightarrow \quad b \perp\!\!\!\perp \pi \ .$$

La différence avec les candidats de réductibilité est que les ensembles $a^{\perp\!\!\!\perp\perp\!\!\!\perp}$ ne sont pas clos par le bas mais par le haut pour $\lesssim_{\perp\!\!\!\perp}$.

Proposition 10.4.20 *Pour tout $a \in A$ on a $a^{\perp\!\!\!\perp\perp\!\!\!\perp} = \{b \mid a \lesssim_{\perp\!\!\!\perp} b\}$.*

Il suit de la proposition 10.4.2 que l'union d'une famille de biorthogonaux est un ensemble clos par le haut pour $\lesssim_{\perp\!\!\!\perp}$.

Définition 10.4.21 *Étant donnés deux ensembles $\mathcal{A}$ et Π et une relation $\perp\!\!\!\perp \subseteq \mathcal{A} \times \Pi$, on désigne par $\mathcal{O}_{\perp\!\!\!\perp}$ l'ensemble des $A \subseteq \mathcal{A}$ non vides et clos par le haut pour $\lesssim_{\perp\!\!\!\perp}$.*

Corollaire 10.4.22 *Étant donnés deux ensembles $\mathcal{A}$ et Π et une relation $\perp\!\!\!\perp \subseteq \mathcal{A} \times \Pi$, l'ensemble $\mathcal{O}_{\perp\!\!\!\perp}$ est le plus petit ensemble contenant $\mathcal{P}^\star(\mathcal{A})^{\perp\!\!\!\perp\perp\!\!\!\perp}$ qui est clos par intersection et union de familles non-vides d'éléments de $\mathcal{P}^\star(\mathcal{A})^{\perp\!\!\!\perp\perp\!\!\!\perp}$.*

Voyons maintenant comment appliquer ces idées à la réductibilité. Soient $\to_R$ une relation de réécriture sur $\Lambda(\Sigma)$ et $\mathcal{E}$ un ensemble de contextes d'élimination pour $\to_R$.

Définition 10.4.23 *On définit $\perp\!\!\!\perp \subseteq \Lambda(\Sigma) \times \mathcal{E}$ par*

$$t \perp\!\!\!\perp E[\,] \qquad \textit{si et seulement si} \qquad E[t] \in \mathcal{SN}_R \ .$$

On désigne $\lesssim_{\perp\!\!\!\perp}$ par $\lesssim_{\mathcal{SN}}$ et $\mathcal{O}_{\perp\!\!\!\perp}$ par $\mathcal{O}_{\mathcal{SN}}$.

Rappelons que $\mathcal{P}^\star(\mathcal{SN}_R)^{\perp\!\!\!\perp\perp\!\!\!\perp} \subseteq \mathfrak{CR}_{R\mathcal{E}}$ d'après le lemme 9.3.39. Notons que $t \to_R u$ implique $t \lesssim_{\mathcal{SN}} u$ et que la relation $\lesssim_{\mathcal{SN}}$ est stable par contextes d'élimination.

Proposition 10.4.24 *Si* $t \lesssim_{\mathcal{SN}} u$ *alors pour tout* $E[\ ] \in \mathcal{E}$ *on a* $E[t] \lesssim_{\mathcal{SN}} E[u]$.

Preuve. Car par définition les contextes d'élimination sont stables par composition. □

Vérifions que $\mathcal{O}_{\mathcal{SN}}$ est un bon candidat pour être une famille de réductibilité. Tout d'abord, il est clair que $\mathcal{SN}_R \in \mathcal{O}_{\mathcal{SN}}$. De plus $\mathcal{HN}_{R\mathcal{E}} \subseteq C$ pour tout $C \in \mathcal{O}_{\mathcal{SN}}$.

Proposition 10.4.25 *Si* $C \in \mathcal{O}_{\mathcal{SN}}$ *alors* $\mathcal{HN}_{R\mathcal{E}} \subseteq C$.

Preuve. Si $t \in \mathcal{HN}_{R\mathcal{E}}$ alors par la proposition 9.3.30, on a $E[t] \in \mathcal{HN}_{R\mathcal{E}} \subseteq \mathcal{SN}_R$ pour tout $E[\ ] \in \mathcal{E} \cap \mathcal{SN}_R$. Il s'en suit que $u \lesssim_{\mathcal{SN}} t$ pour tout $u \in \mathcal{SN}_R$, donc que $t \in C$ car C n'est pas vide. □

De plus, lorsque les contextes d'élimination $\mathcal{E}$ contiennent les contextes d'élimination de la flèche, on a $\Rightarrow : \mathcal{O}_{\mathcal{SN}} \times \mathcal{O}_{\mathcal{SN}} \to \mathcal{O}_{\mathcal{SN}}$.

Proposition 10.4.26 *Supposons que* $[\ ]t \in \mathcal{E}$ *pour tout* $t \in \Lambda(\Sigma)$. *Si* $A, B \in \mathcal{O}_{\mathcal{SN}}$ *alors* $A \Rightarrow B \in \mathcal{O}_{\mathcal{SN}}$.

Preuve. Pour voir que $A \Rightarrow B \subseteq \mathcal{SN}_R$, on raisonne comme à la proposition 9.1.13, en utilisant le fait que $\mathcal{HN}_{R\mathcal{E}} \subseteq A$ par la proposition 9.3.30. De plus, $A \Rightarrow B$ est clos par le haut pour $\lesssim_{\mathcal{SN}}$ car $t \lesssim_{\mathcal{SN}} u$ implique $tv \lesssim_{\mathcal{SN}} uv$ pour tout $v \in \Lambda(\Sigma)$. Enfin, $A \Rightarrow B$ n'est pas vide car pour tout $t \in \mathcal{HN}_{R\mathcal{E}}$, comme $A \subseteq \mathcal{SN}_R$, pour tout $u \in A$ on a $tu \in \mathcal{HN}_{R\mathcal{E}}$, donc $tu \in B$ par la proposition 10.4.25. □

Exemple 10.4.27 *Dans le cas du* λ*-calcul pur, les ensembles* $C \in \mathcal{O}_{\mathcal{SN}}$ *satisfont les clauses* (SAT1) *et* (SAT2$_\beta$). *En effet, les termes de la forme* $x\vec{t}$ *sont héréditairement neutres, et on a* $t[u/x] \lesssim_{\mathcal{SN}} (\lambda x.t)u$ *par standardisation faible (lemme 9.1.9).*

La question, maintenant, est de voir ce qui permet d'avoir $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{R\mathcal{E}}$. Le raisonnement est tout à fait analogue à celui de la section 10.4.2 pour les candidats de réductibilité. Si $C \in \mathcal{O}_{\mathcal{SN}}$, il est clair que C est stable par réduction car

$$\forall t, u. \quad t \to_R u \quad \Longrightarrow \quad t \lesssim_{\mathcal{SN}} u\ .$$

Le cas de la clause ($\mathcal{CR}1$) est plus délicat. Considérons un terme neutre fortement normalisant t tel que $(t)_R$ ne soit pas vide et posons $C =_{\text{def}} \{v \mid \exists u \in (t)_R.\ u \lesssim_{\mathcal{SN}} v\}$. Pour que $t \in C$, il faut qu'il existe un terme $u \in (t)_R$ tel que $u \lesssim_{\mathcal{SN}} t$. Un tel u est un *réduit principal* de t.

Définition 10.4.28 (Réduit principal) *Étant donné* $t \in \mathcal{N}_{R\mathcal{E}} \cap \mathcal{SN}_R$ *tel que* $(t)_R \neq \emptyset$, *un terme* $u \in (t)_R$ *tel que* $u \lesssim_{\mathcal{SN}} t$ *est* un réduit principal de t.

De même que l'existence de réduits principaux forts caractérise le fait que $\mathcal{O}_{\mathcal{N}} \subseteq \mathcal{CR}_{R\mathcal{E}}$ (voir remarque 10.4.15), l'existence de réduits principaux caractérise le fait que $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{R\mathcal{E}}$.

Théorème 10.4.29 *Soit une relation de réécriture* $\to_R$ *et un ensemble* $\mathcal{E}$ *de contextes d'élimination pour* $\to_R$. *Les deux points suivants sont équivalents :*

(i) $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\mathcal{RE}}$.

(ii) Tout terme non normal $t \in \mathcal{N}_{\mathcal{RE}} \cap \mathcal{SN}_{\mathcal{R}}$ *a un réduit principal.*

Preuve.

(i) ⇒ (ii). Soit un terme non normal $t \in \mathcal{N}_{\mathcal{RE}} \cap \mathcal{SN}_{\mathcal{R}}$. Par hypothèse, l'ensemble $C =_{\text{def}} \{v \mid \exists u \in (t)_{\mathcal{R}}.\ u \lesssim_{\mathcal{SN}} v\}$ est un candidat de réductibilité, donc $t \in C$. Il s'en suit qu'il existe $u \in (t)_{\mathcal{R}}$ tel que $u \lesssim_{\mathcal{SN}} t$.

(ii) ⇒ (i). Soit $C \in \mathcal{O}_{\mathcal{SN}}$ et vérifions que c'est un candidat de réductibilité.

($\mathcal{CR}0$) Si $t \in C$ et $t \to_{\mathcal{R}} u$ alors $u \in \mathcal{SN}_{\mathcal{R}}$ et $t \lesssim_{\mathcal{SN}} u$ donc $u \in C$.

($\mathcal{CR}1$) Soit $t \in \mathcal{N}_{\mathcal{RE}}$ tel que $(t)_{\mathcal{R}} \subseteq C$. Notons que $t \in \mathcal{SN}_{\mathcal{R}}$ car $C \subseteq \mathcal{SN}_{\mathcal{R}}$.

Si $(t)_{\mathcal{R}} = \emptyset$ alors comme c'est un terme neutre, on a $E[t] \in \mathcal{SN}_{\mathcal{R}}$ pour tout $E[\,] \in \mathcal{SN}_{\mathcal{R}}$. Il s'en suit que $u \lesssim_{\mathcal{SN}} t$ pour tout $u \in C$ donc $t \in C$ car C n'est pas vide. Sinon, par hypothèse il existe $u \in (t)_{\mathcal{R}}$ tel que $u \lesssim_{\mathcal{SN}} t$, donc $t \in C$ car $(t)_{\mathcal{R}} \subseteq C$. □

La propriété du « réduit principal » est très proche de la propriété du « réduit principal fort » définie en 10.4.12 et utilisée avec les candidats de réductibilité. Le nom « réduit principal fort » vient du fait que tout réduit principal fort de t est un réduit principal de t. Cela est dû à la proposition suivante.

Proposition 10.4.30 *Pour tous* $t, u \in \mathcal{SN}_{\mathcal{R}}$*, on a*

$$t \lesssim_{\mathcal{N}} u \quad \Longrightarrow \quad u \lesssim_{\mathcal{SN}} t\ .$$

Preuve. Soit $u \in \mathcal{SN}_{\mathcal{R}}$. On montre que pour tout $t \in \mathcal{SN}_{\mathcal{R}}$ et tout $E[\,] \in u^{\perp\!\!\perp\perp\!\!\perp}$, on a $E[t] \in \mathcal{SN}_{\mathcal{R}}$. On raisonne par induction sur les paires $(E[\,], t)$ ordonnées par l'extension produit de $\to_{\mathcal{R}}$.

Soit v tel que $E[t] \to_{\mathcal{R}} v$. Il y a deux cas.

— Si $t \notin \mathcal{N}_{\mathcal{RE}}$, alors $u \to_{\mathcal{R}}^{*} t$ car $t \lesssim_{\mathcal{N}} u$, donc $E[u] \to_{\mathcal{R}}^{*} v$ et $v \in \mathcal{SN}_{\mathcal{R}}$.

— Sinon, $v = E'[t']$ avec $(E[\,], t) \to_{\mathcal{R}} (E'[\,], t')$. Si $E[\,] \to_{\mathcal{R}} E'[\,]$, comme $E'[u] \in \mathcal{SN}_{\mathcal{R}}$, on a $v = E'[t] \in \mathcal{SN}_{\mathcal{R}}$ par hypothèse d'induction. Sinon $v = E[t']$ avec $t \to_{\mathcal{R}} t'$. Comme $t' \lesssim_{\mathcal{N}} t$, on a $t' \lesssim_{\mathcal{N}} u$, donc $v = E[t'] \in \mathcal{SN}_{\mathcal{R}}$ par hypothèse d'induction. □

Notons que l'inverse n'est pas vrai en général.

Exemple 10.4.31 *On a* $[\lambda x.y]\vec{t} \in \mathcal{SN}_{\beta}$ *pour tout* $[\,]\vec{t} \in \mathcal{SN}_{\beta}$*, donc* $\lambda x.x \lesssim_{\mathcal{SN}} \lambda x.y$*, alors que* $\lambda x.x$ *et* $\lambda x.y$ *ne sont pas comparables avec* $\lesssim_{\mathcal{N}}$.

Ainsi, si $\mathcal{CR}_{\mathcal{RE}}$ est stable par union alors $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\mathcal{RE}}$.

Lemme 10.4.32 *Soit une relation de réécriture* $\to_{\mathcal{R}}$ *et un ensemble* $\mathcal{E}$ *de contextes d'élimination pour* $\to_{\mathcal{R}}$*. Si* $\mathcal{CR}_{\mathcal{RE}}$ *est stable par union alors* $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\mathcal{RE}}$.

Preuve. Si $\mathcal{CR}_{\mathcal{RE}}$ est stable par union, alors par le théorème 10.4.14, tout terme t neutre non normal et fortement normalisant a un réduit principal fort u. Or, u est un réduit principal de t par la proposition 10.4.30. Il s'en suit que $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\mathcal{RE}}$ par le théorème 10.4.29. □

Remarque 10.4.33 *Il est intéressant de voir si les relation d'équivalences :*

$$\simeq_{\mathcal{N}} \ =_{def} \ \lesssim_{\mathcal{N}} \cap \gtrsim_{\mathcal{N}} \qquad et \qquad \simeq_{\mathcal{SN}} \ =_{def} \ \lesssim_{\mathcal{SN}} \cap \gtrsim_{\mathcal{SN}}$$

engendrées par les préordres $\lesssim_{\mathcal{N}}$ et $\lesssim_{\mathcal{SN}}$ peuvent être étendues en des congruences cohérentes sur $\Lambda(\Sigma)$.

Dans le cas du λ-calcul ce n'est pas le cas. En effet, $\simeq_{\mathcal{N}}$ identifie tous les termes héréditairement neutres, y compris ceux qui ne sont pas βη-convertibles et $\simeq_{\mathcal{SN}}$ identifie les termes $\lambda x.y$ et $\lambda x.z$ qui sont deux termes en forme βη-normale. D'après le théorème de Böhm (théorème 3.5.6), les congruences engendrées par ces relations d'équivalence identifient donc tous les λ-termes purs.

10.5 Application aux systèmes de réécriture simples

Nous allons maintenant voir comment s'adaptent à la réécriture simple les conditions établies aux sections 10.4.2 et 10.4.3.

On se base sur les candidats de réductibilité $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ définis en 10.2.1.

Définition 10.5.1 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. On dit que $\mathcal{R}$ a la* propriété du réduit principal *(resp.* propriété du réduit principal fort*) si tout terme fortement $\beta\mathcal{R}$-normalisant et non normal de la forme $f(t_1,\dots,t_n)$ a un réduit principal (resp. un réduit principal fort).*

Grâce au lemme de standardisation faible pour $\mapsto_\beta$ (lemme 9.1.9), les termes de la forme $E[(\lambda x.t)u]$ on un réduit principal. L'existence de réduits principaux (forts) pour les termes neutres de $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ se ramène alors à la propriété du réduit principal (fort) pour $\mathcal{R}$.

Lemme 10.5.2 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. Alors $\mathcal{R}$ a la propriété du réduit principal (fort) si et seulement si tout terme neutre, non normal et fortement normalisant a un réduit principal (fort).*

Preuve. Dans les deux cas, le sens « seulement si » est trivial car les termes de la forme $f(t_1,\dots,t_n)$ sont neutres.

Considérons maintenant le cas « si ». Un terme neutre t est de la forme $h\vec{v}$ où h est soit une variable, un β-rédex ou un terme de la forme $f(t_1,\dots,t_n)$.

- $h = x \in \mathcal{X}$. Dans ce cas, comme t est héréditairement neutre, il ne se réduit pas vers un terme de la forme $\lambda x.u$, donc s'il n'est pas normal, tous ses réduits sont des réduits principaux forts de t, ce sont en outre des réduits principaux de t par la proposition 10.4.30.
- $h = (\lambda x.u)v$. Dans ce cas, le terme $u[v/x]\vec{v}$ est un réduit principal fort de t.

  Cela se montre par induction sur les tuples $(u, v, \vec{v})$ ordonnés par l'extension produit de $\to_{\beta\mathcal{R}}$ en utilisant la standardisation faible (lemme 9.1.9). Soit donc $w \notin \mathcal{N}_{\sqcap\sqcup\mathcal{R}}$ tel que $t \to^*_{\beta\mathcal{R}} w$. On doit montrer que $u[v/x]\vec{v} \to^*_{\beta\mathcal{R}} w$. Soit w' tel que $t \to_{\beta\mathcal{R}} w' \to^*_{\beta\mathcal{R}} w$. Si $w' \neq u[v/x]\vec{v}$ alors par standardisation faible, $w' = (\lambda x.u')v'\vec{v}'$ avec

$(u, v, \vec{v}) \to_{\beta\mathcal{R}} (u', v', \vec{v}')$ et $u[v/x]\vec{v} \to^*_{\beta\mathcal{R}} u'[v'/x]\vec{v}'$. Par hypothèse d'induction, on a donc $u'[v'/x]\vec{v}' \to^*_{\beta\mathcal{R}} w$, soit $u[v/x]\vec{v} \to^*_{\beta\mathcal{R}} w$.
Comme $u[v/x]\vec{v}$ est réduit principal fort de t, c'en est un réduit principal par la proposition 10.4.30.

$h = f(t_1, \dots, t_n)$. Si $f \in \mathcal{C}$, alors $t \in \mathcal{HN}_{\sqcap\sqcup\mathcal{R}}$ et on raisonne comme dans le cas $h = x$.
Sinon, par la proposition 10.4.9 (resp. 10.4.24), le réduit principal (fort) de h est un réduit principal (fort) de t. □

Remarque 10.5.3 *En particulier, dans le cas du λ-calcul pur, les candidats de réductibilité pour les contextes d'élimination $\mathcal{E}_\Rightarrow$ sont stables par union. Cela a aussi été remarqué par Tatsuta [Tat07].*

Le lemme 10.5.2 nous permet de vérifier facilement si les termes neutres, fortement normalisants et non normaux pour un système de réécriture simple donné ont un réduit principal (fort).

Exemple 10.5.4 *Avec le système non confluent*

$$f(x) \quad \mapsto_{\mathcal{R}} \quad x \qquad f(x) \quad \mapsto_{\mathcal{R}} \quad a \qquad f(x) \quad \mapsto_{\mathcal{R}} \quad b\ ,$$

les candidats $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ sont stables par union. En effet, les termes a et b sont héréditairement neutres, donc tout $t \in \Lambda(\Sigma)$ est un réduit principal fort de $f(t)$. On a donc la stabilité par union de $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ en combinant le lemme 10.5.2 au théorème 10.4.14.

De plus, on peut maintenant comparer précisément les réduits principaux forts et les réduits principaux. On sait déjà par la proposition 10.4.30 que les réduits principaux forts sont des réduits principaux. On peut maintenant montrer que l'inverse n'est pas vrai en général.

Exemple 10.5.5 *Avec le système confluent*

$$p \quad \mapsto_{\mathcal{R}} \quad \lambda x.c_1 \qquad p \quad \mapsto_{\mathcal{R}} \quad \lambda x.c_2 \qquad c_i \quad \mapsto_{\mathcal{R}} \quad d\ ,$$

les candidats de réductibilité ne sont pas clos par union mais on a $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$.

En effet, comme $\lambda x.c_1$ et $\lambda x.c_2$ sont deux termes non neutres distincts, le terme p n'a pas de réduit principal fort, et on en déduit que $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ n'est pas stable par union en combinant le lemme 10.5.2 au théorème 10.4.14.

D'autre part, pour tout $\vec{v}$, on a $(\lambda x.c_1)\vec{v} \in \mathcal{SN}_{\beta\mathcal{R}}$ si et seulement si $(\lambda x.c_2)\vec{v} \in \mathcal{SN}_{\beta\mathcal{R}}$, donc les termes $\lambda x.c_1$ et $\lambda x.c_2$ sont tous les deux des réduits principaux de p. Il s'en suit que $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ en combinant le lemme 10.5.2 au théorème 10.4.29.

Il s'en suit que la propriété du réduit principal est une condition suffisante strictement plus forte que la propriété du réduit principal fort pour la normalisation forte des termes typables dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$.

Lemme 10.5.6 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. Si $\mathcal{R}$ a la propriété du réduit principal alors tout terme typable dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ est fortement normalisable.*

PREUVE. Pour tout $A, B \in \mathcal{O}_{\mathcal{SN}}$ on a $A \cap B, A \cup B \in \mathcal{O}_{\mathcal{SN}}$ par la proposition 10.4.2, et $A \Rightarrow B \in \mathcal{O}_{\mathcal{SN}}$ par la proposition 10.4.26. De plus, en combinant le lemme 10.5.2 au théorème 10.4.29, on a $\mathcal{O}_{\mathcal{SN}} \subseteq \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$. Comme $\mathcal{SN}_{\beta\mathcal{R}} \in \mathcal{O}_{\mathcal{SN}}$, en utilisant le théorème 10.2.6, on peut appliquer le lemme 10.3.6 avec l'interprétation

$$\begin{array}{lll} [\![o]\!]_{\mathcal{O}} & =_{\text{def}} & \mathcal{SN}_{\beta\mathcal{R}} \ , \\ [\![U \Rightarrow T]\!]_{\mathcal{O}} & =_{\text{def}} & [\![U]\!]_{\mathcal{O}} \Rightarrow [\![T]\!]_{\mathcal{O}} \ , \\ [\![T \sqcap U]\!]_{\mathcal{O}} & =_{\text{def}} & [\![T]\!]_{\mathcal{O}} \cap [\![U]\!]_{\mathcal{O}} \ , \\ [\![T \sqcup U]\!]_{\mathcal{O}} & =_{\text{def}} & [\![T]\!]_{\mathcal{O}} \cup [\![U]\!]_{\mathcal{O}} \ , \end{array}$$

d'où la $\beta\mathcal{R}$-normalisation forte des termes typables dans $\boldsymbol{\lambda}_{(\sqcup\,E)\mathcal{R}}(\Sigma)$. □

Remarque 10.5.7 *D'après la définition 5.1.2, un système de réécriture simple $\mathcal{R}$ est orthogonal si chaque symbole est défini par au plus une règle. Il a donc la propriété du réduit principal fort, et par le lemme 10.5.6, les termes typables dans $\boldsymbol{\lambda}_{(\sqcup\,E)\mathcal{R}}(\Sigma)$ sont fortement $\beta\mathcal{R}$-normalisants.*

Enfin, notons que les propriétés du réduit principal (fort) peuvent se caractériser de manière élégante pour les systèmes de réécriture simples.

Remarque 10.5.8 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$.*

(i) $\mathcal{R}$ a la propriété du réduit principal fort si et seulement si pour tout $f \in \mathcal{F}$ et tous $\vec{t}$ tels que $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$, il existe une règle $f(\vec{x}) \mapsto_{\mathcal{R}} d$ telle que

$$d[\vec{t}/\vec{x}] \quad = \quad \sup_{\lesssim_{\mathcal{N}}} \{r[\vec{t}/\vec{x}] \mid f(\vec{t}) \mapsto_{\mathcal{R}} r\} \quad \textit{modulo} \quad \simeq_{\mathcal{N}} \ ,$$

où $\simeq_{\mathcal{N}}$ est la plus petite relation d'équivalence contenant $\lesssim_{\mathcal{N}}$, définie à la remarque 10.4.33.

(ii) $\mathcal{R}$ a la propriété du réduit principal si et seulement si pour tout $f \in \mathcal{F}$ et tous $\vec{t}$ tels que $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$, il existe une règle $f(\vec{x}) \mapsto_{\mathcal{R}} d$ telle que

$$d[\vec{t}/\vec{x}] \quad = \quad \inf_{\lesssim_{\mathcal{SN}}} \{r[\vec{t}/\vec{x}] \mid f(\vec{t}) \mapsto_{\mathcal{R}} r\} \quad \textit{modulo} \quad \simeq_{\mathcal{SN}} \ ,$$

où $\simeq_{\mathcal{SN}}$ est la plus petite relation d'équivalence contenant $\lesssim_{\mathcal{SN}}$, définie à la remarque 10.4.33.

10.6 Comparaison des candidats de réductibilité et des ensembles saturés

Dans cette section, on compare les candidats de réductibilité de Girard et les ensembles saturés de Tait, en rapport avec les outils développés aux sections 10.4.2 et 10.4.3.

10.6.1 Lambda-calcul pur

On commence par le λ-calcul pur. Rappelons qu'à l'exemple 9.3.17.(i), nous avons défini l'ensemble $\mathcal{CR}_\beta$ des candidats de réductibilité pour $\to_\beta$ dans les contextes d'élimination $\mathcal{E}_\Rightarrow$ générés par la grammaire

$$E[\,] \in \mathcal{E}_\Rightarrow \quad ::= \quad [\,] \quad | \quad E[\,]\,t\,,$$

où $t \in \Lambda$.

La stabilité par union de $\mathcal{CR}_\beta$ est une conséquence du lemme 10.5.2. Nous allons voir que l'on peut aussi obtenir ce résultat on montrant que $\mathcal{CR}_\beta$ est exactement l'ensemble des ensembles saturés $S \in \mathcal{SAT}_\beta$ qui sont stables par union.

Rappelons que les ensembles saturés pour le λ-calcul, définis en 9.1.7, sont les ensembles $S \subseteq \mathcal{SN}_\beta$ tels que

($\mathcal{SAT}1$) si $E[\,] \in \mathcal{SN}_\beta \cap \mathcal{E}_\Rightarrow$ et $x \in \mathcal{X}$ alors $E[x] \in S$,

($\mathcal{SAT}2$) pour tout $E[\,] \in \mathcal{E}_\Rightarrow$, si $E[t[u/x]] \in S$ et $u \in \mathcal{SN}_\beta$ alors $E[(\lambda x.t)u]$.

Définition 10.6.1 *On désigne par $\mathcal{SAT}_\to$ l'ensemble des $S \in \mathcal{SAT}_\beta$ tels que*

($\mathcal{SAT}0$) *si $t \in S$ et $t \to_\beta u$ alors $u \in S$.*

Lemme 10.6.2 $\mathcal{CR}_\beta = \mathcal{SAT}_\to$.

Preuve. Si $C \in \mathcal{CR}_\beta$ alors C satisfait ($\mathcal{SAT}0$) par ($\mathcal{CR}0$), et les clauses ($\mathcal{SAT}1$) et ($\mathcal{SAT}2$) suivent du lemme 9.3.18.

Inversement, si $S \in \mathcal{SAT}_\to$ alors S satisfait ($\mathcal{CR}0$). Considérons le cas de ($\mathcal{CR}1$). Si t est un terme neutre tel que $(t)_\beta \subseteq S$, alors on a $t \in \mathcal{SN}_\beta$. De plus, soit $t = E[x]$ et $t \in S$ par ($\mathcal{SAT}1$), soit $t = E[(\lambda x.u)v]$ et comme $E[u[v/x]] \in S$, on a $t \in S$ par ($\mathcal{SAT}2$). □

Comme la clause ($\mathcal{SAT}0$) est préservée par union, on a la stabilité par union de $\mathcal{SAT}_\to$ par le théorème 10.4.3, d'où la stabilité par union de $\mathcal{CR}_\beta$ par le lemme 10.6.2.

10.6.2 Lambda-calcul avec produits

Abordons maintenant le cas du λ-calcul avec produits $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0)$.

Rappelons qu'à l'exemple 9.3.17.(ii), nous avons défini l'ensemble $\mathcal{CR}_{\beta\pi}$ des candidats de réductibilité pour $\to_{\beta\pi}$ dans les contextes d'élimination $\mathcal{E}_{\Rightarrow\times}$ générés par la grammaire

$$E[\,] \in \mathcal{E}_{\Rightarrow\times} \quad ::= \quad [\,] \quad | \quad E[\,]\,t \quad | \quad \pi_1 E[\,] \quad | \quad \pi_2 E[\,]\,,$$

où $t \in \Lambda(\Sigma_\pi)$. On désigne par $\mathcal{N}_{\beta\pi}$ (resp. $\mathcal{HN}_{\beta\pi}$) l'ensemble des termes neutres correspondant (resp. héréditairement neutres).

Nous allons montrer que les candidats de réductibilité $\mathcal{CR}_{\beta\pi}$ sont stables par union. Nous avons vu à la section 10.4.1 les ensembles saturés $\mathcal{SAT}_{\beta\pi}$ forment une famille de réductibilité stable par union. Cependant, contrairement au cas du λ-calcul pur, les ensembles de $\mathcal{SAT}_{\beta\pi}$ clos par réduction ne sont pas tous des candidats de réductibilité. Cela est dû au fait que, comme on considère des termes potentiellement non typables, $\mathcal{HN}_{\beta\pi}$

contient des termes qui ne sont pas de la forme $E[x]$ avec $E[\ ] \in \mathcal{E}_{\beta\pi}$, comme par exemple $\pi_1\lambda x.t$.

Pour montrer la stabilité par union de $\mathcal{CR}_{\beta\pi}$, nous passons par des ensembles saturés modifiés, dont le plus petit élément est $\mathcal{HN}_{\beta\pi}$.

Définition 10.6.3 *L'ensemble* $\mathcal{SAT}^m_{\beta\pi}$ *des* ensembles $\beta\pi$-saturés modifiés *est l'ensemble des* $S \subseteq \mathcal{SN}_{\beta\pi}$ *tels que*

($\mathcal{SAT}0$) *si* $t \in S$ *et* $t \to_{\beta\pi} u$ *alors* $u \in S$,

($\mathcal{SAT}1_m$) $\mathcal{HN}_{\beta\pi} \subseteq S$,

($\mathcal{SAT}2_\Rightarrow$) *pour tout* $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$, *tout* $t \in \Lambda(\Sigma_\pi)$ *et tout* $u \in \mathcal{SN}_{\beta\pi}$, *si* $E[t[u/x]] \in S$ *alors* $E[(\lambda x.t)u] \in S$,

($\mathcal{SAT}2_{\times 1}$) *pour tout* $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$ *et tout* $t_1, t_2 \in \Lambda(\Sigma_\pi)$, *si* $t_2 \in \mathcal{SN}_{\beta\pi}$ *et* $E[t_1] \in S$ *alors* $E[\pi_1(t_1, t_2)] \in S$,

($\mathcal{SAT}2_{\times 2}$) *pour tout* $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$ *et tout* $t_1, t_2 \in \Lambda(\Sigma_\pi)$, *si* $t_1 \in \mathcal{SN}_{\beta\pi}$ *et* $E[t_2] \in S$ *alors* $E[\pi_2(t_1, t_2)] \in S$.

Pour montrer que les ensembles saturés modifiés satisfont ($\mathcal{CR}1$), on utilise un résultat intermédiaire. Rappelons que les contextes d'éliminations atomiques, définis à la section 9.1.3 sont générés par la grammaire :

$$\epsilon[\ ] \quad ::= \quad [\ ]\,t \quad | \quad \pi_1[\ ] \quad | \quad \pi_2[\ ]\,.$$

Proposition 10.6.4 *Si* $t \in (\mathcal{N} \cap \mathcal{SN}_{\beta\pi}) \setminus \mathcal{HN}_{\beta\pi}$, *alors* $t = E[u]$ *avec* $E[\ ] \in \mathcal{E}_{\Rightarrow\times}$ *et* u *est un* $\beta\pi$*-rédex.*

Preuve. On raisonne par induction sur t. Comme t a un réduit non neutre, ce n'est pas une variable, donc t est de la forme $\epsilon[u]$. Si u n'est pas neutre alors $\epsilon[u]$ est un $\beta\pi$-rédex. Sinon, u un réduit non neutre et on conclut par hypothèse d'induction. □

Lemme 10.6.5 $\mathcal{CR}_{\beta\pi} = \mathcal{SAT}^m_{\beta\pi}$.

Preuve. Si $C \in \mathcal{CR}_{\beta\pi}$ alors C satisfait la clause ($\mathcal{SAT}0$) par ($\mathcal{CR}0$), la clause ($\mathcal{SAT}1_m$) par la proposition 9.3.28, et les clauses ($\mathcal{SAT}2_\beta$) et ($\mathcal{SAT}2_{\pi i}$) le lemme 9.3.18.

Inversement, si $S \in \mathcal{SAT}^m_{\beta\pi}$ alors S satisfait ($\mathcal{CR}0$). Considérons le cas de ($\mathcal{CR}1$). Soit t un terme neutre tel que $(t)_{\beta\pi} \subseteq S$. Notons que $t \in \mathcal{SN}_{\beta\pi}$. Si $t \in \mathcal{HN}_{\beta\pi}$ alors $t \in S$ par ($\mathcal{SAT}1_m$). Sinon, par la proposition 10.6.4, t est de la forme $E[u]$ où u est un $\beta\pi$-rédex et on conclut par ($\mathcal{SAT}2_\beta$) si c'est un β-rédex et par ($\mathcal{SAT}2_{\pi i}$) si c'est un π-rédex. □

Comme $\mathcal{SAT}^m_{\beta\pi}$ est stable par union (simple variation de la preuve du théorème 10.4.3), il s'en suit que $\mathcal{CR}_{\beta\pi}$ est stable par union, et donc par le théorème 10.4.14 que les termes neutres, fortement normalisants et non normaux ont un réduit principal fort.

Remarque 10.6.6 *D'après le théorème 10.4.14 (voir aussi la remarque 10.4.16), le lemme 10.6.5 implique que l'ensemble* $\mathcal{CR}_{\beta\pi}$ *est exactement l'ensemble des sous ensembles non vides de* $\mathcal{SN}_{\beta\pi}$ *clos par le bas pour* $\lesssim_{\mathcal{N}_{\beta\pi}}$.

10.6.3 Systèmes de réécriture simples

On termine cette section par les systèmes de réécriture simples. On a vu à la section 10.5 les propriétés du réduit principal fort et du réduit principal. On va voir que l'on peut définir des ensembles saturés clos par union pour les systèmes vérifiant la propriété du réduit principal.

Rappelons que les termes neutre de $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ sont exactement les termes de la forme $h\vec{t}$ où h est soit une variable, un β-rédex, ou un terme de la forme $f(\vec{t})$.

Définition 10.6.7 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$ qui satisfait la propriété du réduit principal. L'ensemble $\mathbb{SAT}_{\sqcap\sqcup\mathcal{R}}$ des* ensembles $\sqcap\sqcup\mathcal{R}$-saturés *est l'ensemble des $S \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ tels que*

($\mathbb{SAT}1_{\sqcap\sqcup\mathcal{R}}$) $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq S$,

($\mathbb{SAT}2_{\sqcap\sqcup\mathcal{R}}$) *si* $t \in \mathcal{N}_{\sqcap\sqcup\mathcal{R}}$ *et* $t \to_{\beta\mathcal{R}} u$ *avec* $u \lesssim_{\mathcal{SN}} t$ *et* $u \in S$ *alors* $t \in S$.

Remarque 10.6.8 *Les ensembles $\sqcap\sqcup\mathcal{R}$-saturés satisfont les clauses ($\mathbb{SAT}1$) et ($\mathbb{SAT}2_\beta$). En effet, ($\mathbb{SAT}1$) suit du fait que les termes de la forme $x\vec{t}$ sont héréditairement neutres, et ($\mathbb{SAT}2_\beta$) du fait que $(\lambda x.t)u$ est un terme neutre tel que $t[u/x] \lesssim_{\mathcal{SN}} (\lambda x.t)u$ par standardisation faible.*

Il est aisé de voir que $\mathbb{SAT}_{\sqcap\sqcup\mathcal{R}}$ est stable par union et intersection. Pour que ce soit une famille de réductibilité stable par union, il reste à vérifier que la flèche préserve les ensembles $\sqcap\sqcup\mathcal{R}$-saturés.

Proposition 10.6.9 *Si $A, B \in \mathbb{SAT}_{\sqcap\sqcup\mathcal{R}}$ alors $A \Rightarrow B \in \mathbb{SAT}_{\sqcap\sqcup\mathcal{R}}$.*

Preuve. Pour voir que $A \Rightarrow B \subseteq \mathcal{SN}_{\mathcal{R}}$, on raisonne comme à la proposition 9.1.13, en utilisant le fait que $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq A$ par ($\mathbb{SAT}1_{\sqcap\sqcup\mathcal{R}}$). De plus, si $t \in \mathcal{HN}_{\sqcap\sqcup\mathcal{R}}$, alors comme $A \subseteq \mathcal{SN}_{\beta\mathcal{R}}$, pour tout $u \in A$ on a $tu \in \mathcal{HN}_{\sqcap\sqcup\mathcal{R}}$ par la proposition 9.3.30, donc $tu \in B$ par ($\mathbb{SAT}1_{\sqcap\sqcup\mathcal{R}}$). Il s'en suit que $t \in A$. Enfin, si $u \in A \Rightarrow B$, $t \in \mathcal{N}_{\sqcap\sqcup\mathcal{R}}$ et $u \lesssim_{\mathcal{SN}} t$, alors par la proposition 10.4.24, on a $uv \lesssim_{\mathcal{SN}} tv$ pour tout $v \in A$. Donc $tv \in B$ par ($\mathbb{SAT}2_{\sqcap\sqcup\mathcal{R}}$) car $tv \in \mathcal{N}_{\sqcap\sqcup\mathcal{R}}$. □

On en déduit une interprétation $[\![\_]\!] : \mathcal{T}_{\sqcap\sqcup} \to \mathbb{SAT}_{\sqcap\sqcup\mathcal{R}}$ qui est valide et adéquate pour $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ lorsque $\mathcal{R}$ a la propriété du réduit principal.

Lemme 10.6.10 *Soit $\mathcal{R}$ un système de réécriture simple qui a la propriété du réduit principal. Étant donné $[\![o]\!] \in \mathbb{SAT}_{\sqcap\sqcup\mathcal{R}}$, l'interprétation $[\![\_]\!]$ telle que*

$$\begin{array}{lcl} [\![U \Rightarrow T]\!] & = & [\![U]\!] \Rightarrow [\![T]\!]\ , \\ [\![T \sqcap U]\!] & = & [\![T]\!] \cap [\![U]\!]\ , \\ [\![T \sqcup U]\!] & = & [\![T]\!] \cup [\![U]\!]\ , \end{array}$$

est valide et adéquate pour $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$.

PREUVE. Pour tout $T \in \mathcal{T}_{\sqcap\sqcup}$, on a $[\![T]\!] \in \mathfrak{SAT}_{\sqcap\sqcup\mathcal{R}}$ car $\mathfrak{SAT}_{\sqcap\sqcup\mathcal{R}}$ est stable par $\_ \cap \_$, $\_ \cup \_$ et $\_ \Rightarrow \_$. Par le lemme 10.3.6, il faut vérifier que $[\![\_]\!]$ est adéquate pour $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$. Comme $\mathfrak{SAT}_{\sqcap\sqcup\mathcal{R}}$ satisfait les clauses ($\mathfrak{SAT}1$) et ($\mathfrak{SAT}2_\beta$), il reste à traiter le cas de la règle (FUN).

$$(\text{FUN})\ \frac{\Gamma \vdash \vec{t} : \vec{T} \qquad \forall f(\vec{x}) \mapsto_{\mathcal{R}} r,\quad \Gamma, \vec{x} : \vec{T} \vdash r : U}{\Gamma \vdash f(\vec{t}) : U}$$

Soit $\sigma \models \Gamma$. Par hypothèse d'induction, on a $\vec{t}\sigma \in [\![\vec{T}]\!]$ et pour toute règle $f(\vec{x}) \mapsto_{\mathcal{R}} r$, on a $r\sigma[\vec{t}\sigma/\vec{x}] \in [\![U]\!]$, soit $r[\vec{t}\sigma/\vec{x}] \in [\![U]\!]$ car $FV(r) \subseteq \vec{x}$.

On doit montrer que $f(\vec{t}\sigma) \in [\![U]\!]$. Comme $\mathcal{R}$ a la propriété du réduit principal, il existe une règle $f(\vec{x}) \mapsto_{\mathcal{R}} r$ telle que $r[\vec{t}\sigma/\vec{x}] \lesssim_{\mathcal{SN}} f(\vec{t}\sigma)$. Comme $f(\vec{t}\sigma)$ est un terme neutre, on a $f(\vec{t}\sigma) \in [\![U]\!]$ par ($\mathfrak{SAT}2_{\sqcap\sqcup\mathcal{R}}$). □

Remarque 10.6.11 *La propriété du réduit principal nous permet donc de définir des ensembles saturés adéquats pour la réécriture simple, ce qui constitue une ébauche de réponse à la question soulevée à la section 9.3.3 sur l'intégration de la réécriture aux ensembles saturés. Ainsi, l'étude de la stabilité par union de familles de réductibilité nous a appris des choses sur l'intégration de la réécriture à la réductibilité.*

D'autre part, il est aisé de voir que les ensembles $\sqcap\sqcup\mathcal{R}$-saturés qui sont stables par réduction sont des éléments de $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ lorsque $\mathcal{R}$ a la propriété du réduit principal. Si de plus $\mathcal{R}$ a la propriété du réduit principal fort, alors $\mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ est exactement l'ensemble des ensembles $\sqcap\sqcup\mathcal{R}$-saturés qui sont stables par réduction.

10.7 Complétude des biorthogonaux pour la réécriture simple

Nous avons vu à la section 10.3 qu'il pouvait y avoir des systèmes de réécriture confluents qui n'admettent pas de famille de réductibilité qui soit stable par union. D'autre part, nous avons donné aux sections 10.5 et 10.6 des conditions suffisantes pour obtenir des interprétations stables par union basées sur les candidats de réductibilité et sur les ensembles saturés.

Dans cette section, nous nous intéressons à une question complémentaire : existe-t-il des interprétations non stables par union, mais qui pour autant soient adéquates pour la règle ($\sqcup$ E) ?

Nous y répondons en montrant, pour tout système de réécriture simple $\mathcal{R}$, que les termes typables dans le système $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$ sont fortement $\beta\mathcal{R}$-normalisants si et seulement si une interprétation basée sur une certaine famille de biorthogonaux est adéquate pour le typage dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$. Ainsi, ces biorthogonaux sont complets par rapport à la normalisation forte des termes typables dans $\boldsymbol{\lambda}_{(\sqcup\,\mathrm{E})\mathcal{R}}(\Sigma)$.

Notre utilisation des biorthogonaux pour interpréter ($\sqcup$ E) est inspirée de [VM04]. Notons cependant que nous nous intéressons ici à la normalisation forte, alors que ce sont des propriétés du type « preservation and progress » qui sont étudiées dans [VM04]. Rappelons que les ensembles clos par biorthogonalité sont caractérisés par un ensemble de contextes avec lesquels tous les termes de l'ensemble interagissent de manière déterminée.

Bien que les biorthogonaux ne soient pas stables par union, ils ont un comportement intéressant. En effet, d'après (10.6),

$$(A \cup B)^{\perp\!\!\perp\perp\!\!\perp} \quad = \quad (A^{\perp\!\!\perp} \cap B^{\perp\!\!\perp})^{\perp\!\!\perp} \ .$$

Ainsi, on a $a \in (A \cup B)^{\perp\!\!\perp\perp\!\!\perp}$ si et seulement si a est orthogonal à tout $\pi \in A^{\perp\!\!\perp} \cap B^{\perp\!\!\perp}$. Considérons maintenant la règle $(\sqcup\,\mathrm{E})$:

$$\frac{\Gamma \vdash t : T_1 \sqcup T_2 \qquad \Gamma, x : T_1 \vdash c : C \qquad \Gamma, x : T_2 \vdash c : C}{\Gamma \vdash c[t/x] : C}$$

Soient un ensemble de contextes $\mathcal{E}$, et une interprétation $(\!|\_|\!) : \mathcal{T}_{\sqcap\sqcup} \to \mathcal{P}(\mathcal{SN}_{\beta\mathcal{R}})^{\perp\!\!\perp\perp\!\!\perp}$ où $t \perp\!\!\!\perp E[\ ]$ si et seulement si $E[t] \in \mathcal{SN}_{\beta\mathcal{R}}$. Supposons de plus que

— $t \in (\!|T_1 \sqcup T_2|\!) = ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp\perp\!\!\perp}$,

— $c[u/x] \in (\!|C|\!)$ pour tout $u \in (\!|T_1|\!) \cup (\!|T_2|\!)$.

On doit montrer $c[t/x] \in (\!|C|\!)$, c'est-à-dire $E[c[t/x]] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $E[\ ] \in (\!|C|\!)^{\perp\!\!\perp}$. Notons que c'est évident dans le cas où $t \in (\!|T_1|\!) \cup (\!|T_2|\!)$, mais cela n'est pas suffisant parce qu'en général $(\!|T_1|\!) \cup (\!|T_2|\!) \subsetneq ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp\perp\!\!\perp}$. Il nous faut donc utiliser des propriétés liées à la biorthogonalité. Par hypothèse, on a

$$\Big(E[\ ] \in (\!|C|\!)^{\perp\!\!\perp} \quad \wedge \quad u \in (\!|T_1|\!) \cup (\!|T_2|\!)\Big) \quad \Longrightarrow \quad c[u/x] \perp\!\!\!\perp E[\ ] \ ,$$

donc si $E[c[[\ ]/x]] \in \mathcal{E}$,

$$\Big(E[\ ] \in (\!|C|\!)^{\perp\!\!\perp} \quad \wedge \quad u \in (\!|T_1|\!) \cup (\!|T_2|\!)\Big) \quad \Longrightarrow \quad u \perp\!\!\!\perp E[c[[\ ]/x]] \ .$$

Il s'en suit que $E[c[[\ ]/x]] \in ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp}$ dès lors que $E[\ ] \in (\!|C|\!)^{\perp\!\!\perp}$. Ainsi, comme $t \in ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp\perp\!\!\perp}$, on a

$$E[\ ] \in (\!|C|\!)^{\perp\!\!\perp} \quad \Longrightarrow \quad t \perp\!\!\!\perp E[c[[\ ]/x]] \ ,$$

d'où $c[t/x] \in (\!|C|\!)^{\perp\!\!\perp\perp\!\!\perp}$.

On peut donc interpréter la règle $(\sqcup\,\mathrm{E})$ avec les biorthogonaux si on utilise des contextes de la forme $c[[\ ]/x]$ avec $c \in \Lambda(\Sigma)$. Rappelons que les biorthogonaux présentés à la section 9.3.5 utilisent des contextes d'élimination qui peuvent être vus comme des contextes d'évaluation en « appel par nom ». L'adéquation de cette interprétation repose sur le fait que grâce aux lemmes de standardisation faible prouvés à la section 9.1, l'expansion faible de tête fortement normalisante dans ces contextes d'élimination préserve la normalisation forte.

D'autre part, nous avons vu à la remarque 10.3.7 que la règle $(\sqcup\,\mathrm{E})$ suggère d'utiliser une forme d'appel par valeur : dans le terme $c[t/x]$, il faudrait normaliser t avant de le substituer. Intuitivement, cela correspond au fait que l'on a besoin de contextes d'élimination de la forme $c[[\ ]/x]$ pour prendre en compte la règle $(\sqcup\,\mathrm{E})$. Or, vis-à-vis de la normalisation forte, un tel contexte $c[[\ ]/x]$ se comporte de la même manière que le contexte $(\lambda x.c)[\ ]$. Il peut donc être vu comme un contexte d'évaluation en « appel par valeur ».

Définition 10.7.1 (Contextes d'évaluation) *L'ensemble* $\mathcal{E}_{(\sqcup \mathrm{E})}$ *des* contextes d'évaluation *est généré par la grammaire suivante :*

$$\mathtt{E}[\,] \in \mathcal{E}_{(\sqcup \mathrm{E})} \quad ::= \quad [\,] \quad | \quad \mathtt{E}[\,]\,\mathtt{t} \quad | \quad \mathtt{t}\,\mathtt{E}[\,] \;,$$

où $\mathtt{t} \in \Lambda(\Sigma)$.

Remarque 10.7.2 *Notons que les contextes d'évaluation ne sont pas des contextes d'élimination au sens de la définition 9.3.13 car ils ne sont pas stables par* β*-réduction.*

D'autre part, vis-à-vis de la réductibilité, il aurait été équivalent de prendre des contextes de la forme $\mathtt{t}[[\,]/x]$*, où* $\mathtt{t} \in \Lambda(\Sigma)$*. Notons que de tels contextes n'auraient pas non plus été des contextes d'élimination car si* $\mathtt{u}$ *est neutre dans ces contextes, alors rien n'assure que* $\mathtt{t}[\mathtt{u}/x]$ *soit aussi neutre dans ces contextes.*

Nous avons choisi d'utiliser les contextes de la définition 10.7.1 car ils correspondent à l'intuition selon laquelle la règle $(\sqcup\,\mathrm{E})$ *a besoin d'une forme « d'appel par valeur ».*

Nous utilisons ces contextes d'évaluation dans l'interprétation suivante.

Définition 10.7.3 *On pose* $\mathtt{t} \perp\!\!\!\perp \mathtt{E}[\,]$ *si et seulement si* $\mathtt{E}[\mathtt{t}] \in \mathcal{SN}_{\beta\mathcal{R}}$*. De plus, on définit l'interprétation* $(\!|\_|\!)$ *par induction sur* $\mathtt{T} \in \mathcal{T}_{\sqcap\sqcup}$ *de la manière suivante :*

$$\begin{array}{lll}
(\!|\mathtt{o}|\!) & =_{def} & \mathcal{SN}_{\beta\mathcal{R}} \;, \\
(\!|\mathtt{U} \Rightarrow \mathtt{T}|\!) & =_{def} & (\!|\mathtt{U}|\!) \Rightarrow (\!|\mathtt{T}|\!) \;, \\
(\!|\mathtt{T} \sqcap \mathtt{U}|\!) & =_{def} & ((\!|\mathtt{T}|\!)^{\perp\!\!\!\perp} \cup (\!|\mathtt{U}|\!)^{\perp\!\!\!\perp})^{\perp\!\!\!\perp} \;, \\
(\!|\mathtt{T} \sqcup \mathtt{U}|\!) & =_{def} & ((\!|\mathtt{T}|\!)^{\perp\!\!\!\perp} \cap (\!|\mathtt{U}|\!)^{\perp\!\!\!\perp})^{\perp\!\!\!\perp} \;.
\end{array}$$

Rappelons que $\mathcal{P}^\star(\mathcal{SN}_{\beta\mathcal{R}})^{\perp\!\!\!\perp\perp\!\!\!\perp}$ désigne l'ensemble des $X^{\perp\!\!\!\perp\perp\!\!\!\perp}$ pour $\emptyset \neq X \subseteq \mathcal{SN}_{\beta\mathcal{R}}$.

La difficulté maintenant est de voir comment obtenir

$$\forall \mathtt{T} \in \mathcal{T}_{\sqcap\sqcup}. \quad (\!|\mathtt{T}|\!) \in \mathcal{CR}_{\sqcap\sqcup\mathcal{R}} \;. \tag{10.8}$$

Cependant, nous avons vu que les biorthogonaux basés sur les contextes d'évaluation $\mathcal{E}_{(\sqcup \mathrm{E})}$ sont adéquats pour la règle $(\sqcup\,\mathrm{E})$. Donc comme il existe des systèmes de réécriture simples $\mathcal{R}$ tels que la typabilité dans $\boldsymbol{\lambda}_{(\sqcup \mathrm{E})\mathcal{R}}(\Sigma)$ n'implique pas la normalisation forte, on ne pourra pas avoir (10.8) pour ces systèmes. L'intérêt des biorthogonaux sur $\mathcal{E}_{(\sqcup \mathrm{E})}$ est qu'ils forment une interprétation adéquate pour $\boldsymbol{\lambda}_{(\sqcup \mathrm{E})\mathcal{R}}(\Sigma)$ *exactement* lorsque les termes typables de ce système sont fortement $\beta\mathcal{R}$-normalisants. Ainsi nous montrons, pour tout système $\mathcal{R}$ de réécriture simple sur $\Lambda(\Sigma)$, que les trois points suivants sont équivalents :

(i) Si $\Gamma \vdash_{(\sqcup \mathrm{E})\mathcal{R}} \mathtt{t} : \mathtt{T}$ alors $\mathtt{t} \in \mathcal{SN}_{\beta\mathcal{R}}$.

(ii) Pour tout $\nu \in \Lambda(\Sigma)$ et tout $f \in \mathcal{F}$, si $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$ et $\nu[r[\vec{t}/\vec{x}]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $f(\vec{t}) \mapsto_{\mathcal{R}} r$, alors $\nu[f(\vec{t})/y] \in \mathcal{SN}_{\beta\mathcal{R}}$.

(iii) L'interprétation $(\!|\_|\!)$ est adéquate pour $\boldsymbol{\lambda}_{(\sqcup \mathrm{E})\mathcal{R}}(\Sigma)$:

$$\big(\Gamma \vdash_{(\sqcup \mathrm{E})\mathcal{R}} \mathtt{t} : \mathtt{T} \quad \wedge \quad \sigma \vDash_{(\!|\_|\!)} \Gamma\big) \quad \Longrightarrow \quad \mathtt{t}\sigma \in (\!|\mathtt{T}|\!) \;.$$

Remarque 10.7.4 *La propriété (ii) dit que si tous les réduits de tête de* $f(\vec{t})$ *peuvent être placés de manière « sûre » dans le « contexte »* ν*, alors le terme* $f(\vec{t})$ *peut lui-même être placé dans de manière « sûre » dans* ν*. Cela veut dire que si chaque réduit de tête de* $f(\vec{t})$ *interagit bien avec des copies de lui-même dans* ν*, alors les copies* des différents réduits *de tête de* $f(\vec{t})$ *interagissent bien* entre elles *dans* ν*.*

Cette propriété correspond à celle que nous avons identifiée à la remarque 10.3.7.

Le cas (iii) $\Rightarrow$ (i) provient du fait que $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq (\!|T|\!) \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $T \in \mathcal{T}_{\sqcap\sqcup}$.

Lemme 10.7.5 *Pour tout* $T \in \mathcal{T}_{\sqcap\sqcup}$*, on a* $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq (\!|T|\!) \subseteq \mathcal{SN}_{\beta\mathcal{R}}$*.*

Preuve. Par induction sur T.

- $T = o$. Car $(\!|T|\!) = \mathcal{SN}_{\beta\mathcal{R}}$.
- $T = T_2 \Rightarrow T_1$. Soit $t \in \mathcal{HN}_{\sqcap\sqcup\mathcal{R}}$ Comme $(\!|T_2|\!) \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ par hypothèse d'induction, pour tout $u \in (\!|T_2|\!)$ on a $tu \in \mathcal{HN}_{\sqcap\sqcup\mathcal{R}}$, donc $tu \in (\!|T_1|\!)$ par hypothèse d'induction. Il s'en suit que $t \in (\!|T_2|\!) \Rightarrow (\!|T_1|\!)$.
  Si $t \in (\!|T_2|\!) \Rightarrow (\!|T_1|\!)$, alors comme $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq (\!|T_2|\!)$ par hypothèse d'induction, pour tout $x \in \mathcal{X}$ on a $tx \in (\!|T_1|\!)$, donc $tx \in \mathcal{SN}_{\beta\mathcal{R}}$ par hypothèse d'induction. Il s'en suit que $t \in \mathcal{SN}_{\beta\mathcal{R}}$.
- $T = T_1 \sqcap T_2$. On a $(\!|T|\!) = (\!|T_1|\!) \cap (\!|T_2|\!)$, donc $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq (\!|T|\!) \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ par hypothèse d'induction.
- $T = T_1 \sqcup T_2$. On a $(\!|T|\!) = ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp\perp\!\!\perp}$, donc $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq (\!|T|\!)$ par hypothèse d'induction. Comme $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq (\!|T_1|\!), (\!|T_2|\!)$, on a $\emptyset \neq ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp} \subseteq \mathcal{SN}_{\beta\mathcal{R}}$ par hypothèse d'induction, d'où $((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp\perp\!\!\perp} \subseteq \mathcal{SN}_{\beta\mathcal{R}}$. □

En ce qui concerne l'implication (ii) $\Rightarrow$ (iii), on commence par vérifier que pour tout système de réécriture $\mathcal{R}$ simple satisfaisant (ii), on a $(\!|T|\!) \in \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ pour tout $T \in \mathcal{T}_{\sqcap\sqcup}$.

Pour ce faire, on utilise un résultat intermédiaire, qui se montre aisément en utilisant l'adéquation et la complétude du typage dans $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$, étudiées à la section 10.2. En l'absence de réécriture, ce résultat peut être montré de manière directe, sans passer par un système de types[4].

Lemme 10.7.6 *Supposons que* $(\lambda x.t)u \in \mathcal{SN}_{\beta\mathcal{R}}$ *et* $\nu[t[u/x]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$*. Alors on a* $\nu[(\lambda x.t)u/y] \in \mathcal{SN}_{\beta\mathcal{R}}$*.*

Preuve. Comme $(\lambda x.t)u \in \mathcal{SN}_{\beta\mathcal{R}}$ on aussi $u \in \mathcal{SN}_{\beta\mathcal{R}}$ et $t[u/x] \in \mathcal{SN}_{\beta\mathcal{R}}$. Par le théorème 10.2.12 il existe des contextes Γ_1 et Γ_2 et des types T et U tels que $\Gamma_1 \vdash_{\sqcap\sqcup\mathcal{R}} u : U$ et $\Gamma_2 \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T$. D'autre part, toujours d'après le théorème 10.2.12, comme $\nu[t[u/x]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ il existe Γ_3 et V tels que $\Gamma_3 \vdash_{\sqcap\sqcup\mathcal{R}} \nu[t[u/x]/y] : V$.

On en déduit qu'avec $\Gamma =_{\text{def}} \Gamma_1 \sqcap \Gamma_2 \sqcap \Gamma_3$, on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} (\lambda y.\nu)(t[u/x]) : V$ par le lemme 10.2.10. D'après la proposition 10.2.7.(ii), il existe un type T' tel que

$$\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \lambda y.\nu : T' \Rightarrow V \qquad \text{et} \qquad \Gamma \vdash_{\sqcap\sqcup\mathcal{R}} t[u/x] : T' \,.$$

[4] Communication personnelle de Makoto Tatsuta, 2007.

Comme $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} u : U$, en appliquant le lemme 10.2.10 on a $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} (\lambda x.t)u : T'$. Il s'en suit que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} (\lambda y.v)((\lambda x.t)u) : V$. On a alors $(\lambda y.v)((\lambda x.t)u) \in \mathcal{SN}_{\beta\mathcal{R}}$ par le théorème 10.2.6, d'où $v[(\lambda x.t)u/y] \in \mathcal{SN}_{\beta\mathcal{R}}$. □

Remarque 10.7.7 *Notons que dans le lemme 10.7.6, la substitution sans capture dans v est essentielle. La propriété n'est en effet pas vérifiée si on remplace v par un contexte $C[\ ]$ pouvant capturer des variables. On trouve dans [RS95] un exemple de ce fait. Rappelons que les contextes sur $\Lambda(\Sigma)$ sont définis en 1.2.18.*

Avec

$$C[\ ] =_{def} (v \in \Lambda(\Sigma) \mapsto (\lambda y.v)\omega) \qquad t =_{def} z \qquad et \qquad u =_{def} yy\ ,$$

on a $C[t[u/x]] = (\lambda y.z)\omega \in \mathcal{SN}_\beta$ *et*

$$C[(\lambda x.t)u] \quad = \quad (\lambda y.(\lambda x.z)(yy))\omega \quad \to_\beta \quad (\lambda x.z)(\omega\omega) \notin \mathcal{SN}_\beta\ .$$

On montre maintenant que les types sont interprétés par des candidats de réductibilité lorsque $\mathcal{R}$ satisfait (ii).

Lemme 10.7.8 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$ tel que pour tout $v \in \Lambda(\Sigma)$ et tout $f \in \mathcal{F}$, si $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$ et $v[r[\vec{t}/\vec{x}]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $f(\vec{t}) \mapsto_{\mathcal{R}} r$, alors $v[f(\vec{t})/y] \in \mathcal{SN}_{\beta\mathcal{R}}$.*

Alors on a $(\!|T|\!) \in \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ pour tout $T \in \mathcal{T}_{\sqcap\sqcup}$.

Preuve. On montre que $A^{\perp\perp} \in \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ pour tout A tel que $\mathcal{HN}_{\sqcap\sqcup\mathcal{R}} \subseteq A^{\perp\perp} \subseteq \mathcal{SN}_{\beta\mathcal{R}}$. Le résultat pour $T \in \mathcal{T}_{\sqcap\sqcup}$ suit alors du lemme 10.7.5.

On vérifie que $A^{\perp\perp}$ satisfait les clauses ($\mathcal{CR}0$) et ($\mathcal{CR}1$).

($\mathcal{CR}0$). Soit $t \in A^{\perp\perp}$ tel que $t \to_{\beta\mathcal{R}} u$. Alors pour tout $E[\ ] \in A^{\perp}$, on a $E[t] \to_{\beta\mathcal{R}} E[u]$, donc $E[u] \in \mathcal{SN}_{\beta\mathcal{R}}$ car $E[t] \in \mathcal{SN}_{\beta\mathcal{R}}$. Il s'en suit que $u \in A^{\perp\perp}$.

($\mathcal{CR}1$). Soit $t \in \mathcal{N}_{\sqcap\sqcup\mathcal{R}}$ tel que $(t)_{\beta\mathcal{R}} \subseteq A^{\perp\perp}$.

Si $t \in \mathcal{HN}_{\beta\mathcal{R}}$, alors $t \in A^{\perp\perp}$ par hypothèse. Sinon, pour tout $E[\ ] \in A^{\perp}$ on doit montrer que $E[t] \in \mathcal{SN}_{\beta\mathcal{R}}$. On a $t = h\vec{v}$ et il y a deux cas :

— $h = (\lambda x.u)v$. Par hypothèse $E[u[v/x]\vec{v}] \in \mathcal{SN}_{\beta\mathcal{R}}$, d'où $E[(\lambda x.u)v\vec{v}] \in \mathcal{SN}_{\beta\mathcal{R}}$ d'après le lemme 10.7.6.

— $h = f(\vec{f})$ avec $f \in \mathcal{F}$. Dans ce cas, on a $E[f(\vec{t})\vec{v}] \in \mathcal{SN}_{\beta\mathcal{R}}$ par hypothèse. □

On peut maintenant montrer que (ii) implique (iii).

Lemme 10.7.9 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$ tel que pour tout $v \in \Lambda(\Sigma)$ et tout $f \in \mathcal{F}$, si $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$ et $v[r[\vec{t}/\vec{x}]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $f(\vec{t}) \mapsto_{\mathcal{R}} r$, alors $v[f(\vec{t})/y] \in \mathcal{SN}_{\beta\mathcal{R}}$.*

Si $\Gamma \vdash_{(\sqcup E)\mathcal{R}} t : T$ et $\sigma \models_{(\!|\_|\!)} \Gamma$ alors $t\sigma \in (\!|T|\!)$.

Preuve. Tout d'abord, par le lemme 10.7.8 on a $(\!|T|\!) \in \mathcal{CR}_{\sqcap\sqcup\mathcal{R}}$ pour tout $T \in \mathcal{T}_{\sqcap\sqcup}$.

On raisonne par induction sur $\Gamma \vdash_{(\sqcup E)\mathcal{R}} t : T$, comme au théorème 10.2.6. La seule différence est la cas de la règle $(\sqcup E)$.

$$(\sqcup E)\ \frac{\Gamma \vdash t : T_1 \sqcup T_2 \qquad \Gamma, x : T_1 \vdash c : C \qquad \Gamma, x : T_2 \vdash c : C}{\Gamma \vdash c[t/x] : C}$$

Soit $\sigma \models_{(\!|\_|\!)} \Gamma$. Par hypothèse d'induction on a $t\sigma \in ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp\perp\!\!\perp}$, et de plus pour tout $u \in (\!|T_1|\!) \cup (\!|T_2|\!)$ et tout $E[\ ] \in (\!|C|\!)^{\perp\!\!\perp}$, on a $E[c(\sigma[u/x])] \in \mathcal{SN}_{\beta\mathcal{R}}$.

D'autre part, par le lemme 1.2.10, on peut supposer que $x \notin FV(\sigma) \cup \mathcal{D}om(\sigma)$. On a donc $c(\sigma[u/x]) = (c\sigma)[u/x]$ par la proposition 1.3.5. Il s'en suit que $E[(c\sigma)[[\ ]/x]] \in ((\!|T_1|\!) \cup (\!|T_2|\!))^{\perp\!\!\perp}$, donc que $E[(c\sigma)[t\sigma/x]] \in \mathcal{SN}_{\beta\mathcal{R}}$.

De plus, $(c[t/x])\sigma = (c\sigma)(\sigma \circ [t/x])$ par le lemme 1.3.9.(ii), soit $(c[t/x])\sigma = (c\sigma)[t\sigma/x]$. Il s'en suit que $E[c[t/x]\sigma] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $E[\ ] \in (\!|C|\!)^{\perp\!\!\perp}$, d'où $c[t/x]\sigma \in (\!|C|\!)$. □

Il reste à vérifier que (i) implique (ii). Pour cela, on utilise la proposition suivante, qui se montre par induction sur $n \in \mathbb{N}$.

Proposition 10.7.10 *Pour tout $n \geq 0$, la règle suivante est dérivable dans $\boldsymbol{\lambda}_{(\sqcup E)\mathcal{R}}(\mathcal{S})$:*

$$\frac{\Gamma \vdash_{(\sqcup E)\mathcal{R}} t : \bigsqcap_{i\in\{1,\dots,n\}}(U_i \Rightarrow T)}{\Gamma \vdash_{(\sqcup E)\mathcal{R}} \lambda x.tx : (\bigsqcup_{i\in\{1,\dots,n\}} U_i) \Rightarrow T}\ (x \notin FV(t))$$

Lemme 10.7.11 *Supposons que les termes typables dans $\boldsymbol{\lambda}_{(\sqcup E)\mathcal{R}}(\Sigma)$ soient fortement $\beta\mathcal{R}$-normalisants. Pour tout $v \in \Lambda(\Sigma)$ et tout $f \in \mathcal{F}$, si $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$ et $v[r[\vec{t}/\vec{x}]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $f(\vec{t}) \mapsto_{\mathcal{R}} r$, alors $v[f(\vec{t})/y] \in \mathcal{SN}_{\beta\mathcal{R}}$.*

Preuve. Soit $f \in \mathcal{F}$ et $\vec{t}$ tels que $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$ et $v[r[\vec{t}/\vec{x}]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $f(\vec{t}) \mapsto_{\mathcal{R}} r$. On raisonne comme au lemme 10.7.6, en utilisant le théorème 10.2.12 et le lemme 10.2.10 : il existe un contexte Γ, et des types V et $(U_r)_{r\in\mathcal{R}(f)}$ tels que $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} f(\vec{t}) : \bigsqcup_{r\in\mathcal{R}(f)} U_r$, et $\Gamma \vdash_{\sqcap\sqcup\mathcal{R}} \lambda y.v : U_r \Rightarrow V$ pour tout $r \in \mathcal{R}(f)$.

Comme $\mathcal{R}(f)$ est fini, par la proposition 10.7.10, on a $\Gamma \vdash_{(\sqcup E)\mathcal{R}} (\lambda x.(\lambda y.v)x)f(\vec{t}) : V$, donc $v[f(\vec{t})/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ par hypothèse. □

On en déduit le résultat principal de cette section : les biorthogonaux sur $\mathcal{E}_{(\sqcup E)}$ sont complets vis-à-vis de la normalisation forte dans $\boldsymbol{\lambda}_{(\sqcup E)\mathcal{R}}(\Sigma)$.

Théorème 10.7.12 *Soit $\mathcal{R}$ un système de réécriture simple sur $\Lambda(\Sigma)$. Les trois points suivants sont équivalents :*

(i) Si $\Gamma \vdash_{(\sqcup E)\mathcal{R}} t : T$ alors $t \in \mathcal{SN}_{\beta\mathcal{R}}$.

(ii) Pour tout $v \in \Lambda(\Sigma)$ et tout $f \in \mathcal{F}$, si $f(\vec{t}) \in \mathcal{SN}_{\beta\mathcal{R}}$ et $v[r[\vec{t}/\vec{x}]/y] \in \mathcal{SN}_{\beta\mathcal{R}}$ pour tout $f(\vec{t}) \mapsto_{\mathcal{R}} r$, alors $v[f(\vec{t})/y] \in \mathcal{SN}_{\beta\mathcal{R}}$.

(iii) L'interprétation $(\!|\_|\!)$ est adéquate pour $\boldsymbol{\lambda}_{(\sqcup E)\mathcal{R}}(\Sigma)$:

$$\big(\Gamma \vdash_{(\sqcup E)\mathcal{R}} t : T \quad\wedge\quad \sigma \models_{(\!|\_|\!)} \Gamma\big) \quad\Longrightarrow\quad t\sigma \in (\!|T|\!)\ .$$

Preuve. On a (i) ⇒ (ii) par le lemme 10.7.11, (ii) ⇒ (iii) par le lemme 10.7.9 et (iii) ⇒ (i) par le lemme 10.7.5. □

Chapitre 11

Types contraints pour la normalisation forte

Dans ce chapitre, nous présentons un critère de normalisation forte pour la réécriture conditionnelle combinée au λ-calcul. Ce critère repose sur les types inductifs et utilise un système de types contraints. C'est à notre connaissance la seule méthode générale qui permette, avec la réécriture conditionnelle, de prendre en compte dans l'argument de terminaison des informations issues de la satisfaction des conditions des règles de réécriture.

Nous commençons à la section 11.1 par introduire le calcul sur lequel nous travaillons. Celui-ci consiste en la combinaison de la réécriture conditionnelle normale au λ-calcul simplement typé avec produits. Nous utilisons de plus les booléens non curryfiés, tels que présentés à l'exemple 3.1.3, ainsi que la construction $\mathsf{let}\ x = t\ \mathsf{in}\ u$.

Ensuite, nous présentons à la section 11.2 quelques notions importantes sur les types inductifs. Cela nous permet de passer rapidement en revue deux critères de terminaison pour la réécriture dans le λ-calcul basés sur les types inductifs : le schéma général à la section 11.2.3.(a), et les types annotés à la section 11.2.3.(b). Par rapport au schéma général, les types annotés permettent de formuler un critère plus fin et moins sensible à la syntaxe. En particulier, ils permettent de montrer la normalisation forte de systèmes de réécriture non simplement terminants (voir par exemple [Ohl02] pour des précisions sur cette notion).

Les types contraints, que nous présentons à la section 11.3, sont un enrichissement des types annotés : les annotations de types peuvent être gardées par des contraintes de l'arithmétique de Presburger avec booléens, et les variables de taille peuvent être quantifiées universellement ou existentiellement. Par exemple, on peut exprimer le fait que la fonction pivot définie à l'exemple 9.1.32 renvoie une paire de listes dont la somme des tailles est égale à la taille de la liste qui lui est passée en argument. Cela permet de montrer que certaines implantations du tri QuickSort utilisant pivot préservent la taille de la liste à trier. De plus, en utilisant la réécriture conditionnelle, les types contraints permettent de formuler la terminaison de la fonction 91 de McCarthy, ce qui ne peut être fait, dans le cadre de la réécriture, avec les types annotés de [Bla04].

Intuitivement, notre système de types contraints peut être vu comme une version implicite du système de types dépendants pour ML développé par Xi [Xi98]. Pour la vérification du typage, au lieu d'utiliser un mécanisme d'élaboration comme le fait Xi, nous définissons à la section 11.3.3 un algorithme de vérification du typage qui fonctionne

par génération de contraintes.

Nous passons ensuite aux problèmes de normalisation forte. Les contraintes existentielles sont interprétées par des unions d'ensembles de réductibilité. Or nous avons vu au chapitre 10 qu'il existe des systèmes de réécriture confluents qui n'admettent pas de famille de réductibilité stable par union. Nous traitons cette question à la section 11.4.1, et nous adoptons la solution de limiter la forme du type de sortie des symboles définis par des règles de réécriture. Nous définissons une interprétation qui mélange les ensembles saturés et les candidats de réductibilité, et nous montrons à la section 11.4.2 qu'elle est adéquate pour le système de types contraint lorsque les symboles appartiennent à l'interprétation de leur type.

D'autre part, la sémantique naturelle pour nos types contraints est d'utiliser des interprétations « singletons » pour les types inductifs. C'est aussi ce qui est fait dans [Xi02] pour le langage ML. Cependant, avec la réécriture, la définition d'une interprétation singleton est plus délicate. Nous présentons notre solution à la section 11.4.3.

Enfin, la section 11.4.4 est dévolue à la définition de notre critère de terminaison et à la preuve de sa correction. De plus, nous détaillons son application à la fonction pivot et à la fonction 91 de McCarthy.

Les résultats présentés dans ce chapitre ont été publiés dans [BR06].

11.1 Réécriture conditionnelle avec produits et booléens

Nous commençons par introduire le calcul sur lequel nous travaillons. Celui-ci consiste en la combinaison de réécriture conditionnelle normale au λ-calcul simplement typé avec produits. Nous utilisons de plus les booléens non curryfiés, tels que présentés à l'exemple 3.1.3, ainsi que la construction $\mathsf{let}\ x = t\ \mathsf{in}\ u$.

Rappelons que les booléens présentés à l'exemple 3.1.3 sont construits par les symboles true et false. De plus, ils sont éliminés par le symbole non curryfié $\mathsf{ite}(\_,\_,\_)$ défini par les règles :

$$\mathsf{ite}(\mathsf{true}, x, y) \quad \mapsto_{\mathsf{ite}} \quad x \qquad \text{et} \qquad \mathsf{ite}(\mathsf{false}, x, y) \quad \mapsto_{\mathsf{ite}} \quad y\ .$$

Enfin, Σ_{Bool} est la signature $\{\mathsf{true}, \mathsf{false}, \mathsf{ite}(\_,\_,\_)\}$, et si $\mathcal{B}_0$ est un ensemble de types de base, alors on pose $\mathcal{B}_{0\mathsf{Bool}} =_{\text{def}} \mathcal{B}_0 \uplus \{\mathsf{Bool}\}$.

On considère une signature curryfiée Σ_0, et un ensemble $\mathcal{R}$ de règles de réécriture conditionnelles sur $\Sigma_0 \uplus \Sigma_{\mathsf{Bool}}$ de la forme

$$d_1 = c_1 \ \wedge\ \ldots\ \wedge\ d_n = c_n \ \supset\ f\, l_1 \ldots l_k \ \mapsto_{\mathcal{R}}\ r$$

où $f \in \Sigma_0$ et $c_i \in \{\mathsf{true}, \mathsf{false}\}$ pour tout $i \in \{1, \ldots, n\}$. Rappelons que les notions de règle de réécriture conditionnelle et de relation de réécriture conditionnelle sont définies à la section 4.1. On considère la réécriture conditionnelle normale (voir aussi la remarque 11.1.3).

Soit $\mathcal{B}_0$ un ensemble de types de base. On suppose donné τ tel que la signature (Σ_0, τ) soit typée dans $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ et basée sur les produits, au sens de la définition 9.1.31. Le

type des symboles $f \in \Sigma_0$ est donc de la forme

$$\tau(f) \quad = \quad T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow H_f \ ,$$

où n est l'arité applicative de f, notée a_f, et où H_f est un type sur la grammaire

$$T, U \in \mathcal{T}_\times(\mathcal{B}_{0\mathsf{Bool}}) \quad ::= \quad B \ \mid \ T \times U \ ,$$

avec $B \in \mathcal{B}_{0\mathsf{Bool}}$.

Enfin, on ajoute aux termes la construction $\mathsf{let}\ x = t\ \mathsf{in}\ u$ dans laquelle la variable x est liée dans u. Elle est éliminée par la règle de réduction suivante :

$$\mathsf{let}\ x = t\ \mathsf{in}\ u \quad \mapsto_{\mathsf{let}} \quad u[t/x] \ .$$

Le terme $\mathsf{let}\ x = t\ \mathsf{in}\ u$ peut être vu comme du sucre syntaxique pour $(\lambda x.u)t$, mais dans le système que nous allons présenter il est utile de le considérer comme une construction distincte. Nous ne détaillons pas la relation d'α-conversion qui lui correspond.

Définition 11.1.1 *Soit un ensemble de types de bases $\mathcal{B}_0$ et (Σ_0, τ) une signature basée sur les produits dans $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}})$.*

(i) On désigne par $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \tau)$ le système dont

— *les termes sont ceux générés par la grammaire suivante :*

$$\begin{array}{rcl} t, u, v \in \Lambda_{\mathcal{S}}(\Sigma_0) & ::= & x \mid f \\ & \mid & \lambda x.t \mid t\,u \\ & \mid & (t,u) \mid \pi_1\,t \mid \pi_2\,t \\ & \mid & \mathsf{true} \mid \mathsf{false} \mid \mathsf{ite}(t,u,v) \\ & \mid & \mathsf{let}\ x = t\ \mathsf{in}\ u \ , \end{array}$$

où $f \in \Sigma_0$ et $x \in \mathcal{X}$,

— *les types sont les éléments de $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}})$,*

— *la relation de typage, notée $\Gamma \vdash_{\underline{\mathcal{S}}} t : T$, est celle de $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}}, \Sigma_0, \tau)$ à laquelle on ajoute les règles suivantes :*

$$(\mathsf{let})\ \frac{\Gamma \vdash t : T \qquad \Gamma, x : T \vdash u : U}{\Gamma \vdash \mathsf{let}\ x = t\ \mathsf{in}\ u : U}$$

$$(\mathsf{Bool})\ \frac{\Gamma \vdash b : \mathsf{Bool} \qquad \Gamma \vdash t_1 : T \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash \mathsf{ite}(b, t_1, t_2) : T}\ (T \in \mathcal{T}_\times(\mathcal{B}_{0\mathsf{Bool}}))$$

— *la relation de réduction est*

$$\mapsto_{\mathcal{S}} \quad =_{def} \quad \mapsto_\beta \ \cup \ \mapsto_\pi \ \cup \ \mapsto_{\mathsf{ite}} \ \cup \ \mapsto_{\mathsf{let}} \ .$$

(ii) Soit $\mathcal{R}$ un système de réécriture conditionnelle sur $\Lambda_{\mathcal{S}}(\Sigma_0)$ dont les règles sont de la forme

$$d_1 = c_1 \ \wedge \ \ldots \ \wedge \ d_n = c_n \ \supset \ f\ l_1 \ldots l_k \ \mapsto_{\mathcal{R}} \ r$$

où $f \in \Sigma_0$ *avec*

$$\tau(f) \quad = \quad T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow H \ ,$$

et $c_i \in \{\mathsf{true}, \mathsf{false}\}$ *pour tout* $i \in \{1, \ldots, n\}$. *Alors* $\mathcal{R}$ *est* typable dans $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \tau)$ *s'il existe un contexte* Γ *tel que*

$$\Gamma \vdash_{\underline{\mathcal{S}}} l_1 : T_1 \quad \ldots \quad \Gamma \vdash_{\underline{\mathcal{S}}} l_k : T_k \ , \qquad \Gamma \vdash_{\underline{\mathcal{S}}} r : H \qquad \textit{et} \qquad \Gamma \vdash_{\underline{\mathcal{S}}} \vec{d} : \mathsf{Bool} \ .$$

Si $\mathcal{R}$ *est typable dans* $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \tau)$ *alors on définit la relation de réécriture* $\to_{\mathcal{SR}}$ *comme étant l'union de* $\to_{\mathcal{S}}$ *et de la relation* $\to_{\mathcal{R}}$ *de réécriture* $\mathcal{S}$*-conditionnelle par joignabilité issue de* $\mathcal{R}$.

(iii) Étant donné un système de réécriture conditionnelle $\mathcal{R}$ *typable dans* $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \tau)$, *on désigne par* $\boldsymbol{\lambda}_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \tau)$ *le système identique à* $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \tau)$, *sauf pour la relation de réduction qui est* $\to_{\mathcal{SR}}$.

Rappelons que les relations de réécriture conditionnelle sont définies en 4.1.5.

Remarque 11.1.2 *La règle de typage* (Bool) *correspond, au sens de la définition 3.6.1, au typage suivant pour le symbole* $\mathsf{ite}(\_, \_, \_)$ *:*

$$\tau(\mathsf{ite}) \quad = \quad T \in \mathcal{T}_\times(\mathcal{B}_{0\mathsf{Bool}}) \ \mapsto \ (\mathsf{Bool}, T, T, T) \ .$$

Remarque 11.1.3 *La forme des règles de réécriture* $\mathcal{R}$ *impose que* true *et* false *sont en forme* $\mathcal{SR}$*-normale. La relation de réécriture* $\mathcal{SR}$ *est donc une relation de réécriture conditionnelle normale au sens de la définition 4.1.8.*

De plus, nous supposerons aux sections 11.4.3 et 11.4.4 la confluence de $\to_{\mathcal{SR}}$. *Cette propriété nous sera utile pour la correction de notre interprétation « singleton » des types inductifs (définie à la section 11.4.3).*

Exemple 11.1.4 *On considère des entiers curryfiés tels que présentés à l'exemple 3.1.13. Il sont donc construits avec les symboles* 0 *et* S *typés de la manière suivante :*

$$\tau_{\mathsf{Nat}}(0) \ = \ \mathsf{Nat} \qquad \textit{et} \qquad \tau_{\mathsf{Nat}}(S) \ = \ \mathsf{Nat} \Rightarrow \mathsf{Nat} \ .$$

Présentons quelques fonctions sur les entiers. On commence par l'addition entière, qui peut être définie par les règles suivantes :

$$\begin{array}{lll} \mathsf{plus}\ x\ 0 & \mapsto_{\mathsf{plus}} & x \\ \mathsf{plus}\ x\ (S\ y) & \mapsto_{\mathsf{plus}} & \mathsf{plus}\ (S\ x)\ y \ . \end{array}$$

Le système de réécriture suivant définit la version curryfiée de la soustraction entière présentée à l'exemple 3.1.11 :

$$\begin{array}{lll} \mathsf{minus}\ x\ 0 & \mapsto_{\mathsf{minus}} & x \\ \mathsf{minus}\ 0\ y & \mapsto_{\mathsf{minus}} & 0 \\ \mathsf{minus}\ (S\ x)\ (S\ y) & \mapsto_{\mathsf{minus}} & \mathsf{minus}\ x\ y \ . \end{array}$$

Les symboles plus *et* minus *ont tous deux pour type* Nat ⇒ Nat ⇒ Nat. *Enfin, voici une version curryfiée de la fonction « strictement supérieur à » présentée à l'exemple 3.1.11 :*

$$\begin{array}{lll} > \ 0\ y & \mapsto_{>} & \mathsf{false}\ , \\ > \ (\mathsf{S}\ x)\ 0 & \mapsto_{>} & \mathsf{true}\ , \\ > \ (\mathsf{S}\ x)\ (\mathsf{S}\ y) & \mapsto_{>} & > x\ y\ . \end{array}$$

Le symbole > *a pour type* Nat ⇒ Nat ⇒ Bool.

Exemple 11.1.5 *Le système de réécriture suivant définit la division entière :*

$$\begin{array}{lll} \mathsf{div}\ 0\ y & \mapsto_{\mathsf{div}} & 0 \\ \mathsf{div}\ (\mathsf{S}\ x)\ (\mathsf{S}\ y) & \mapsto_{\mathsf{div}} & \mathsf{S}\ (\mathsf{div}\ (\mathsf{minus}\ x\ (\mathsf{S}\ y))\ (\mathsf{S}\ y))\ . \end{array}$$

Le symbole div *a pour type* Nat ⇒ Nat ⇒ Nat.

Exemple 11.1.6 *Les listes curryfiées, telles que présentées à l'exemple 3.1.13, sont définies par les constructeurs* nil *et* cons. *On considère des listes d'entiers, qui peuvent être typées de la manière suivante :*

$$\tau_{\mathsf{List}}(\mathsf{nil}) \ = \ \mathsf{List} \qquad \textit{et} \qquad \tau_{\mathsf{List}}(\mathsf{cons}) \ = \ \mathsf{Nat} \Rightarrow \mathsf{List} \Rightarrow \mathsf{List}\ .$$

Exemple 11.1.7 *Considérons le symbole* pivot *présenté à l'exemple 9.1.32, de type* Nat ⇒ List ⇒ List × List. *Il définit, par deux règles de réécriture, une fonction utilisée dans certaines implantations du tri* QuickSort. *La première règle est*

$$\mathsf{pivot}\ x\ \mathsf{nil} \quad \mapsto_{\mathsf{pivot}} \quad (\mathsf{nil}, \mathsf{nil})\ ,$$

et la seconde est

$$\begin{array}{lll} \mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) & \mapsto_{\mathsf{pivot}} & \mathsf{ite}(>\ y\ x,\ (\pi_1\ (\mathsf{pivot}\ x\ l),\ \mathsf{cons}\ y\ (\pi_2\ (\mathsf{pivot}\ x\ l))), \\ & & \quad (\mathsf{cons}\ y\ (\pi_1\ (\mathsf{pivot}\ x\ l)),\ \pi_2\ (\mathsf{pivot}\ x\ l))) \end{array} \tag{11.1}$$

On peut aussi l'écrire en utilisant un let *pour partager les occurrences de* pivot x l *dans le membre droit :*

$$\begin{array}{lll} \mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) & \mapsto_{\mathsf{pivot}} & \mathsf{let}\ z = (\mathsf{pivot}\ x\ l)\ \mathsf{in} \\ & & \mathsf{ite}(>\ y\ x,\ (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ & & \quad (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \end{array} \tag{11.2}$$

Exemple 11.1.8 *La fonction 91 de McCarthy, de type* Nat ⇒ Nat, *est définie par les règles suivantes :*

$$\begin{array}{lllllll} > \ x\ 100 & = & \mathsf{true} & \supset & \mathsf{f}\ x & \mapsto_{91} & \mathsf{minus}\ x\ 10\ , \\ > \ x\ 100 & = & \mathsf{false} & \supset & \mathsf{f}\ x & \mapsto_{91} & \mathsf{f}\ (\mathsf{f}\ (\mathsf{plus}\ x\ 11))\ , \end{array}$$

où, pour tout $n \in \mathbb{N}$, *le terme* $\mathsf{S}^n 0$ *est représenté par* n. *On a* $\mathsf{f}\ n \to_{91}^{+} \mathsf{minus}\ n\ 10$ *si* $n > 100$ *et* $\mathsf{f}\ n \to_{91}^{+} 91$ *sinon.*

11.2 Terminaison avec types inductifs

Dans cette section nous rappelons quelques notions sur les types inductifs, qui sont un des éléments centraux des critères de terminaison proposés dans [BJO02, Abe04, BR06].

Les types inductifs sont des types de données construits par induction. Ils sont très intéressants vis-à-vis de la terminaison car ils permettent de formuler le fait que les règles de réécriture calculent par récursion sur des structures de données inductives.

D'autres approches existent pour la normalisation forte de la réécriture combinée au λ-calcul. Les travaux précurseurs à ce sujet sont [JO91, JO97], citons aussi [Pol93, Pol96]. Une lignée importante de travaux concerne l'adaptation à l'ordre supérieur de l'ordre récursif sur les chemins [JR99, JR07]. Certains de ces travaux ont été adaptés au calcul des constructions [WC03a, WC03b]. Plus récemment, dans [Bla06b, BJR07], l'ordre récursif sur les chemins d'ordre supérieur a été comparé au schéma général, évoqué à la section 11.2.3.(a). Citons enfin une adaptation des paires de dépendances à la réécriture combinée au λ-calcul simplement typé [Bla06a, KS07].

Avant de présenter les types inductifs, on commence par rappeler ce que l'on a vu à la section 9.1.4 sur la normalisation forte de la réécriture dans le λ-calcul typé. Considérons un système non conditionnel $\mathcal{R}$ typé dans $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$. La condition suffisante donnée au lemme 9.1.35 pour la normalisation forte de $\rightarrow_{\beta\pi\mathcal{R}}$ sur les termes typables de $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$ est la suivante. Comme $\mathcal{R}$ est typé dans $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$, ses membres gauches sont de la forme $\mathsf{f}\,\mathsf{l}_1 \dots \mathsf{l}_k$ avec

$$\tau(\mathsf{f}) \;=\; \mathsf{T}_1 \Rightarrow \cdots \Rightarrow \mathsf{T}_k \Rightarrow \mathsf{H} \qquad \text{et} \qquad k \leq a_{\mathsf{f}} \;.$$

Nous supposons que la relation $\rightarrow_{\beta\pi\mathcal{R}}$ est finiment branchante et que l'interprétation $(\mathcal{CR}_{\beta\pi\mathcal{R}}, [\![\_]\!])$ est telle que $[\![\_ \times \_]\!] = \_ \times \_$.

D'après le lemme 9.1.35, si pour toute règle $\mathsf{f}\,\mathsf{l}_1 \dots \mathsf{l}_k \mapsto_{\mathcal{R}} \mathsf{r}$ avec

$$\tau(\mathsf{f}) \;=\; \mathsf{T}_1 \Rightarrow \cdots \Rightarrow \mathsf{T}_k \Rightarrow \mathsf{H} \;,$$

et pour toute substitution σ, on a

$$(\forall i \in \{1, \dots, k\}. \quad \mathsf{l}_i\sigma \in [\![\mathsf{T}_i]\!]) \quad \Longrightarrow \quad \mathsf{r}\sigma \in [\![\mathsf{H}]\!] \;, \tag{11.3}$$

alors $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$ est fortement $\beta\pi\mathcal{R}$-normalisant.

11.2.1 Une présentation informelle

Avant d'aborder les constructions formelles, nous essayons d'en donner une intuition en nous basant sur l'exemple du système T de Gödel présenté à la section 3.7.2.

Remarque 11.2.1 *Dans ce qui suit, nous avons volontairement laissé de côté les notions de réductibilité utilisées dans l'interprétation des types inductifs.*

Rappelons le type des constructeurs $\mathsf{0}$ et S des entiers curryfiés, tels que présentés à l'exemple 11.1.4 :

$$\tau(\mathsf{0}) \;=\; \mathsf{Nat} \qquad \text{et} \qquad \tau(\mathsf{S}) \;=\; \mathsf{Nat} \Rightarrow \mathsf{Nat} \;.$$

Ces types suggèrent d'interpréter Nat en utilisant une famille d'ensembles $(\llbracket\mathsf{Nat}\rrbracket^n)_{n\in\mathbb{N}}$ tels que chaque $\llbracket\mathsf{Nat}\rrbracket^n$ contiennent, intuitivement, « les entiers de taille au plus n », c'est-à-dire les termes de la forme $\mathsf{S}^m\,0$ pour $m \leq n$. Les $\llbracket\mathsf{Nat}\rrbracket^n$ peuvent donc être définis par induction sur n comme suit : $0 \in \llbracket\mathsf{Nat}\rrbracket^0$, et pour tout $n \geq 0$, si $t \in \llbracket\mathsf{Nat}\rrbracket^n$ alors $\mathsf{S}\,t \in \llbracket\mathsf{Nat}\rrbracket^{n+1}$. On peut alors poser $\llbracket\mathsf{Nat}\rrbracket =_{\text{def}} \bigcup_{n\in\mathsf{Nat}} \llbracket\mathsf{Nat}\rrbracket^n$.

Considérons un ensemble $\mathcal{B}_0$ de types de base qui contient Nat. À la section 3.7.2, nous avons présenté le récurseur rec par des règles de réécriture qui, pour tout $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$ peuvent être typées de la manière suivante :

$$\begin{array}{rcll} x : T, y : T \Rightarrow \mathsf{Nat} \Rightarrow T & \vdash_{\mathsf{Nat}} & \mathsf{rec}\ x\ y\ 0 & \mapsto_{\mathsf{rec}}\ x : T \\ z : \mathsf{Nat}, x : T, y : T \Rightarrow \mathsf{Nat} \Rightarrow T & \vdash_{\mathsf{Nat}} & \mathsf{rec}\ x\ y\ (\mathsf{S}\ z) & \mapsto_{\mathsf{rec}}\ y\ (\mathsf{rec}\ x\ y\ z)\ z : T \end{array}$$

Remarque 11.2.2 *Ces règles supposent que pour tout $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$, on ait $\mathsf{rec} : T \Rightarrow (T \Rightarrow \mathsf{Nat} \Rightarrow T) \Rightarrow \mathsf{Nat} \Rightarrow T$. Comme cela ne correspond pas à notre hypothèse que les symboles ont un type simple unique, nous considérons en fait une famille de symboles $(\mathsf{rec}_T)_{T\in\mathcal{T}_\Rightarrow(\mathcal{B}_0)}$, et nous désignons toujours rec_T par rec.*

Nous allons voir comment raisonner sur ces règles en utilisant une induction sur la construction de l'interprétation de Nat. Considérons un terme $\mathsf{rec}\ u\ v\ t$ avec $u \in \llbracket T\rrbracket$, $v \in \llbracket T\rrbracket \Rightarrow \llbracket\mathsf{Nat}\rrbracket \Rightarrow \llbracket T\rrbracket$ et $t \in \llbracket\mathsf{Nat}\rrbracket$. On va montrer par induction sur $n \in \mathbb{N}$ que si $t \in \llbracket\mathsf{Nat}\rrbracket^n$, alors $\mathsf{rec}\ u\ v\ t \in \llbracket T\rrbracket$. D'après (11.3), il faut vérifier que si $\mathsf{rec}\ u\ v\ t \mapsto_{\mathsf{rec}} r$, alors $r \in \llbracket T\rrbracket$. Rappelons que nous avons supposé que $\llbracket\mathsf{Nat}\rrbracket^n$ ne contient que « des entiers de taille au plus n ».

Cas de base $n = 0$. On a alors $t = 0$ et la seule règle qui peut s'appliquer à $\mathsf{rec}\ u\ v\ t$ est la première règle, et on a $u \in \llbracket T\rrbracket$ par hypothèse.

Cas d'induction. Supposons que $\mathsf{rec}\ u\ v\ t \in \llbracket T\rrbracket$, avec $t \in \llbracket\mathsf{Nat}\rrbracket^n$, et montrons que $\mathsf{rec}\ u\ v\ (\mathsf{S}\ t) \in \llbracket T\rrbracket$. La règle qui s'applique à notre terme est la second règle. Or, comme $\mathsf{rec}\ u\ v\ t \in \llbracket T\rrbracket$ par hypothèse d'induction, on a bien $v\ (\mathsf{rec}\ u\ v\ t)\ t \in \llbracket T\rrbracket$.

Nous allons maintenant donner un cadre formel à ce raisonnement.

11.2.2 Types inductifs

Un type avec constructeurs est la donnée d'un type de base C et de symboles $c_1, \ldots, c_n$, appelés constructeurs, et dont le type est de la forme

$$T_1 \Rightarrow \cdots \Rightarrow T_{m_i} \Rightarrow C \qquad \text{pour tout} \qquad i \in \{1, \ldots, n\}\ .$$

Un type inductif est un type avec constructeurs $(C, c_1, \ldots, c_n)$ tel que les occurrences de C dans le type des arguments de ses constructeurs vérifient une condition de *positivité*, identifiée par Mendler [Men87]. Cette condition permet de définir l'interprétation de C comme étant le plus petit point fixe d'une fonction monotone. Ceci permet de voir les types inductifs comme étant des types de données construits par induction.

Par simplicité on se restreint aux types inductifs construits sur $\mathcal{T}_\Rightarrow(\mathcal{B}_0)$. Rappelons que comme les types de $\mathcal{T}_\Rightarrow(\mathcal{B}_0)$ sont des termes de $\mathcal{T}\mathit{er}(\mathcal{B}_0 \cup \{\_ \Rightarrow \_\})$, les notions de positions et d'occurrences pour $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$ sont définies en 1.1.12. De plus, les positions dans les types $T \in \mathcal{T}_\Rightarrow(\mathcal{B}_0)$ sont des mots sur $\{1, 2\}$.

Définition 11.2.3 (Positions positives et négatives) *Soit* $\mathsf{T} \in \mathcal{T}_{\Rightarrow}(\mathcal{B}_0)$. *Une position* $\mathsf{p} \in \mathcal{P}\mathsf{os}(\mathsf{T})$ *est* positive *si elle contient un nombre pair de* 1 *et négative sinon.*

Exemple 11.2.4 *Dans les types suivants, les occurrences de* B *sont positives et celles de* A *sont négatives :*

$$\mathsf{B} \qquad \mathsf{A} \Rightarrow \mathsf{B} \qquad (\mathsf{B} \Rightarrow \mathsf{A}) \Rightarrow \mathsf{B}\ .$$

Définition 11.2.5 (Types inductifs) *Soit un ensemble de types de base* $\mathcal{B}_0$ *et une signature* (Σ_0, τ) *typée sur* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0)$.

(i) *Un* type avec constructeurs sur $(\mathcal{B}_0, \Sigma_0, \tau)$ *est un tuple* $\mathsf{C} = (\mathsf{C}, \mathsf{c}_1, \ldots, \mathsf{c}_n)$ *où* $\mathsf{C} \in \mathcal{B}_0$ *et* $\mathsf{c}_i \in \Sigma_0$ *avec* $\tau(\mathsf{c}_i) = \mathsf{T}_1 \Rightarrow \cdots \Rightarrow \mathsf{T}_{m_i} \Rightarrow \mathsf{C} \in \mathcal{T}_{\Rightarrow}(\mathcal{B}_0)$ *pour tout* $i \in \{1, \ldots, n\}$.

(ii) *Étant donnés* C *et* D *deux types avec constructeurs sur* $(\mathcal{B}_0, \Sigma_0, \tau)$, *on dit que* C précède D dans $(\mathcal{B}_0, \Sigma_0, \tau)$, *notation* $\mathsf{C} \leq_{(\mathcal{B}_0,\Sigma_0,\tau)} \mathsf{D}$, *si* C *apparaît dans le type d'un des constructeurs de* D.

(iii) *Un type avec constructeurs* $\mathsf{C} = (\mathsf{C}, \mathsf{c}_1, \ldots, \mathsf{c}_n)$ *sur* $(\mathcal{B}_0, \Sigma_0, \tau)$ *est un* type pré-inductif *sur* $(\mathcal{B}_0, \Sigma_0, \tau)$ *si pour tout* $i \in \{1, \ldots, n\}$, $\tau(\mathsf{c}_i) = \mathsf{T}_1 \Rightarrow \cdots \Rightarrow \mathsf{T}_{m_i} \Rightarrow \mathsf{C}$ *est tel que pour tout* $j \in \{1, \ldots, m_i\}$, *pour tout* $\mathsf{D} \geq_{(\mathcal{B}_0,\Sigma_0,\tau)} \mathsf{C}$, *toutes les occurrences de* D *dans* T_j *sont à des positions positives.*

(iv) *Un* type inductif sur $(\mathcal{B}_0, \Sigma_0, \tau)$ *est un type pré-inductif* C *sur* $(\mathcal{B}_0, \Sigma_0, \tau)$ *tel que la relation* $<_{(\mathcal{B}_0,\Sigma_0,\tau)}$ *est bien fondée sur* $\{\mathsf{D} \mid \mathsf{D} \leq_{(\mathcal{B}_0,\Sigma_0,\tau)} \mathsf{C}\}$.

Si $\mathfrak{C}$ *est une classe d'équivalence pour* $\simeq_{(\mathcal{B}_0,\Sigma_0,\tau)}$, *alors on désigne par* $\leq^{\mathfrak{C}}_{(\mathcal{B}_0,\Sigma_0,\tau)}$ *(resp. par* $<^{\mathfrak{C}}_{(\mathcal{B}_0,\Sigma_0,\tau)}$*) l'ensemble des* D *tels que* $\mathsf{D} \leq_{(\mathcal{B}_0,\Sigma_0,\tau)} \mathsf{C}$ *(resp.* $\mathsf{D} <_{(\mathcal{B}_0,\Sigma_0,\tau)} \mathsf{C}$*) pour tout* $\mathsf{C} \in \mathfrak{C}$.

Lorsque le contexte le permet, on parle de « type inductif sur $\mathcal{B}_0$ » au lieu de « type inductif sur $(\mathcal{B}_0, \Sigma_0, \tau)$ », et on désigne $\leq_{(\mathcal{B}_0,\Sigma_0,\tau)}$ par $\leq_{\mathcal{B}_0}$. On utilisera toujours C pour dénoter à la fois le type avec constructeur $\mathsf{C} = (\mathsf{C}, \mathsf{c}_1, \ldots, \mathsf{c}_n)$ et l'ensemble $\{\mathsf{c}_1, \ldots, \mathsf{c}_n\}$.

Notons que si C apparaît dans le type des arguments d'un de ses constructeurs, alors on a $\mathsf{C} \leq_{\mathcal{B}_0} \mathsf{C}$, mais que, en général, la relation $\leq_{\mathcal{B}_0}$ n'est pas nécessairement réflexive. D'autre part, dans le cas 11.2.5.(iii), si $\mathsf{D} \geq_{\mathcal{B}_0} \mathsf{C}$ apparaît dans le type d'un argument d'un des constructeurs de C, alors on a $\mathsf{C} \simeq_{\mathcal{B}_0} \mathsf{D}$.

Exemple 11.2.6 *Voici quelques types inductifs :*

(i) *le type* Bool *des booléens, avec pour constructeurs*

$$\mathsf{true} : \mathsf{Bool} \qquad \textit{et} \qquad \mathsf{false} : \mathsf{Bool}\ ,$$

(ii) *le type* Nat *entiers, avec pour constructeurs*

$$0 : \mathsf{Nat} \qquad \textit{et} \qquad \mathsf{S} : \mathsf{Nat} \Rightarrow \mathsf{Nat}\ ,$$

(iii) *le type* List *des listes d'entiers, avec pour constructeurs*

$$\mathsf{nil} : \mathsf{List} \qquad \textit{et} \qquad \mathsf{cons} : \mathsf{Nat} \Rightarrow \mathsf{List} \Rightarrow \mathsf{List}\ ,$$

(iv) le type Ord *des ordinaux de Brouwer, avec pour constructeurs*

$$0 : \mathsf{Ord} \qquad \mathsf{S} : \mathsf{Ord} \Rightarrow \mathsf{Ord} \qquad \textit{et} \qquad \mathsf{lim} : (\mathsf{Nat} \Rightarrow \mathsf{Ord}) \Rightarrow \mathsf{Ord}\ ,$$

(v) le type Cont *représentant des continuations sur les entiers, avec pour constructeurs*

$$\mathsf{D} : \mathsf{Cont} \qquad \textit{et} \qquad \mathsf{C} : ((\mathsf{Cont} \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Cont}\ .$$

Notons que l'on a $\mathsf{Nat} <_{\{\mathsf{Nat},\mathsf{List},\mathsf{Ord},\mathsf{Cont}\}} \mathsf{List}, \mathsf{Ord}, \mathsf{Cont}$.

Exemple 11.2.7 *Le type* D*, avec pour constructeur*

$$\mathsf{c} : (\mathsf{D} \Rightarrow \mathsf{U}) \Rightarrow \mathsf{D}$$

n'est pas inductif. En effet, D *a une occurrence négative dans* $\mathsf{D} \Rightarrow \mathsf{U}$.

Définition 11.2.8 *Étant donné un ensemble de types de base* $\mathcal{B}_0$*, une signature* (Σ_0, τ) *typée sur* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0)$ *et un ensemble* $\mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$ *de types inductifs sur* $(\mathcal{B}_0, \Sigma_0, \tau)$*, on pose* $\mathcal{F}(\mathcal{B}_0, \Sigma_0, \tau) =_{def} \Sigma_0 \setminus \mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$

Lorsque le contexte le permet, on désigne $\mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$ par $\mathcal{C}$ et $\mathcal{F}(\mathcal{C}_0, \Sigma_0, \tau)$ par $\mathcal{F}$. De plus, on écrit indifféremment $\mathsf{c} \in \mathcal{C}$ ou $\mathsf{c} \in \mathsf{C}$ pour dire que c est un constructeur d'un type inductif $\mathsf{C} \in \mathcal{C}$.

Avec les types inductifs, on peut considérer des systèmes de réécriture ayant une forme particulière, correspondant à l'idée que les fonctions définies par réécriture calculent par récursion sur des types inductifs.

Définition 11.2.9 (Système de réécriture avec constructeurs inductifs) *Soit un ensemble de types de base* $\mathcal{B}_0$*, une signature* (Σ_0, τ) *typée sur* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0)$ *et un ensemble* $\mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$ *de types inductifs sur* $(\mathcal{B}_0, \Sigma_0, \tau)$*. Un système de réécriture (conditionnel)* $\mathcal{R}$ *sur* $\Lambda(\Sigma_0)$ *est* un système de réécriture à constructeurs inductifs sur $\mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$ *si les membres gauches* l *de* $\mathcal{R}$ *sont des termes sur la grammaire suivante :*

$$\begin{array}{rrcl} & l & ::= & f\ p_1 \ldots p_n \\ \textit{avec} & p & ::= & x \mid c\ p_1 \ldots p_m \end{array}$$

où $x \in \mathcal{X}$, $f \in \mathcal{F}(\mathcal{B}_0, \Sigma_0, \tau)$, $c \in \mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$, $n \leq a_f$ *et* $m = a_c$.

Exemple 11.2.10 *Tous les systèmes de réécriture présentés aux exemples de la section 11.1 sont des systèmes avec constructeurs inductifs. C'est aussi le cas du système* $\mapsto_{\mathsf{rec}}$ *présenté à la section 11.2.1.*

Exemple 11.2.11 (Non terminaison avec types non inductifs) *La condition de positivité des types inductifs est très importante [Men87]. C'est elle qui permet de parler de calcul par récursion bien fondée sur un type inductif. Ainsi, certains types non inductifs interdisent toute forme de récursion bien fondée, comme c'est le cas avec le type* D *présenté à l'exemple 11.2.7, dont le constructeur est*

$$\mathsf{c} : (\mathsf{D} \Rightarrow \mathsf{U}) \Rightarrow \mathsf{D}\ .$$

En utilisant D*, on peut facilement typer un système de réécriture non récursif et non normalisant (voir [Men87]). Considérons le symbole* $\mathsf{p} : \mathsf{D} \Rightarrow \mathsf{D} \Rightarrow \mathsf{U}$ *défini par la règle*

$$\mathsf{p}\,(\mathsf{c}\,x) \quad \mapsto_{\mathsf{D}} \quad x\ .$$

En posant $\omega_{\mathsf{D}} =_{def} \lambda x.\mathsf{p}xx : \mathsf{D} \Rightarrow \mathsf{U}$*, on a* $\omega_{\mathsf{D}}\,(\mathsf{c}\,\omega_{\mathsf{D}}) : \mathsf{U}$ *et*

$$\omega_{\mathsf{D}}\,(\mathsf{c}\,\omega_{\mathsf{D}}) \quad \to_{\beta} \quad \mathsf{p}\,(\mathsf{c}\,\omega_{\mathsf{D}})\,(\mathsf{c}\,\omega_{\mathsf{D}}) \quad \to_{\mathsf{D}} \quad \omega_{\mathsf{D}}\,(\mathsf{c}\,\omega_{\mathsf{D}}) \quad \notin \quad \mathcal{SN}_{\beta\mathsf{D}}\ .$$

11.2.3 Critères de terminaison

Nous allons maintenant esquisser la façon dont on peut construire un critère de terminaison pour la réécriture avec types inductifs. On se place pour le moment dans le système $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0, \Sigma_0, \tau)$.

On repart de ce que l'on a vu au début de la section. Nous pouvons aller un peu plus loin et décrire la méthode utilisée dans [BJO02, Bla07] pour obtenir la propriété (11.3). Posons $l =_{\text{def}} \mathsf{f}l_1 \dots l_k$ et supposons qu'il existe un contexte Γ tel que $\mathcal{D}\mathrm{om}(\Gamma) = \mathsf{FV}(l)$ et que

$$\Gamma \vdash l_1 : T_1 \quad \dots \quad \Gamma \vdash l_k : T_k \quad \text{et} \quad \Gamma \vdash r : H\ .$$

On peut essayer d'obtenir (11.3) en procédant en deux temps. Soit une substitution σ telle que $l_i\sigma \in [\![T_i]\!]$ pour tout $i \in \{1, \dots, k\}$.

(i) La première étape est d'obtenir la *calculabilité* des variables instantiées de l, c'est-à-dire d'avoir une substitution σ telle que $\sigma \models \Gamma$. Cette propriété est assurée par la condition d'*accessibilité* [BJO02].

(ii) La seconde étape est d'obtenir $r\sigma \in [\![H]\!]$. Comme on a $\Gamma \vdash r : H$ et $\sigma \models \Gamma$, on peut utiliser le lemme d'adéquation 9.2.7. Pour cela, il faut que pour toute occurrence d'un terme $g\vec{u}$ dans r, $g\vec{u}\sigma$ appartienne à l'interprétation du type de $g\vec{u}$. C'est pour cela qu'a été introduite la notion de *fermeture calculable* [BJO02].

On se concentre sur la fermeture calculable. Comme nous l'avons vu, elle doit spécifier quels termes de la forme $g\vec{u}$ peuvent apparaître dans le membre droit d'une règle dont le membre gauche est $f\vec{l}$. Pour aboutir à une relation fortement normalisante, il faut qu'il y ait une forme de décroissance entre $f\vec{l}$ et $g\vec{u}$. Pour ce faire, nous comparons les arguments des termes $g\vec{u}$ et $f\vec{l}$.

Essayons maintenant de voir comment utiliser les types inductifs pour effectuer cette comparaison. L'idée est de définir une interprétation de ces types en utilisant une induction sur les ordinaux dénombrables $\mathfrak{a} \in \mathfrak{O}$ (voir la section 9.3.4).

Soit un ensemble de types de base $\mathcal{B}_0$, une signature (Σ_0, τ) typée sur $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ et un ensemble $\mathcal{C}$ de types inductifs sur $(\mathcal{B}_0, \Sigma_0, \tau)$ tels que tout $C \in \mathcal{B}_0$ est un type inductif. Pour chaque classe d'équivalence $\mathfrak{C}$ de $\simeq_{\mathcal{B}_0}$, on définit l'interprétation $[\![C]\!](\mathfrak{a})$ de $C \in \mathfrak{C}$ par induction sur l'ordinal dénombrable $\mathfrak{a} \in \mathfrak{O}$ comme suit.

On raisonne par induction sur $<_{\mathcal{B}_0}$. Soit $\mathfrak{C}$ une classe d'équivalence pour $\simeq_{\mathcal{B}_0}$ et supposons définie

$$[\![D]\!] \quad = \quad \bigcup_{\mathfrak{a}\in\mathfrak{O}} [\![D]\!](\mathfrak{a}) \tag{11.4}$$

pour tout $D \in <^{\mathfrak{C}}_{\mathcal{B}_0}$. Étant donné $C \in \mathfrak{C}$, l'interprétation $[\![C]\!](a+1)$ est définie de telle sorte que pour tout $c \in C$ de type

$$\tau(c) \quad = \quad T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow C \ ,$$

et tous termes $t_1, \ldots, t_n$, on ait

$$c\, t_1 \ldots t_n \in [\![C]\!](a+1) \qquad \text{si et seulement si} \qquad \forall i \in \{1, \ldots, n\}. \quad t_i \in [\![T_i]\!]_{\mathfrak{C}}(a) \ , \quad (11.5)$$

où $[\![T_i]\!]_{\mathfrak{C}}(a)$ est l'interprétation de T_i dans laquelle les $D \in \mathfrak{C}$ sont interprétés par $[\![D]\!](a)$.

Notons que la condition de positivité des types inductifs implique qu'on ait $D \leq_{\mathcal{B}_0} C$ pour tout $D \in \mathcal{B}_0$ apparaissant dans T_i, et de plus que toute occurrence d'un $D \in \mathfrak{C}$ dans T_i soit à une position positive. D'après la proposition 10.2.4, ceci implique la monotonicité de la fonction $[\![T_i]\!]_{\mathfrak{C}}(\_)$.

Étant donnée une classe d'équivalence $\mathfrak{C}$ pour $\simeq_{\mathcal{B}_0}$, un type T tel que tout $D \in \mathcal{B}_0$ apparaissant dans T soit dans $\leq^{\mathfrak{C}}_{\mathcal{B}_0}$, et un terme t, on définit $o_{\mathfrak{C},T}(t)$ comme étant le plus petit ordinal $a \in \mathfrak{O}$ tel que $t \in [\![T]\!]_{\mathfrak{C}}(a)$.

Remarque 11.2.12 *La définition de $[\![D]\!]$ donnée en (11.4) fait intervenir une union. Or on a vu au chapitre 10 que l'union est problématique pour les familles de réductibilité.*

En fait, dans le cas de (11.4) il n'y a pas de problèmes car c'est une union cumulative *: on a $[\![D]\!](a) \subseteq [\![D]\!](b)$ pour tous $a, b \in \mathfrak{O}$ tels que $a \leq b$.*

Ainsi, concernant la propriété (11.5), étant donnés $a, b \in \mathfrak{O}$ avec $a + 1 \leq b$, on a $c\, t_1 \ldots t_n \in [\![C]\!](b)$ si $t_i \in [\![T_i]\!]_{\mathfrak{C}}(a)$ pour tout $i \in \{1, \ldots, n\}$.

Évoquons maintenant deux critères de terminaison s'appuyant sur les types inductifs.

11.2.3.(a) Schéma général

Tout d'abord, abordons le critère introduit dans [JO91] puis développé dans [BJO02]. On suppose que les types inductifs $C \in \mathcal{B}_0$ sont *strictement positifs*, c'est-à-dire tels que dans le type

$$\tau(c) \quad = \quad T_1 \Rightarrow \cdots \Rightarrow T_n \Rightarrow C \ ,$$

d'un $c \in C$, où, pour tout $i \in \{1, \ldots, n\}$, T_i est un type de la forme

$$U_1 \Rightarrow \cdots \Rightarrow U_m \Rightarrow B \ ,$$

alors, pour tout $j \in \{1, \ldots, m\}$, aucun $D \simeq_{\mathcal{B}_0} C$ n'apparaît dans U^i_j. Par exemple, les types inductifs Bool, Nat et Ord de l'exemple 11.2.6 sont strictement positifs.

Dans ce cas, par la propriété (11.5) on a

$$\begin{array}{rcll} o_C(c\, t_1 \ldots t_n) & > & o_C(t_i\, u_1 \ldots u_m) & \\ & \text{si} & & \\ (T_i \ = \ U_1 \Rightarrow \cdots \Rightarrow U_m \Rightarrow C & \text{et} & \forall j \in \{1, \ldots, m\}. \quad u_j \in [\![U_j]\!]) \ . & \end{array} \qquad (11.6)$$

Le critère de terminaison proposé dans [BJO02] est une condition syntaxique reposant sur (11.6). Il permet par exemple de montrer, pour les termes typés de $\lambda_{\Rightarrow}(\mathcal{B}_0, \Sigma_0, \tau)$,

la normalisation forte de la β-réduction combinée aux relations de réécriture issues du système $\mapsto_{\mathsf{rec}}$ de la section 11.2.1, et des systèmes $\mapsto_{\mathsf{minus}}$, $\mapsto_{\mathsf{plus}}$ et $\mapsto_{>}$ de la section 11.1.

Par contre, il ne peut pas prendre en compte le système $\mapsto_{\mathsf{div}}$ de l'exemple 11.1.5. Ceci est dû au fait que la règle

$$\mathsf{div}\ (\mathsf{S}\ x)\ (\mathsf{S}\ y) \quad \mapsto_{\mathsf{div}} \quad \mathsf{S}\ (\mathsf{div}\ (\mathsf{minus}\ x\ (\mathsf{S}\ y))\ (\mathsf{S}\ y)) \tag{11.7}$$

en fait un système non simplement terminant.

11.2.3.(b) Terminaison avec types annotés

Nous allons maintenant présenter un critère de terminaison prenant en compte la règle (11.7). Remarquons que les règles de réduction de $\mapsto_{\mathsf{minus}}$ suggèrent que pour tous $t, u \in [\![\mathsf{Nat}]\!]$, on ait

$$o_{\mathsf{Nat}}(\mathsf{S}\ t) \quad > \quad o_{\mathsf{Nat}}(\mathsf{minus}\ t\ (\mathsf{S}\ u)) \ .$$

Les *types annotés* permettent d'internaliser cette information dans le système de types. L'idée est d'annoter les types inductifs C par des *expressions de taille* définies par la grammaire suivante :

$$a \in S \quad ::= \quad \alpha \ \mid \ 0 \ \mid \ a+1 \ \mid \ \infty \ ,$$

où α appartient à un ensemble infini dénombrables de variables $\mathcal{V}s$. On a aboutit à la syntaxe suivante pour les types annotés :

$$T, U \in \mathcal{T}_{\Rightarrow \times S}(\mathcal{B}_0) \qquad ::= \qquad C^a \ \mid \ C \ \mid \ U \Rightarrow T \ \mid \ T \times U \ ,$$

où $C \in \mathcal{B}_0$ et $a \in S$. L'interprétation des types est alors définie en utilisant des assignements $\mu : \mathcal{V}s \to \mathfrak{O}$, et en posant $[\![C^a]\!]\mu =_{\text{def}} [\![C]\!](\mu(a))$.

Cette idée a été initiée dans [HPS96], puis développée dans [BFG+04, BGP05, Abe03, Abe04, Abe06] dans le cas du λ-calcul avec point fixes. Elle a été étendue au calcul des constructions avec réécriture [Bla04, Bla05a], et au calcul des constructions avec fonctions définies par point fixes [BGP06].

Pour utiliser les types annotés dans un critère de terminaison, on associe aux symboles $f \in \Sigma_0$ un type annoté $\tau_S(f) \in \mathcal{T}_{\Rightarrow \times S}(\mathcal{B}_0)$, et on montre qu'ils en respectent les annotations. Considérons par exemple le type des constructeurs de Nat et du symbole minus de l'exemple 11.1.4 :

$$0 : \mathsf{Nat}^0 \qquad \mathsf{S} : \mathsf{Nat}^\alpha \Rightarrow \mathsf{Nat}^{\alpha+1} \qquad \mathsf{minus} : \mathsf{Nat}^\alpha \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Nat}^\alpha \ .$$

Le règles de typage des types annotés sont les mêmes que celles de $\boldsymbol{\lambda}_{\Rightarrow}(\mathcal{B}_0, \Sigma, \tau)$. En ce qui concerne les symboles $f \in \Sigma_0$, on s'autorise n'importe quelle instantiation des variables de taille de $\tau_S(f)$ par des expressions de taille $a \in S$.

Par exemple, en dénotant par $\Gamma \vdash_S t : T$ les dérivations de typage avec types annotés on obtient :

$$x : \mathsf{Nat}^\alpha \vdash_S \mathsf{S}\ x \ : \ \mathsf{Nat}^{\alpha+1} \qquad \text{et} \qquad x : \mathsf{Nat}^\alpha, y : \mathsf{Nat} \vdash_S \mathsf{minus}\ x\ y \ : \ \mathsf{Nat}^\alpha \ .$$

On peut dans ce système montrer la normalisation forte de $\mapsto_{\mathsf{minus}}$ (voir [Bla04]). Le symbole div a le type suivant :

$$\tau_S(\mathsf{div}) \quad =_{\text{def}} \quad \mathsf{Nat}^\alpha \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Nat}^\alpha \ .$$

On peut alors typer la règle (11.7) de la manière suivante :

$$\begin{array}{rl} & \mathsf{x} : \mathsf{Nat}^\alpha, \mathsf{y} : \mathsf{Nat} \ \vdash_S \ \mathsf{div}\ (\mathsf{S}\ \mathsf{x})\ (\mathsf{S}\ \mathsf{y}) \quad : \ \mathsf{Nat}^{\alpha+1} \\ \text{et} \quad & \mathsf{x} : \mathsf{Nat}^\alpha, \mathsf{y} : \mathsf{Nat} \ \vdash_S \ \mathsf{S}\ (\mathsf{div}\ (\mathsf{minus}\ \mathsf{x}\ (\mathsf{S}\ \mathsf{y}))\ (\mathsf{S}\ \mathsf{y})) \ : \ \mathsf{Nat}^{\alpha+1} \ . \end{array}$$

L'argument de terminaison pour cette règle est que dans le membre gauche, le symbole div est utilisé avec le type $\mathsf{Nat}^{\alpha+1} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Nat}^{\alpha+1}$, alors que dans le membre droit il est utilisé avec le type $\mathsf{Nat}^\alpha \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Nat}^\alpha$. Il y a donc une décroissance dans *les annotations des types des arguments* de div.

Cette formulation de l'argument de terminaison dans les types des arguments des symboles est très flexible. Elle permet par exemple de ne pas se limiter aux types *strictement* positifs, comme c'était le cas à la section 11.2.3.(a).

Un exemple de type positif mais non strictement positif est le type Cont des continuations défini à l'exemple 11.2.6. Rappelons qu'il a un constructeur C de type

$$\tau(\mathsf{C}) \quad = \quad ((\mathsf{Cont} \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Cont} \ .$$

Les types annotés permettent de prendre en compte la règle suivante, définissant le symbole $\mathsf{e} : \mathsf{Cont} \Rightarrow \mathsf{Nat}$:

$$\mathsf{e}\ (\mathsf{C}\ \mathsf{x}) \quad \mapsto_{\mathsf{e}} \quad \mathsf{x}\ \mathsf{e} \ .$$

La particularité de cette règle est que e apparaît dans le membre droit sans que lui soient appliqués d'arguments. Le symbole e et le type Cont sont utilisés dans [Mat00] écrire un algorithme de parcours en largeur d'arbres binaires. Avec les types annotés

$$\tau_S(\mathsf{C}) \ =_{\text{def}} \ ((\mathsf{Cont}^\alpha) \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Cont}^{\alpha+1} \qquad \text{et} \qquad \tau_S(\mathsf{e}) \ =_{\text{def}} \ \mathsf{Cont}^\alpha \Rightarrow \mathsf{Nat} \ ,$$

on a

$$\begin{array}{rl} & \mathsf{x} : (\mathsf{Cont}^\alpha \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Nat} \ \vdash_S \ \mathsf{e}\ (\mathsf{C}\ \mathsf{x}) \ : \ \mathsf{Nat} \\ \text{et} \quad & \mathsf{x} : (\mathsf{Cont}^\alpha \Rightarrow \mathsf{Nat}) \Rightarrow \mathsf{Nat} \ \vdash_S \ \mathsf{x}\ \mathsf{e} \quad : \ \mathsf{Nat} \ . \end{array}$$

L'argument de terminaison est que e est utilisé avec le type $\mathsf{Cont}^{\alpha+1} \Rightarrow \mathsf{Nat}$ dans le membre gauche et avec le type $\mathsf{Cont}^\alpha \Rightarrow \mathsf{Nat}$ dans le membre droit.

Considérons maintenant le cas du symbole pivot, défini à l'exemple 11.1.7. On se concentre sur la règle (11.1) :

$$\begin{array}{rll} \mathsf{pivot}\ \mathsf{x}\ (\mathsf{cons}\ \mathsf{y}\ \mathsf{l}) & \mapsto_{\mathsf{pivot}} & \mathsf{ite}(>\ \mathsf{y}\ \mathsf{x},\ (\pi_1\ (\mathsf{pivot}\ \mathsf{x}\ \mathsf{l}),\ \mathsf{cons}\ \mathsf{y}\ (\pi_2\ (\mathsf{pivot}\ \mathsf{x}\ \mathsf{l}))), \\ & & (\mathsf{cons}\ \mathsf{y}\ (\pi_1\ (\mathsf{pivot}\ \mathsf{x}\ \mathsf{l})),\ \pi_2\ (\mathsf{pivot}\ \mathsf{x}\ \mathsf{l}))) \end{array}$$

Rappelons que le symbole pivot a pour type

$$\tau(\mathsf{pivot}) \quad = \quad \mathsf{Nat} \Rightarrow \mathsf{List} \Rightarrow \mathsf{List} \times \mathsf{List} \ .$$

Pour montrer la normalisation forte de $\mapsto_{\mathsf{pivot}}$, on peut lui donner le type annoté

$$\tau_S(\mathsf{pivot}) \quad =_{\mathrm{def}} \quad \mathsf{Nat} \Rightarrow \mathsf{List}^{\alpha} \Rightarrow \mathsf{List}^{\alpha} \times \mathsf{List}^{\alpha} \ .$$

Ceci est suffisant pour montrer que le tri QUICKSORT utilisant pivot est fortement normalisant. Cependant, ce type ne reflète pas le fait que la fonction définie par $\mapsto_{\mathsf{pivot}}$ découpe la liste qui lui est passée en argument en deux listes, la taille de la liste en entrée étant la somme des tailles des listes en sortie. Or, cet argument est nécessaire pour montrer que QUICKSORT préserve la taille de la liste à trier.

11.2.3.(c) Des types annotés aux types contraints

Nous allons maintenant voir comment affiner les types annotés pour pouvoir prendre en compte le système $\mapsto_{\mathsf{pivot}}$ de manière plus satisfaisante.

On veut exprimer le fait que pivot x l renvoie une paire de listes dont la somme des tailles est égale à la taille de l. Cela revient à dire que

$$\mathsf{pivot} : \mathsf{Nat} \Rightarrow \mathsf{List}^{\alpha} \Rightarrow \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2} \qquad \text{avec} \qquad \alpha = \beta_1 + \beta_2 \ ,$$

ce qui peut être formulé de la manière suivante :

$$\mathsf{pivot} : \mathsf{Nat} \Rightarrow \forall\alpha.\ \mathsf{List}^{\alpha} \Rightarrow \exists\beta_1\beta_2\ (\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2} \ . \tag{11.8}$$

Remarque 11.2.13 *Notons que le type donné en (11.8) pour* pivot *n'est réellement représentatif du comportement de* $\mapsto_{\mathsf{pivot}}$ *que si* List^{α} *représente les listes de taille* exactement α*, au lieu de représenter les listes de taille au plus* α*, comme c'est le cas avec les types annotés de la section 11.2.3.(b) (voir la remarque 11.2.12).*

Ainsi, on s'intéresse à des *types contraints*.

11.3 Un système de types contraints

Cette section est consacrée à la présentation d'un système de types contraints. Avant d'entrer dans les détails, nous tentons d'en expliquer de manière informelle quelques points importants.

Supposons que nous disposons d'un ensemble de formules Φ. On considère des types annotés dont les annotations peuvent être gardées par des formules $\mathsf{P} \in \Phi$, et dont les variables de taille peuvent être quantifiées universellement et existentiellement. On considère donc des types qui peuvent être de la forme

$$\forall\alpha\mathsf{P}.\mathsf{T} \qquad \text{ainsi que} \qquad \exists\alpha\mathsf{P}.\mathsf{T} \ .$$

La sémantique naturelle pour ces contraintes est de les interpréter respectivement par des intersections et par des unions :

$$\begin{array}{lcl} [\![\forall\alpha\mathsf{P}.\mathsf{T}]\!]\mu & = & \bigcap_{a\in\{a \mid \mu[a/\alpha]\models\mathsf{P}\}} [\![\mathsf{T}]\!]\mu[a/\alpha] \ , \\ [\![\exists\alpha\mathsf{P}.\mathsf{T}]\!]\mu & = & \bigcup_{a\in\{a \mid \mu[a/\alpha]\models\mathsf{P}\}} [\![\mathsf{T}]\!]\mu[a/\alpha] \ . \end{array}$$

De plus, ces types sont interprétés comme des singletons : par exemple $[\![\mathsf{Nat}]\!](\mathfrak{a})$ ne contient plus les entiers de taille inférieure ou égale à $\mathfrak{a}$, mais les entiers de taille exactement $\mathfrak{a}$ (voir aussi la remarque 11.2.13). Ainsi, $[\![\exists\alpha\mathsf{P}.\mathsf{T}]\!]\mu$ est interprété par une union possiblement disjointe, et cela pose des difficultés d'après ce que l'on a vu au chapitre 10. On revient sur ce point à la section 11.4.1.

Une approche similaire a déjà été développée pour le langage ML par Xi dans [Xi02], en s'appuyant sur le système de types développé dans [Xi98]. Il s'agit de *types dépendants* dans le sens où les quantifications $\forall$ et $\exists$ au niveau des types sont reflétées au niveau des termes.

Cette formulation se comporte très bien en tant que système de types. En particulier, la contrainte existentielle est représentée par une somme dépendante, qui a l'avantage de ne pas nécessiter d'union dans son interprétation.

Cependant, avec les types dépendants, l'information de typage se retrouve au niveau des termes, et de ce fait les informations de taille manipulées dans les types apparaissent au niveau des termes. Un tel système n'est donc pas directement utilisable comme critère de terminaison pour $\boldsymbol{\lambda}_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \tau)$. Pour pallier à ce problème, Xi [Xi98] développe une approche à deux niveaux : l'utilisateur écrit sont programme dans un langage proche de ML et un mécanisme *d'élaboration* infère les annotations sur les termes qui correspondent au typage dépendant.

Nous avons choisi pour notre système d'essayer une approche différente. Nos types contraints sont typés de manière implicite (sauf pour l'élimination de la quantification existentielle — nous revenons sur ce point à la section 11.3.2), et nous avons défini un algorithme de vérification de typage adapté, présenté à la section 11.3.3.

Il y a deux autres différences importantes entre notre système et le système de Xi [Xi02]. D'une part, nous nous intéressons à la réécriture alors qu'il s'intéresse aux définitions de fonctions par point fixes, et d'autre part nous nous intéressons à la normalisation forte de termes non clos alors qu'il s'intéresse à la terminaison de la réduction en appel par valeur sur des termes clos.

11.3.1 Types et contraintes

Dans cette section nous définissons formellement le système de types contraints que nous utilisons pour notre critère de terminaison.

De même que Xi [Xi02], nous ne considérons que des types inductifs du premier ordre.

Définition 11.3.1 *Soit* C *un type inductif sur* $(\mathcal{B}_0, \Sigma_0, \tau)$ *et soit* $\mathfrak{C}$ *sa classe d'équivalence pour* $\simeq_{\mathcal{B}_0}$. *Alors* C *est un* type inductif du premier ordre *si pour tout* $\mathsf{c} \in \mathsf{C}$, *on a*

$$\tau(\mathsf{c}) \quad = \quad \mathsf{D}_1 \Rightarrow \cdots \Rightarrow \mathsf{D}_k \Rightarrow \mathsf{C}_{k+1} \Rightarrow \cdots \Rightarrow \mathsf{C}_n \Rightarrow \mathsf{C}\ ,$$

où $\mathsf{D}_i <^{\mathfrak{C}}_{\mathcal{B}_0}$ *pour tout* $i \in \{1, \ldots, k\}$ *et* $\mathsf{C}_i \in \mathfrak{C}$ *pour tout* $i \in \{k+1, \ldots, n\}$.

Exemple 11.3.2 *Les types inductifs* Bool, Nat *et* List *de l'exemple 11.2.6 sont des types inductifs du premier ordre.*

Nous allons maintenant voir quelles sont les contraintes que nous considérons, et comment elles sont interprétées. Ensuite, nous définissons les types contraints et enfin nous définissons une relation de typage contraint qui leur correspond.

11.3.1.(a) Contraintes

L'intérêt des types inductifs du premier ordre est qu'ils peuvent être interprétés par induction sur $\mathbb{N}$. Ainsi, nous pouvons utiliser des contraintes de l'arithmétique de Presburger pour les types inductifs $\mathsf{C} \in \mathcal{B}_0$ et des contraintes booléennes pour Bool.

Définition 11.3.3 *Soit $\mathfrak{S}$ une signature multisortée, au sens de la définition 1.1.20, contenant*

— *les sortes* nat *et* bool,

— *les opérations* $\mathsf{t} : \mathsf{bool}$ *et* $\mathsf{f} : \mathsf{bool}$ *ainsi que* $0, 1 : \mathsf{nat}$, $\_ + \_ : \mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$ *et* $\mathsf{max} : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$ *;*

soit de plus $\mathcal{Vs} = \mathcal{Vs}_{\mathsf{bool}} \uplus \mathcal{Vs}_{\mathsf{nat}}$ un ensemble dénombrable de variables de taille *et $\mathcal{P}$ une signature contenant $\_ = \_$, $\_ \leq \_$ et $\_ < \_$.*

L'ensemble $\mathsf{F}(\mathcal{P}, \mathfrak{S}, \mathcal{Vs})$ des contraintes brutes *est défini par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{C}, \mathsf{D} \in \mathsf{F}(\mathcal{P}, \mathfrak{S}, \mathcal{Vs}) & ::= & \top \mid \mathsf{P}(\mathfrak{a}_1, \ldots, \mathfrak{a}_n) \\ & & \mid \mathsf{C} \wedge \mathsf{D} \mid \mathsf{C} \vee \mathsf{D} \mid \mathsf{C} \supset \mathsf{D} \\ & & \mid \exists \vec{\alpha}.\mathsf{C} \mid \forall \vec{\alpha}.\mathsf{D} \end{array}$$

où $\mathsf{P} \in \mathcal{P}_n$, $\mathfrak{a}_1, \ldots, \mathfrak{a}_n \in \mathcal{T}\mathrm{er}(\mathfrak{S}, \mathcal{Vs})$ et où $\vec{\alpha}$ est une séquence de variables de taille deux à deux distinctes. On désigne par $\mathsf{FV}_{\mathcal{S}}(\mathsf{C})$ l'ensemble des variables de tailles libres dans la contrainte brute C.

L'ensemble $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{Vs})$ des contraintes *est l'ensemble $\mathsf{F}(\mathcal{P}, \mathfrak{S}, \mathcal{Vs})/\equiv$, où $\equiv$ est la plus petite relation d'équivalence sur $\mathsf{F}(\mathcal{P}, \mathfrak{S}, \mathcal{Vs})$ telle que*

$$\begin{array}{rcll} \exists\vec{\alpha}.\ \mathsf{P} \wedge \exists\vec{\beta}.\ \mathsf{Q} & \equiv & \exists\vec{\alpha}\vec{\beta}.\ \mathsf{P} \wedge \mathsf{Q} & \text{si } \vec{\beta} \notin \mathsf{FV}_{\mathcal{S}}(\mathsf{P}) \ , \\ \forall\vec{\alpha}.\ \mathsf{P} \supset (\forall\vec{\beta}.\ \mathsf{Q} \supset \mathsf{R}) & \equiv & \forall\vec{\alpha}\vec{\beta}.\ (\mathsf{P} \wedge \mathsf{Q}) \supset \mathsf{R} & \text{si } \vec{\beta} \notin \mathsf{FV}_{\mathcal{S}}(\mathsf{P}) \\ \mathcal{Q}\vec{\alpha}\vec{\beta}.\ \mathsf{P} & \equiv & \mathcal{Q}\vec{\beta}\vec{\alpha}.\ \mathsf{P} & \text{si } \mathcal{Q} \in \{\forall, \exists\} \ . \end{array}$$

Les variables de taille $\vec{\alpha}$ sont liées dans les expressions de la forme $\mathcal{Q}\vec{\alpha}.\mathsf{P}$, où $\mathcal{Q} \in \{\forall, \exists\}$, et les contraintes sont identifiées modulo renommage des variables liées. On ne détaille pas cette construction, qui peut être définie pour les contraintes de manière analogue à ce que l'on a fait à la section 1.2.

On désigne par $\mathsf{C}[\vec{\mathfrak{a}}/\vec{\alpha}]$ la substitution sans capture de $\vec{\mathfrak{a}}$ pour $\vec{\alpha}$ dans la contrainte C et par $\mathsf{FV}_{\mathcal{S}}(\mathsf{C})$ l'ensemble des variables de tailles libres dans la contrainte C. Lorsque que le contexte le permet, on écrit $\mathsf{FV}(\mathsf{C})$ pour $\mathsf{FV}_{\mathcal{S}}(\mathsf{C})$.

Voyons maintenant comment ces contraintes sont interprétées.

Définition 11.3.4 *On désigne par $\mathfrak{D} = (\mathfrak{D}_s)_{s \in \{\mathsf{bool}, \mathsf{nat}\}}$ une $\mathfrak{S}$-algèbre telle que*

— $\mathfrak{D}_{\mathsf{bool}} = \{\mathfrak{t}, \mathfrak{f}\}$ *avec* $\mathsf{t}_{\mathfrak{D}} = \mathfrak{t}$, *et* $\mathsf{f}_{\mathfrak{D}} = \mathfrak{f}$,

— $\mathfrak{D}_{\mathsf{nat}} = \mathbb{N}$, *dans lequel les opérations* $0, 1$, $\_ + \_$ *et* max *sont interprétées de façon standard.*

Les termes de $\mathcal{T}er(\mathfrak{S}, \mathcal{V}s)$ sont interprétés dans $\mathfrak{D}$ via des assignements $\mu : \mathcal{V}s \to \mathfrak{D}$ respectant les sortes : pour tout $s \in \{\mathsf{bool}, \mathsf{nat}\}$, on a $\mu(\alpha) \in \mathfrak{D}_s$ si $\alpha \in \mathcal{V}s_s$.

On s'intéresse à la *satisfaction* des formules de $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$ dans $\mathfrak{D}$, qui est définie de façon standard, où $\top$ est interprétée par la proposition toujours vraie. Étant donné $C \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$, on écrit $\mu \models C$ si la formule C est satisfaite par μ dans $\mathfrak{D}$. Notons que pour tout $\vec{a} \in \mathcal{T}er(\mathfrak{S}, \mathcal{V}s)$, on a $\mu \models P[\vec{a}/\vec{\alpha}]$ si et seulement si $\mu[\mu(\vec{a})/\vec{\alpha}] \models P$.

11.3.1.(b) Types contraints

On considère des types annotés dont les annotations peuvent être gardées par des formules $P \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$, et dont les variables de taille peuvent être quantifiées universellement et existentiellement.

Définition 11.3.5 (Types contraints) *Soit $\mathcal{B}_0$ un ensemble de types de base, une signature $\mathcal{P}$, une signature multisortée $\mathfrak{S}$ et un ensemble de variables $\mathcal{V}s$ vérifiant les conditions de la définition 11.3.3.*

(i) L'ensemble $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$ des types contraints *sur $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$ avec types de base dans $\mathcal{B}_0$ est défini par la grammaire suivante :*

$$\begin{array}{rcl} T, U \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)) & ::= & B^a \\ & | & U \Rightarrow T \quad | \quad T \times U \\ & | & \forall\vec{\alpha}P.\, T \quad | \quad \exists\vec{\alpha}P.\, T \end{array}$$

où $B \in \mathcal{B}_{0\mathsf{Bool}}$, $a \in \mathcal{T}er(\mathfrak{S}, \mathcal{V}s)$, $\vec{\alpha} \in \mathcal{V}s$ et $P \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$.

(ii) L'ensemble des types contraints simples *sur $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$ avec types de base dans $\mathcal{B}_0$ est défini par la grammaire suivante :*

$$T, U \quad ::= \quad \exists\alpha\top.\, B^\alpha \quad | \quad U \Rightarrow T \quad | \quad T \times U\ ,$$

où $B \in \mathcal{B}_{0\mathsf{Bool}}$, $a \in \mathcal{T}er(\mathfrak{S}, \mathcal{V}s)$ et $\vec{\alpha} \in \mathcal{V}s$.

De même qu'avec les contraintes, les variables de taille $\vec{\alpha}$ sont liées dans les expressions de la forme $\mathcal{Q}\vec{\alpha}P.T$, où $\mathcal{Q} \in \{\forall, \exists\}$, et les types contraints sont identifiés modulo renommage des variables liées.

On désigne par $T[\vec{a}/\vec{\alpha}]$ la substitution sans capture de $\vec{a}$ pour $\vec{\alpha}$ dans le type T, et par $\mathsf{FV}_{\mathcal{S}}(T)$ l'ensemble des variables de taille libres dans le type T. Lorsque que le contexte le permet, on écrit $\mathsf{FV}(T)$ pour $\mathsf{FV}_{\mathcal{S}}(T)$. De plus, on écrit $\exists\vec{\alpha}.T$ (resp. $\forall\vec{\alpha}.T$) pour désigner $\exists\vec{\alpha}\top.T$ (resp. $\forall\vec{\alpha}\top.T$).

Définition 11.3.6 *Parmi les types contraints sur $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s)$ avec types de base dans $\mathcal{B}_0$, on distingue*

— *les* types annotés basés sur les produits *:*

$$T, U \in \mathcal{T}_{\times}(\mathcal{B}_0, \mathfrak{S}, \mathcal{V}s) \quad ::= \quad B^a \quad | \quad T \times U$$

où $B \in \mathcal{B}_{0\mathsf{Bool}}$ et $a \in \mathcal{T}er(\mathfrak{S}, \mathcal{V}s)$,

— *les* types contraints basés sur les produits *:*

$$\mathsf{T} \in \mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})) \quad ::= \quad \mathsf{B} \quad | \quad \exists\vec{\alpha}\mathsf{P}.\mathsf{T} \quad | \quad \forall\vec{\alpha}\mathsf{P}.\mathsf{T}$$

où $\vdash \exists\vec{\alpha}.\mathsf{P}$, $\mathsf{B} \in \mathcal{T}_\times(\mathcal{B}_0, \mathfrak{S}, \mathcal{V}\mathfrak{s})$, $\mathsf{P} \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})$ *et* $\vec{\alpha} \in \mathcal{V}\mathfrak{s}$.

Lorsque le contexte le permet, on désigne $\mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ par $\mathcal{T}_{\exists\forall\times}$.

Remarque 11.3.7 *Les types contraints basés sur les produits sont les « ∃-basic types » de [BR06] étendus au cas de la quantification universelle.*

Exemple 11.3.8 *Voici le type contraint qu'on peut donner aux constructeurs des types* Nat *et* List *:*

$$\begin{array}{rcl@{\qquad}rcl} \tau_{\mathcal{S}}(0) & =_{def} & \mathsf{Nat}^0 & \tau_{\mathcal{S}}(\mathsf{S}) & =_{def} & \forall\alpha.\ \mathsf{Nat}^\alpha \Rightarrow \mathsf{Nat}^{\alpha+1} \\ \tau_{\mathcal{S}}(\mathsf{nil}) & =_{def} & \mathsf{List}^0 & \tau_{\mathcal{S}}(\mathsf{cons}) & =_{def} & \mathsf{Nat} \Rightarrow \forall\alpha.\ \mathsf{List}^\alpha \Rightarrow \mathsf{List}^{\alpha+1}\ . \end{array}$$

Comme on l'a vu en (11.8), il peut être intéressant de typer le symbole pivot *de la manière suivante :*

$$\tau_{\mathcal{S}}(\mathsf{pivot}) \quad = \quad \mathsf{Nat} \Rightarrow \forall\alpha.\ \mathsf{List}^\alpha \Rightarrow \exists\beta_1\beta_2\ (\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ .$$

En effaçant les contraintes et les annotations de taille, les types contraints correspondent aux types simples avec produits et les types contraints basés sur les produits correspondent aux types basés sur les produits, au sens de la définition 9.1.31.

Définition 11.3.9 *On définit la fonction* $\underline{(\ )} : \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})) \to \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ *par induction de la manière suivante :*

$$\begin{array}{rcll} \underline{\mathsf{B}^a} & =_{def} & \mathsf{B} & si\ \mathsf{B} \in \mathcal{B}_0\ , \\ \underline{\mathcal{Q}\vec{\alpha}\mathsf{P}.\mathsf{T}} & =_{def} & \underline{\mathsf{T}} & si\ \mathcal{Q} \in \{\exists, \forall\}\ , \\ \underline{\mathsf{U}\ \Xi\ \mathsf{T}} & =_{def} & \underline{\mathsf{U}}\ \Xi\ \underline{\mathsf{T}} & si\ \Xi \in \{\Rightarrow, \times\}\ . \end{array}$$

Si $\tau_{\mathcal{S}} : \Sigma_0 \to \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$, alors $\underline{\tau_{\mathcal{S}}}$ désigne la fonction de Σ_0 dans $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ telle que $\underline{\tau_{\mathcal{S}}}(\mathsf{f}) = \underline{\tau_{\mathcal{S}}(\mathsf{f})}$ pour tout $\mathsf{f} \in \Sigma_0$. De plus, il est clair que $\underline{\mathsf{T}} \in \mathcal{T}_\times(\mathcal{B}_0)$ pour tout $\mathsf{T} \in \mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$.

11.3.1.(c) Typage et sous-typage contraints

Nous terminons cette section par la relation de typage pour les types contraints. Les règles de typage pour les quantifications universelles et existentielles sont dérivées des règles standard [MPS86], excepté la règle ($\exists$ E) d'élimination de la quantification existentielle, sur laquelle nous revenons à la section 11.3.2.

Les jugements de typage sont de la forme

$$\mathsf{C}; \Gamma \vdash \mathsf{t} : \mathsf{T}\ ,$$

où $\mathsf{C} \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})$.

La sémantique de ces jugements repose sur l'interprétation des types définie à la section 11.4. Pour le moment, supposons donné, pour chaque type contraint T et chaque assignement $\mu : \mathcal{V}\mathfrak{s} \to \mathfrak{D}$, un ensemble $[\![\mathsf{T}]\!]\mu \subseteq \mathcal{SN}_{\mathcal{SR}}$. Alors, comme nous le verrons au théorème 11.4.24, $\mathsf{C};\Gamma \vdash \mathsf{t} : \mathsf{T}$ signifie que pour tout (μ, σ),

$$(\mu \models \mathsf{C} \quad \wedge \quad \forall x \in \mathcal{D}om(\Gamma). \quad \sigma(x) \in [\![\Gamma(x)]\!]\mu) \quad \Longrightarrow \quad t\sigma \in [\![\mathsf{T}]\!]\mu \ .$$

D'autre part, on utilise une relation de sous-typage qui est basée sur les inclusions entre interprétations de types contraints. Dans notre cadre, il est intéressant de la formuler sous forme de génération de contraintes. Ainsi on définit une contrainte $\|\mathsf{T} \sqsubseteq \mathsf{U}\|$, telle que, comme le nous verrons à la proposition 11.4.22,

$$\forall \mu. \quad \mu \models \|\mathsf{T} \sqsubseteq \mathsf{U}\| \quad \Longrightarrow \quad [\![\mathsf{T}]\!]\mu \subseteq [\![\mathsf{U}]\!]\mu \ .$$

Définition 11.3.10 *Étant donnés* $\mathsf{T}, \mathsf{U} \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$*, on définit la* contrainte de sous-typage $\|\mathsf{T} \sqsubseteq \mathsf{U}\|$ *par induction sur* (T, U) *comme suit :*

$$\begin{array}{rcll}
\|\mathsf{B}^a \sqsubseteq \mathsf{B}^b\| & =_{def} & a = b \ , & \\
\|\mathsf{T}_2 \Rightarrow \mathsf{T}_1 \sqsubseteq \mathsf{U}_2 \Rightarrow \mathsf{U}_1\| & =_{def} & \|\mathsf{U}_2 \sqsubseteq \mathsf{T}_2\| \wedge \|\mathsf{T}_1 \sqsubseteq \mathsf{U}_1\| \ , & \\
\|\mathsf{T}_1 \times \mathsf{T}_2 \sqsubseteq \mathsf{U}_1 \times \mathsf{U}_2\| & =_{def} & \|\mathsf{T}_1 \sqsubseteq \mathsf{U}_1\| \wedge \|\mathsf{T}_2 \sqsubseteq \mathsf{U}_2\| \ , & \\
\|\mathsf{T} \sqsubseteq \exists\vec{\alpha}\mathsf{P}.\mathsf{U}\| & =_{def} & \exists\vec{\alpha}.\ \mathsf{P} \wedge \|\mathsf{T} \sqsubseteq \mathsf{U}\| & \textit{si } \vec{\alpha} \notin \mathsf{FV}(\mathsf{T}) \textit{ et } \mathsf{T} \neq \exists\vec{\beta}\mathsf{Q}.\mathsf{V} \ , \\
\|\exists\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \mathsf{U}\| & =_{def} & \forall\vec{\alpha}.\ \mathsf{P} \supset \|\mathsf{T} \sqsubseteq \mathsf{U}\| & \textit{si } \vec{\alpha} \notin \mathsf{FV}(\mathsf{U}) \ , \\
\|\mathsf{T} \sqsubseteq \forall\vec{\alpha}\mathsf{P}.\mathsf{U}\| & =_{def} & \forall\vec{\alpha}.\ \mathsf{P} \supset \|\mathsf{T} \sqsubseteq \mathsf{U}\| & \textit{si } \vec{\alpha} \notin \mathsf{FV}(\mathsf{T}) \ , \\
\|\forall\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \mathsf{U}\| & =_{def} & \exists\vec{\alpha}.\ \mathsf{P} \wedge \|\mathsf{T} \sqsubseteq \mathsf{U}\| & \textit{si } \vec{\alpha} \notin \mathsf{FV}(\mathsf{U}) \textit{ et } \mathsf{U} \neq \forall\vec{\beta}\mathsf{Q}.\mathsf{V} \ .
\end{array}$$

Remarque 11.3.11 *La définition 11.3.10 peut paraître ambiguë dans les cas*

$$\|\forall\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \exists\vec{\beta}\mathsf{Q}.\mathsf{U}\| \qquad et \qquad \|\exists\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \forall\vec{\beta}\mathsf{Q}.\mathsf{U}\| \ .$$

En effet, si $\vec{\alpha} \notin \mathsf{FV}(\mathsf{Q}, \mathsf{U})$ *et* $\vec{\beta} \notin \mathsf{FV}(\mathsf{P}, \mathsf{T})$*, on a*

$$\begin{array}{rrcl}
 & \|\forall\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \exists\vec{\beta}\mathsf{Q}.\mathsf{U}\| & = & \exists\vec{\alpha}.\ \mathsf{P} \wedge \exists\vec{\beta}.(\mathsf{Q} \wedge \|\mathsf{T} \sqsubseteq \mathsf{U}\|) \\
et & \|\forall\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \exists\vec{\beta}\mathsf{Q}.\mathsf{U}\| & = & \exists\vec{\beta}.\ \mathsf{Q} \wedge \exists\vec{\alpha}.(\mathsf{P} \wedge \|\mathsf{T} \sqsubseteq \mathsf{U}\|) \ ;
\end{array}$$

ainsi que

$$\begin{array}{rrcl}
 & \|\exists\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \forall\vec{\beta}\mathsf{Q}.\mathsf{U}\| & = & \forall\vec{\alpha}.\ \mathsf{P} \supset \forall\vec{\beta}.(\mathsf{Q} \supset \|\mathsf{T} \sqsubseteq \mathsf{U}\|) \\
et & \|\exists\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \forall\vec{\beta}\mathsf{Q}.\mathsf{U}\| & = & \forall\vec{\beta}.\ \mathsf{Q} \supset \forall\vec{\alpha}.(\mathsf{P} \supset \|\mathsf{T} \sqsubseteq \mathsf{U}\|)
\end{array}$$

L'ambiguïté est levée par le fait que ces formules sont identifiées dans $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})$*. On a en fait*

$$\begin{array}{rcl}
\|\forall\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \exists\vec{\beta}\mathsf{Q}.\mathsf{U}\| & = & \exists\vec{\alpha}\vec{\beta}.\ \mathsf{Q} \ \wedge \ \mathsf{P} \ \wedge \ \|\mathsf{T} \sqsubseteq \mathsf{U}\| \ , \\
\|\exists\vec{\alpha}\mathsf{P}.\mathsf{T} \sqsubseteq \forall\vec{\beta}\mathsf{Q}.\mathsf{U}\| & = & \forall\vec{\alpha}\vec{\beta}.\ (\mathsf{Q} \ \wedge \ \mathsf{P}) \ \supset \ \|\mathsf{T} \sqsubseteq \mathsf{U}\| \ .
\end{array}$$

Remarque 11.3.12 *Il est aisé de voir par induction sur* $\mathsf{T} \in \mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ *que la contrainte* $\|\mathsf{T} \sqsubseteq \mathsf{T}\|$ *est valide (c.-à-d. satisfaite par tout assignement). Dans ce qui suit, nous l'identifions à la contrainte* $\top$*.*

Exemple 11.3.13 *Comme les types contraints sont des « types singletons », le type* $\mathsf{Nat}^{\mathfrak{a}}$ *n'est un sous-type de* $\mathsf{Nat}^{\mathfrak{b}}$ *pour l'assignement* μ *que si* $\mu(\mathfrak{a}) = \mu(\mathfrak{b})$. *Par contre,* $\exists\alpha\ \mathsf{P}.\ \mathsf{Nat}^{\alpha}$ *est un sous-type de* $\exists\alpha.\ \mathsf{Nat}^{\alpha}$ *pour tout* $\mathsf{P} \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{X})$.

D'autre part, on a

$$\vdash \quad \alpha = \beta_1 + \beta_2 \quad \supset \quad \|\mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2+1} \sqsubseteq \exists\gamma_1\gamma_2\ (\alpha + 1 = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}\| \ ,$$

car la contrainte $\|\mathsf{List}^{\beta_1+1} \times \mathsf{List}^{\beta_2} \sqsubseteq \exists\gamma_1\gamma_2\ (\alpha + 1 = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}\|$ *est, par définition,*

$$\exists\gamma_1\gamma_2.\quad (\alpha + 1 = \gamma_1 + \gamma_2) \quad \wedge \quad (\gamma_1 = \beta_1 + 1) \quad \wedge \quad (\gamma_2 = \beta_2)\ .$$

Notre relation de typage peut être vue comme une version implicite de la relation de typage de [Xi98].

Définition 11.3.14 *On appelle* contextes contraints *sur* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ *les paires* $\mathsf{C};\Gamma$ *où* $\mathsf{C} \in \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})$ *et* Γ *est un contexte à valeurs dans* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$.

Soit $\tau_{\mathcal{S}} : \Sigma_0 \to \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$.

(i) La relation de typage $\mathsf{C};\Gamma \vdash_{\tau_{\mathcal{S}}} \mathsf{t} : \mathsf{T}$, *où* $\mathsf{t} \in \Lambda_{\mathcal{S}}(\Sigma_0)$, $\mathsf{T} \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ *et* $\mathsf{C};\Gamma$ *est un contexte contraint sur* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$, *est la plus petite relation satisfaisant aux règles de la figure 11.1.*

(ii) On désigne par $\boldsymbol{\lambda}_{\mathcal{S}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}), \Sigma_0, \tau_{\mathcal{S}})$ *le système dont*
- *les termes sont ceux de* $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$,
- *les types sont les éléments de* $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$,
- *la relation de typage est* $\mathsf{C};\Gamma \vdash_{\tau_{\mathcal{S}}} \mathsf{t} : \mathsf{T}$,
- *la relation de réduction est la relation* $\to_{\mathcal{S}}$.

(iii) Si $\mathcal{R}$ *est un système de réécriture conditionnelle typé sur* $\boldsymbol{\lambda}_{\underline{\mathcal{S}}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$ *(voir définition 11.1.1) tel que la relation* $\to_{\mathcal{SR}}$ *est confluente sur* $\mathcal{SN}_{\mathcal{SR}}$, *alors on désigne par* $\boldsymbol{\lambda}_{\mathcal{SR}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}), \Sigma_0, \tau_{\mathcal{S}})$ *le système identique à* $\boldsymbol{\lambda}_{\mathcal{S}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}), \Sigma_0, \tau_{\mathcal{S}})$, *sauf pour la relation de réduction qui est* $\to_{\mathcal{SR}}$.

Remarque 11.3.15 *Notons que contrairement à [Xi02], pour le typage de* $\mathsf{ite}(\mathsf{b}, \mathsf{t}_1, \mathsf{t}_2)$ *par la règle* (Bool), *le typage de* t_1 *et* t_2 *ne dépend pas de* b. *Cela est dû au fait que l'on s'intéresse à la normalisation forte, et non pas à la normalisation faible.*

Enfin, mentionnons une propriété importante, qui est une forme *d'affaiblissement* pour les contraintes.

Lemme 11.3.16 (Affaiblissement) *Si* $\mathsf{C};\Gamma \vdash \mathsf{t} : \mathsf{T}$ *et* $\vdash \mathsf{D} \supset \mathsf{C}$ *alors* $\mathsf{D};\Gamma \vdash \mathsf{t} : \mathsf{T}$.

Preuve. Par induction sur $\mathsf{C};\Gamma \vdash \mathsf{t} : \mathsf{T}$. On raisonne par cas sur la dernière règle utilisée. Les seuls cas qui ne suivent pas directement de l'hypothèse d'induction sont ceux des règles $(\forall\,\mathrm{I})$ et $(\exists\,\mathrm{E})$. Les deux cas sont similaires : en appliquant le lemme 1.2.10 à $\forall\vec{\alpha}\mathsf{P}.\mathsf{T}$ et $\exists\vec{\alpha}\mathsf{P}.\mathsf{T}$, on peut supposer que $\vec{\alpha} \notin \mathsf{FV}(\mathsf{D})$, et le résultat suit de l'hypothèse d'induction. □

$$(\text{Ax})\ \frac{}{C;\Gamma,x:T\vdash x:T} \qquad (\textsc{Symb}\,\text{I})\ \frac{}{C;\Gamma\vdash f:\tau(f)}\ (f\in\Sigma_0)$$

$$(\Rightarrow\text{I})\ \frac{C;\Gamma,x:U\vdash t:T}{C;\Gamma\vdash\lambda x.t:U\Rightarrow T} \qquad (\Rightarrow\text{E})\ \frac{C;\Gamma\vdash t:U\Rightarrow T \qquad C;\Gamma\vdash u:U}{C;\Gamma\vdash tu:T}$$

$$(\times\text{I})\ \frac{C;\Gamma\vdash t_1:T_1 \qquad C;\Gamma\vdash t_2:T_2}{C;\Gamma\vdash(t_1,t_2):T_1\times T_2} \qquad (\times\text{E})\ \frac{C;\Gamma\vdash t:T_1\times T_2}{C;\Gamma\vdash\pi_i t:T_i}\ (i\in\{1,2\})$$

$$(\mathsf{Bool})\ \frac{C;\Gamma\vdash b:\mathsf{Bool} \qquad C;\Gamma\vdash t_1:T \qquad C;\Gamma\vdash t_2:T}{C;\Gamma\vdash\mathsf{ite}(b,t_1,t_2):T}\ (T\in\mathcal{T}_{\exists\forall\times}(\mathcal{B}_0,\Phi(\mathcal{P},\mathcal{S},\mathcal{V}s)))$$

$$(\mathsf{let})\ \frac{C;\Gamma\vdash t:T \qquad C;\Gamma,x:T\vdash u:U}{C;\Gamma\vdash\mathsf{let}\ x=t\ \mathsf{in}\ u:U}$$

$$(\forall\,\text{I})\ \frac{C\wedge P;\Gamma\vdash t:T \qquad \vdash C\supset\exists\vec{\alpha}.P}{C;\Gamma\vdash t:\forall\vec{\alpha}P.T}\ (\vec{\alpha}\notin FV(C,\Gamma))$$

$$(\forall\,\text{E})\ \frac{C;\Gamma\vdash t:\forall\vec{\alpha}P.T \qquad \vdash C\supset P[\vec{a}/\vec{\alpha}]}{C;\Gamma\vdash t:T[\vec{a}/\vec{\alpha}]}$$

$$(\exists\,\text{I})\ \frac{C;\Gamma\vdash t:T[\vec{a}/\vec{\alpha}] \qquad \vdash C\supset P[\vec{a}/\vec{\alpha}]}{C;\Gamma\vdash t:\exists\vec{\alpha}P.T}$$

$$(\exists\,\text{E})\ \frac{C;\Gamma\vdash t:\exists\vec{\alpha}P.T \qquad C\wedge P;\Gamma,x:T\vdash u:U \qquad \vdash C\supset\exists\vec{\alpha}.P}{C;\Gamma\vdash\mathsf{let}\ x=t\ \mathsf{in}\ u:U}\ (\vec{\alpha}\notin FV(C,\Gamma,U))$$

$$(\textsc{Sub})\ \frac{C;\Gamma\vdash t:T \qquad \vdash C\supset\|T\sqsubseteq U\|}{C;\Gamma\vdash t:U}$$

FIG. 11.1: Typage contraint

11.3.2 Élimination de la quantification existentielle

Si on fait abstraction de la gestion des contraintes, le typage de la quantification universelle est standard. Par contre, la règle d'élimination de la quantification existentielle n'est pas usuelle. Une règle plus proche des règles habituelles [MPS86] serait :

$$\frac{C;\Gamma \vdash t : \exists\vec{\alpha}P.T \qquad C \wedge P;\Gamma,x:T \vdash u : U \qquad \vdash C \supset \exists\vec{\alpha}.P}{C;\Gamma \vdash u[t/x] : U} \ (\vec{\alpha} \notin FV(C,\Gamma,U))$$

Cependant, cette règle n'est pas très bonne pour la vérification du typage : si on la lit de bas en haut, étant donné un terme v, il faut chercher deux termes u et t tels que $v = u[t/x]$, et qui de plus vérifient le typage des prémisses. Nous utilisons une règle d'élimination de la quantification existentielle dans laquelle cette substitution est explicitée, au moyen de la construction $\mathsf{let}\ x = t\ \mathsf{in}\ u$:

$$(\exists\,E)\ \frac{C;\Gamma \vdash t : \exists\vec{\alpha}P.T \qquad C \wedge P;\Gamma,x:T \vdash u : U \qquad \vdash C \supset \exists\vec{\alpha}.P}{C;\Gamma \vdash \mathsf{let}\ x = t\ \mathsf{in}\ u : U} \ (\vec{\alpha} \notin FV(C,\Gamma,U))$$

C'est pour cela que l'on étudie la normalisation forte dans $\boldsymbol{\lambda}_{\underline{S}\mathcal{R}}(\mathcal{B}_0,\Sigma_0,\tau)$ et non pas dans $\boldsymbol{\lambda}_{\Rightarrow\times}(\mathcal{B}_0,\Sigma_0,\tau)$. Ainsi, les termes de notre système contiennent un peu d'information sur le typage des contraintes, ce qui fait que ce n'est pas un système de types complètement implicite.

D'autre part, la construction $\mathsf{let}\ x = t\ \mathsf{in}\ u$ code une forme de partage d'information. Pour le voir, détaillons le typage de la seconde règle définissant la fonction pivot de l'exemple 11.1.7.

Exemple 11.3.17 *À l'exemple 11.3.8, nous avons donné le type suivant au symbole* pivot *:*

$$\tau_{\mathcal{S}}(\mathsf{pivot}) \quad = \quad \mathsf{Nat} \Rightarrow \forall\alpha.\ \mathsf{List}^{\alpha} \Rightarrow \exists\beta_1\beta_2\,(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ .$$

Considérons tout d'abord la version (11.1) de sa seconde règle de réduction :

$$\begin{array}{rcl}\mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) & \mapsto_{\mathsf{pivot}} & \mathsf{ite}(>\ y\ x,\ (\pi_1\ (\mathsf{pivot}\ x\ l),\ \mathsf{cons}\ y\ (\pi_2\ (\mathsf{pivot}\ x\ l))),\\ & & (\mathsf{cons}\ y\ (\pi_1\ (\mathsf{pivot}\ x\ l)),\ \pi_2\ (\mathsf{pivot}\ x\ l)))\end{array}$$

Comme dans le contexte $\Gamma =_{def} x,y : \mathsf{Nat}, l : \mathsf{List}^{\alpha}$ *on a*

$$\Gamma \vdash \mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) \quad : \quad \exists\beta_1\beta_2\,(\alpha + 1 = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ , \qquad (11.9)$$

on voudrait avoir

$$\begin{array}{l}\Gamma \vdash \mathsf{ite}(>\ y\ x,\ (\pi_1\ (\mathsf{pivot}\ x\ l),\ \mathsf{cons}\ y\ (\pi_2\ (\mathsf{pivot}\ x\ l))),\\ (\mathsf{cons}\ y\ (\pi_1\ (\mathsf{pivot}\ x\ l)),\ \pi_2\ (\mathsf{pivot}\ x\ l))) \quad : \quad \exists\beta_1\beta_2\,(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\end{array}$$

Or ce terme fait intervenir $\mathsf{pivot}\ x\ l$*, qui a un type existentiel*

$$\Gamma \vdash \mathsf{pivot}\ x\ l \quad : \quad \exists\beta_1\beta_2\,(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ . \qquad (11.10)$$

Pour pouvoir typer le membre droit de la règle (11.1), il faut éliminer ce type existentiel et pour cela appliquer la règle $(\exists\,\mathrm{E})$*. On utilise donc la version (11.2) de la règle (11.1) :*

$$\begin{array}{lll} \mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) & \mapsto_{\mathsf{pivot}} & \mathsf{let}\ z = (\mathsf{pivot}\ x\ l)\ \mathsf{in} \\ & & \mathsf{ite}(>\ y\ x,\ (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ & & (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \end{array}$$

On a

$$\begin{array}{rcl} \alpha = \beta_1 + \beta_2\ ;\ \Gamma, z : \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2} \vdash (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)) & : & \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2+1} \\ (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z) & : & \mathsf{List}^{\beta_1+1} \times \mathsf{List}^{\beta_2} \end{array}$$

D'autre part, d'après l'exemple 11.3.13, on a

$$\begin{array}{l} \vdash \alpha = \beta_1 + \beta_2 \supset \|\mathsf{List}^{\beta_1+1} \times \mathsf{List}^{\beta_2} \sqsubseteq \exists\gamma_1\gamma_2\,(\alpha + 1 = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}\| \\ \vdash \alpha = \beta_1 + \beta_2 \supset \|\mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2+1} \sqsubseteq \exists\gamma_1\gamma_2\,(\alpha + 1 = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}\| \end{array}$$

En utilisant $>$ *avec le type*

$$\tau_{\mathcal{S}} \quad = \quad \mathsf{Nat} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Bool}\ ,$$

il s'en suit que dans le contexte contraint $\alpha = \beta_1 + \beta_2\ ;\ \Gamma, z : \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}$*, on a*

$$\begin{array}{l} \mathsf{ite}(>\ y\ x, (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \qquad\qquad : \quad \exists\gamma_1\gamma_2\,(\alpha + 1 = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2} \end{array} \tag{11.11}$$

En appliquant la règle $(\exists\,\mathrm{E})$ *à (11.10) et (11.11), on en déduit que*

$$\begin{array}{l} \Gamma \vdash \mathsf{let}\ z = (\mathsf{pivot}\ x\ l)\ \mathsf{in} \\ \mathsf{ite}(>\ y\ x,\ (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \qquad\qquad : \quad \exists\gamma_1\gamma_2\,(\alpha + 1 = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2} \end{array}$$

ce qui correspond bien à (11.9).

Remarque 11.3.18 *On peut essayer d'interpréter la règle* $(\exists\,\mathrm{E})$ *de la manière suivante : Dans la règle de réécriture (11.1), le terme* $\mathsf{pivot}\ x\ l$ *qui a un type existentiel apparaît à plusieurs endroits. Ainsi, dans un système local comme la déduction naturelle, on ne pourrait pas éliminer les quantifications existentielles des différentes occurrences de manière indépendante (par exemple en les remplaçant par des variables d'unification).*

C'est ce à quoi sert la seconde prémisse de la règle $(\exists\,\mathrm{E})$ *:*

$$\frac{C; \Gamma \vdash t : \exists\vec{\alpha}P.T \qquad C \wedge P; \Gamma, x : T \vdash u : U \qquad \vdash C \supset \exists\vec{\alpha}.P}{C; \Gamma \vdash \mathsf{let}\ x = t\ \mathsf{in}\ u : U}\ (\vec{\alpha} \notin \mathsf{FV}(C, \Gamma, U))$$

Elle dit que dans $u[t/x]$*, on peut utiliser plusieurs copies de* t*, dans la mesure où ses variables existentielles sont instantiées par des variables fraîches, qui sont les mêmes pour chaque copie.*

Remarque 11.3.19 (Préservation du type par réduction) *À cause de notre utilisation du* let *comme une substitution retardée, les types ne sont pas préservés par réduction. Par exemple, dans le contexte* $\Gamma =_{def} x : \exists\alpha.\,\mathsf{Nat}^\alpha, y : \forall\alpha.\,\mathsf{Nat}^\alpha \Rightarrow \mathsf{Nat}^\alpha$ *on ne peut pas dériver* $\top;\Gamma \vdash yx : \exists\alpha.\,\mathsf{Nat}^\alpha$.

D'un autre coté, on a $\mathsf{let}\ z = x\ \mathsf{in}\ yz \mapsto_{\mathsf{let}} yx$ *et comme on peut facilement dériver* $\top;\Gamma, z : \mathsf{Nat}^\alpha \vdash yz : \exists\alpha.\,\mathsf{Nat}^\alpha$*, on en déduit que*

$$(\exists\,\mathrm{E})\ \frac{\top;\Gamma \vdash x : \exists\alpha.\,\mathsf{Nat}^\alpha \qquad \top;\Gamma, z : \mathsf{Nat}^\alpha \vdash yz : \exists\alpha.\,\mathsf{Nat}^\alpha}{\top;\Gamma \vdash \mathsf{let}\ z = x\ \mathsf{in}\ yz : \exists\alpha.\,\mathsf{Nat}^\alpha}$$

Ceci n'est pas gênant dans notre cas. En effet, ce qui nous intéresse, c'est la normalisation forte des termes typables dans le système sans contraintes $\boldsymbol{\lambda}_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \tau)$*, et on utilise les types contraints uniquement pour raisonner statiquement sur les règles de réécriture.*

11.3.3 Vérification du typage

Nous abordons maintenant la vérification du typage contraint. La question qui nous intéresse est la suivante : étant donnés C, Γ, t et T, a-t-on $C;\Gamma \vdash t : T$? Ce problème semble indécidable dans notre système. Nous pouvons cependant donner un ensemble de règles d'inférence, à partir desquelles on peut dériver un algorithme partiel de vérification.

D'une manière analogue au processus d'élaboration de Xi [Xi98], nous définissons deux relations, l'une de synthèse et l'autre de vérification de type, chacune générant une contrainte.

La relation $C;\Gamma \vdash t \uparrow T$ de *synthèse de type*. Étant donnés Γ et t, cette relation essaye de générer une contrainte C et de synthétiser un type T tels que $C;\Gamma \vdash t : T$.

La relation $C;\Gamma \vdash t \downarrow T$ de *vérification de type*. Étant donnés Γ, t et T, cette relation essayer de générer une contrainte C telle que $C;\Gamma \vdash t : T$.

Ainsi, de même que dans [Xi98], ces relations de synthèse et de vérification de type mélangent deux paradigmes de l'inférence/vérification de type :

— L'inférence/vérification de type bi-directionnelle, voir par exemple [DP00, Abe04], dans laquelle sont utilisées une relation d'inférence de type et une relation de vérification de types. Le processus d'inférence/vérification de type passe de l'une à l'autre en fonction des changements de polarité.

— L'inférence/vérification de type par génération de contraintes, telle que présentée par exemple dans [SP07].

Définition 11.3.20 *Les relations* $C;\Gamma \vdash t \uparrow T$ *de* synthèse de type *et* $C;\Gamma \vdash t \downarrow T$ *de* vérification de types *sont définies sur la figure 11.2.*

Théorème 11.3.21 *Si* $C;\Gamma \vdash t \downarrow T$ *ou* $C;\Gamma \vdash t \uparrow T$ *alors* $C;\Gamma \vdash t : T$.

Preuve. Par induction sur la définition des relations $C;\Gamma \vdash t \downarrow T$ et $C;\Gamma \vdash t \uparrow T$.

($\uparrow$ Ax) et ($\uparrow$ Symb I). On a $\top;\Gamma, x : T \vdash x : T$ et $\top;\Gamma \vdash f : \tau(f)$.

($\uparrow\Rightarrow$ E). D'après l'hypothèse d'induction et le lemme 11.3.16 on a $C \wedge D;\Gamma \vdash t : U \Rightarrow T$ et $C \wedge D;\Gamma \vdash u : U$, d'où $C \wedge D;\Gamma \vdash tu$.

$$(\uparrow \text{Ax})\ \frac{}{\top; \Gamma, x : T \vdash x \uparrow T} \qquad (\uparrow \text{Symb I})\ \frac{}{\top; \Gamma \vdash f \uparrow \tau(f)}\ (f \in \Sigma_0)$$

$$(\uparrow \Rightarrow \text{E})\ \frac{C; \Gamma \vdash t \uparrow U \Rightarrow T \qquad D; \Gamma \vdash u \downarrow U}{C \wedge D; \Gamma \vdash tu \uparrow T}$$

$$(\uparrow \times \text{I})\ \frac{C_1; \Gamma \vdash t_1 \uparrow T_1 \qquad C_1; \Gamma \vdash t_2 \uparrow T_2}{C_1 \wedge C_2; \Gamma \vdash (t_1, t_2) \uparrow T_1 \times T_2} \qquad (\uparrow \times \text{E})\ \frac{C; \Gamma \vdash t \uparrow T_1 \times T_2}{C; \Gamma \vdash \pi_i t \uparrow T_i}\ (i \in \{1, 2\})$$

$$(\uparrow \forall \text{E})\ \frac{C; \Gamma \vdash t \uparrow \forall \vec{\alpha} P.T}{C \wedge P; \Gamma \vdash t \uparrow T}\ (\vec{\alpha} \notin FV(C, \Gamma))$$

$$(\uparrow \exists \text{E})\ \frac{C; \Gamma \vdash t \uparrow \exists \vec{\alpha} P.T \qquad D; \Gamma, x : T \vdash u \uparrow U}{C \wedge \exists \vec{\alpha} P \wedge \forall \vec{\alpha}(P \supset D); \Gamma \vdash \text{let } x = t \text{ in } u \uparrow \exists \vec{\alpha} P.U}\ \begin{cases} \vec{\alpha} \notin FV(C, \Gamma) \\ n \geq 0 \end{cases}$$

$$(\downarrow \Rightarrow \text{I})\ \frac{C; \Gamma, x : U \vdash t \downarrow T}{C; \Gamma \vdash \lambda x.t \downarrow U \Rightarrow T}$$

$$(\downarrow \text{Bool})\ \frac{C; \Gamma \vdash b \downarrow \exists \alpha \text{Bool}^{\alpha} \quad D; \Gamma \vdash t_1 \downarrow T \quad E; \Gamma \vdash t_2 \downarrow T}{C \wedge D \wedge E; \Gamma \vdash \text{ite}(b, t_1, t_2) \downarrow T}\ (T \in \mathcal{T}_{\exists \forall \times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathcal{S}, \mathcal{V}s)))$$

$$(\downarrow \forall \text{I})\ \frac{C; \Gamma \vdash t \downarrow T}{\exists \vec{\alpha} P \wedge \forall \vec{\alpha}(P \supset C); \Gamma \vdash t \downarrow \forall \vec{\alpha} P.T}\ (\vec{\alpha} \notin FV(\Gamma))$$

$$(\downarrow \forall \text{E})\ \frac{C; \Gamma \vdash t \uparrow \forall \vec{\alpha} P.T}{C \wedge P[\vec{a}/\vec{\alpha}]; \Gamma \vdash t \downarrow T[\vec{a}/\vec{\alpha}]}$$

$$(\downarrow \exists \text{I})\ \frac{C; \Gamma \vdash t \downarrow T}{C \wedge P; \Gamma \vdash t \downarrow \exists \vec{\alpha} P.T}$$

$$(\downarrow \exists \text{E})\ \frac{C; \Gamma \vdash t \uparrow \exists \vec{\alpha} P.T \qquad D; \Gamma, x : T \vdash u \downarrow U}{C \wedge \exists \vec{\alpha} P \wedge \forall \vec{\alpha}(P \supset D); \Gamma \vdash \text{let } x = t \text{ in } u \downarrow U}\ (\vec{\alpha} \notin FV(C, \Gamma, U))$$

$$(\downarrow \text{Sub})\ \frac{C; \Gamma \vdash t \uparrow U}{C \wedge \|U \sqsubseteq T\| \Gamma \vdash t \downarrow T}$$

Fig. 11.2: Vérification du typage contraint

($\uparrow \times$I). D'après l'hypothèse d'induction et le lemme 11.3.16 on a $C_1 \wedge C_2; \Gamma \vdash t_i : T_i$ pour tout $i \in \{1,2\}$, d'où $C_1 \wedge C_2; \Gamma \vdash (t_1, t_2) : T_1 \times T_2$.

($\uparrow \times$E). Par hypothèse d'induction.

($\uparrow \forall$E). D'après l'hypothèse d'induction et le lemme 11.3.16 on a $C \wedge P; \Gamma \vdash t : \forall \vec{\alpha} P.T$, d'où $C \wedge P; \Gamma \vdash t : T$ car $\vdash (C \wedge P) \supset P$ par hypothèse.

($\uparrow \exists$E). Posons $E =_{\text{def}} C \wedge \exists \vec{\alpha} P \wedge \forall \vec{\alpha}(P \supset D)$. D'après l'hypothèse d'induction et le lemme 11.3.16, on a $E; \Gamma \vdash t : \exists \vec{\alpha} P.T$ et $E \wedge P; \Gamma, x : T \vdash u : U$. Il s'en suit que $E \wedge P; \Gamma, x : T \vdash u : \exists \vec{\alpha} P.U$ car $\vdash (E \wedge P) \supset P$.

D'autre part, on a $\vdash E \supset \exists \vec{\alpha} P$ et $\vec{\alpha} \notin FV(E, \Gamma, \exists \vec{\alpha} P.U)$ car $\vec{\alpha} \notin FV(C, \Gamma)$.

On en déduit que $E; \Gamma \vdash \mathsf{let}\ x = t\ \mathsf{in}\ u : \exists \vec{\alpha} P.U$.

($\downarrow \Rightarrow$I). Par hypothèse d'induction.

($\downarrow \mathsf{Bool}$). Par hypothèse d'induction et en utilisant le lemme 11.3.16.

($\downarrow \forall$I). Posons $E =_{\text{def}} \exists \vec{\alpha} P \wedge \forall \vec{\alpha}(P \supset C)$. Par l'hypothèse d'induction et le lemme 11.3.16, on a $E \wedge P; \Gamma \vdash t : T$, d'où $E; \Gamma \vdash t : \forall \vec{\alpha} P.T$ car $\vdash E \supset \exists \vec{\alpha} P$ et $\vec{\alpha} \notin FV(E, \Gamma)$ car $\vec{\alpha} \notin FV(\Gamma)$.

($\downarrow \forall$E). Posons $E =_{\text{def}} C \wedge P[\vec{a}/\vec{\alpha}]$. D'après l'hypothèse d'induction et le lemme 11.3.16, on a $E; \Gamma \vdash t : \forall \vec{\alpha} P.T$, d'où $E; \Gamma \vdash t : T[\vec{a}/\vec{\alpha}]$ car $\vdash E \supset P[\vec{a}/\vec{\alpha}]$.

($\downarrow \exists$I). D'après l'hypothèse d'induction et le lemme 11.3.16, on a $C \wedge P; \Gamma \vdash t : T$, d'où $C \wedge P; \Gamma \vdash t : \exists \vec{\alpha} P.T$ car $\vdash (C \wedge P) \supset P$.

($\downarrow \exists$E). Posons $E =_{\text{def}} C \wedge \exists \vec{\alpha} P \wedge \forall \vec{\alpha}(P \supset D)$. D'après l'hypothèse d'induction et le lemme 11.3.16, on a $E; \Gamma \vdash t : \exists \vec{\alpha} P.T$ et $E \wedge P; \Gamma, x : T \vdash u : U$.

D'autre part, on a $\vdash E \supset \exists \vec{\alpha} P$ et $\vec{\alpha} \notin FV(E, \Gamma, U)$ car $\vec{\alpha} \notin FV(C, \Gamma, U)$.

On en déduit que $E; \Gamma \vdash \mathsf{let}\ x = t\ \mathsf{in}\ u : U$.

($\downarrow$ Sub). Par hypothèse d'induction et en utilisant le lemme 11.3.16. □

Remarque 11.3.22 *Pour vérifier que* $C; \Gamma \vdash t : T$ *est dérivable, on peut essayer de générer une contrainte* D *telle que* $D; \Gamma \vdash t \downarrow T$. *Par le théorème 11.3.21, on en déduit que* $D; \Gamma \vdash t : T$, *donc d'après le lemme 11.3.16, on a* $C; \Gamma \vdash t : T$ *si* $\vdash C \supset D$.

Exemple 11.3.23 *Nous allons voir comment dériver une contrainte* C *et un type* T *tels que* $C; \Gamma \vdash \mathsf{pivot}\ x\ l \uparrow T$ *dans le contexte* $\Gamma =_{def} x : \mathsf{Nat}, l : \mathsf{List}^{\gamma}$.

Rappelons que par convention Nat *est une notation abrégée pour* $\exists \alpha.\ \mathsf{Nat}^{\alpha}$. *De plus, pour alléger les notations, nous ne faisons pas apparaître la contrainte* $\top$.

Pour inférer le type de pivot x l, *il faut commencer par inférer le type de* pivot x. *Comme on a*

$$; \Gamma \vdash \mathsf{pivot} \uparrow \mathsf{Nat} \Rightarrow \forall \gamma.\ \mathsf{List}^{\gamma} \Rightarrow \exists \delta_1 \delta_2\ (\gamma = \delta_1 + \delta_2).\ \mathsf{List}^{\delta_1} \times \mathsf{List}^{\delta_2}\ ,$$

on en déduit par la règle ($\uparrow \Rightarrow$E) *que*

$$C; \Gamma \vdash \mathsf{pivot}\ x \uparrow \forall \gamma.\ \mathsf{List}^{\gamma} \Rightarrow \exists \delta_1 \delta_2\ (\gamma = \delta_1 + \delta_2).\ \mathsf{List}^{\delta_1} \times \mathsf{List}^{\delta_2}\ ,$$

où $C; \Gamma \vdash x \downarrow \mathsf{Nat}$. *De plus, on a* $C = \|\mathsf{Nat} \sqsubseteq \mathsf{Nat}\|$ *car* $; \Gamma \vdash x \uparrow \mathsf{Nat}$.

Or, la contrainte $\|\mathsf{Nat} \sqsubseteq \mathsf{Nat}\|$, *est en fait* $\|\exists\alpha.\ \mathsf{Nat}^{\alpha} \sqsubseteq \exists\beta.\ \mathsf{Nat}^{\beta}\|$, *c'est-à-dire* $\forall\alpha.\ \exists\beta.\alpha = \beta$. *Cette contrainte est évidemment valide (voir aussi la remarque 11.3.22), et on l'identifie à* $\top$.

Ainsi, en utilisant $(\uparrow \forall \mathrm{E})$ *on obtient*

$$;\Gamma \vdash \mathsf{pivot}\ x \uparrow \mathsf{List}^{\gamma} \Rightarrow \exists\delta_1\delta_2\ (\gamma = \delta_1 + \delta_2).\ \mathsf{List}^{\delta_1} \times \mathsf{List}^{\delta_2}\ .$$

De même, comme $;\Gamma \vdash l \uparrow \mathsf{List}^{\gamma}$, *on a* $\|\mathsf{List}^{\gamma} \sqsubseteq \mathsf{List}^{\gamma}\|;\Gamma \vdash l \downarrow \mathsf{List}^{\gamma}$, *ce que l'on note* $;\Gamma \vdash l \downarrow \mathsf{List}^{\gamma}$. *Il s'en suit que* $;\Gamma \vdash \mathsf{pivot}\ x\ l \uparrow \exists\delta_1\delta_2\ (\gamma = \delta_1 + \delta_2).\ \mathsf{List}^{\delta_1} \times \mathsf{List}^{\delta_2}$.

Remarque 11.3.24 *Notons que dans l'exemple 11.3.23, il est capital, lors de l'utilisation de la règle* $(\uparrow \forall \mathrm{E})$, *de pouvoir choisir le « bon » représentant de* $\forall\vec{\alpha}\mathsf{P}.\mathsf{T}$ *pour l'*α*-équivalence.*

Remarque 11.3.25 *Si* $\Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})$ *ne permet de former que des contraintes de l'arithmétique de Presburger booléens, alors les contraintes générées par les relations* $\mathsf{C};\Gamma \vdash t \uparrow \mathsf{T}$ *et* $\mathsf{C};\Gamma \vdash t \downarrow \mathsf{T}$ *sont dans l'arithmétique de Presburger avec booléens.*

Il est connu que l'arithmétique de Presburger est décidable [Pre29], avec une complexité doublement exponentielle par rapport à la taille de la formule [FR74].

Dans notre cas, il nous semble que cette complexité élevée n'est pas trop gênante dans la mesure où les termes sur lesquels ont veut appliquer notre processus de vérification du typage sont petits (ce sont des membres droits de règles de réécriture).

11.4 Normalisation forte

Dans cette section nous définissons une sémantique de normalisation forte pour le système $\boldsymbol{\lambda}_{\mathcal{SR}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}), \Sigma_0, \tau_0)$. Nous montrons ensuite que cette interprétation est adéquate dès lors que les symboles sont calculables et que l'on dispose d'une interprétation des types de base vérifiant certaines propriétés. Enfin, nous définissons une interprétation singleton des types inductifs qui satisfait ces propriétés.

Rappelons qu'étant donné un ensemble de règles conditionnelles vérifiant les conditions de la définition 11.1.1, la relation de réécriture $\to_{\mathcal{SR}}$ est l'union de la relation

$$\to_{\mathcal{S}} \quad =_{\mathrm{def}} \quad \to_{\beta} \cup \to_{\pi} \cup \to_{\mathsf{ite}} \cup \to_{\mathsf{let}}\ ,$$

et de la relation de réécriture $\mathcal{S}$-conditionnelle par joignabilité issue de $\mathcal{R}$ (voir section 4.1), que l'on désigne par $\to_{\mathcal{R}}$.

11.4.1 Sémantique de normalisation

Les types $\mathsf{T} \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ sont interprétés, via des assignements $\mu : \mathcal{V}\mathfrak{s} \to \mathfrak{D}$, par des ensembles $[\![\mathsf{T}]\!]\mu \subseteq \mathcal{SN}_{\mathcal{SR}}$. La sémantique naturelle pour les types universels et existentiels est d'utiliser respectivement des intersections et des unions :

$$\begin{array}{lcl} [\![\forall\vec{\alpha}\mathsf{P}.\mathsf{T}]\!]\mu & = & \bigcap_{\vec{a}\in\{\vec{a}\ \mid\ \mu[\vec{a}/\vec{\alpha}]\models\mathsf{P}\}} [\![\mathsf{T}]\!]\mu[\vec{a}/\vec{\alpha}]\ , \\ [\![\exists\vec{\alpha}\mathsf{P}.\mathsf{T}]\!]\mu & = & \bigcup_{\vec{a}\in\{\vec{a}\ \mid\ \mu[\vec{a}/\vec{\alpha}]\models\mathsf{P}\}} [\![\mathsf{T}]\!]\mu[\vec{a}/\vec{\alpha}]\ . \end{array}$$

Or, on a vu au chapitre 10 que pour certains systèmes de réécriture il est impossible d'avoir une famille de réductibilité stable par union. On aborde ici le problème en séparant les cas des réductions $\to_{\beta\pi\mathsf{let}}$ et des réductions $\to_{\mathcal{R}\mathsf{ite}}$.

11.4.1.(a) Termes neutres

On commence par voir quels peuvent être les termes neutres pour notre système. On va voir qu'il est intéressant d'étendre un peu le système pour pouvoir appliquer aux types inductifs la méthode présentée à la section 9.3.4.

On part de ce que l'on a déjà vu à la section 9.3.4. On suppose donné un ensemble $\mathcal{B}_0$ de types de base ainsi qu'une signature (Σ_0, τ) typée dans $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_0)$ et basée sur les produits, au sens de la définition 9.1.31. Le type des symboles $\mathsf{f} \in \Sigma_0$ est donc de la forme

$$\tau(\mathsf{f}) \quad = \quad \mathsf{T}_1 \Rightarrow \cdots \Rightarrow \mathsf{T}_{a_\mathsf{f}} \Rightarrow \mathsf{H} \ ,$$

où $\mathsf{H} \in \mathcal{T}_\times(\mathcal{B}_0)$.

Si on considère les contextes d'élimination définis en 9.1.19 :

$$\mathsf{E}[\ ] \in \mathcal{E}_{\Rightarrow\times} \quad ::= \quad [\ ] \quad | \quad \mathsf{E}[\ ]\,\mathsf{t} \quad | \quad \pi_1\mathsf{E}[\ ] \quad | \quad \pi_2\mathsf{E}[\ ] \ ,$$

alors d'après la remarque 9.3.15, les termes neutres sur $\mathcal{E}_{\Rightarrow\times}$ pour $\to_{\beta\pi\mathcal{R}}$ sont exactement les termes qui ne sont pas de la forme

— $\lambda\mathsf{x}.\mathsf{t}$,
— $(\mathsf{t}_1, \mathsf{t}_2)$,
— $\mathsf{f}\vec{\mathsf{t}}$ tels qu'il existe une règle $\mathsf{f}\vec{\mathsf{l}} \mapsto_{\mathcal{R}} \mathsf{r}$, une substitution σ, et des termes $\vec{\mathsf{u}}$ avec $|\vec{\mathsf{u}}| > 0$ tels que $\vec{\mathsf{t}}\vec{\mathsf{u}} = \vec{\mathsf{l}}\sigma$.

On considère maintenant le cas des booléens et des types inductifs. On se place dans le système $\boldsymbol{\lambda}_{\underline{\mathcal{S}}\mathcal{R}}(\mathcal{B}_0, \Sigma_0, \tau)$.

Définition 11.4.1 (Types inductifs sur les booléens) *Soit $\mathcal{B}_0$ un ensemble de types de base et (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}})$. Un* type inductif basé sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ *est un type inductif $\mathsf{B} \in \mathcal{B}_0$ sur*

$$(\mathcal{B}_{0\mathsf{Bool}},\ \Sigma_0 \uplus \{\mathsf{true}, \mathsf{false}\},\ \tau \uplus (\mathsf{true}, \mathsf{false} \mapsto \mathsf{Bool})) \ .$$

On suppose donné un ensemble $\mathcal{C}$ de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma, \tau)$. D'après la définition 11.2.8, on a $\mathcal{F} = \Sigma_0 \setminus \mathcal{C}$.

En prenant exemple sur [Bla05b], les termes suivants ne devraient pas être neutres :
— true, false,
— $\mathsf{c}\mathsf{t}_1 \dots \mathsf{t}_k$, si $\mathsf{c} \in \mathcal{C}$ et $\mathsf{k} \leq \mathsf{a}_\mathsf{c}$.

Nous allons maintenant voir comment cette notion de terme neutre peut être obtenue en suivant la méthodologie présentée à la section 9.3.4. On part de l'intuition développée à la section 10.4.2 selon laquelle les termes non neutres sont les termes « observables », dans le sens où se sont les termes avec lesquels on peut interagir. Pour cela, on va enrichir les contextes d'élimination de manière à produire les bonnes interactions.

Le cas des booléens est le plus simple : comme les symboles true et false sont éliminés par $\mathsf{ite}([\,], t, u)$, ce sont des termes non neutres pour $\mapsto_{\mathcal{SR}}$ dans les contextes d'élimination

$$E[\,] \in \mathcal{E}_{\Rightarrow\times\mathsf{Bool}} \quad ::= \quad [\,] \;\mid\; E[\,]\,t \;\mid\; \pi_1\,E[\,] \;\mid\; \pi_2\,E[\,] \;\mid\; \mathsf{ite}(E[\,], t, u)\ .$$

On aborde maintenant le cas des constructeurs $c \in \mathcal{C}$. Afin de produire les bonnes interactions, on utilise des *éliminateurs* pour les constructeurs. À chaque constructeur $c \in \mathcal{C}$ avec $a_c > 0$, on associe des symboles d'arité nulle $d_{c,1}, \ldots, d_{c,a_c}$ appelés *destructeurs*. Pour tout $j \in \{1, \ldots, a_c\}$ le destructeur $d_{c,j}$ est défini par la règle de réécriture :

$$d_{c,j}\,(c\,t_1 \ldots t_{a_c}) \quad \mapsto_{\mathcal{D}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}} \quad t_j\ .$$

Comme on veut que les constructeurs d'arité nulle ne soient pas neutres, il nous faudrait aussi des destructeurs pour les constructeurs d'arité nulle. La question est alors de savoir « que détruire ». En effet, à la différence des constructeurs d'arité non nulle, les destructeurs pour les constructeurs d'arité nulle ne peuvent pas être des « projecteurs ». À chaque constructeur c avec $a_c = 0$, nous associons un destructeur $d_{c,0}$ défini par la règle

$$d_{c,0}\,c \quad \mapsto_{\mathcal{D}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}} \quad \mho\ ,$$

où $\mho$ est un symbole d'arité nulle, qui n'est pas dans $\Sigma_0 \uplus \Sigma_{\mathsf{Bool}}$, et que nous supposons différent des $d_{c,j}$.

On désigne par $\mathcal{D}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$ (ou par $\mathcal{D}$ lorsque le contexte le permet) l'ensemble des destructeurs des types inductifs de $\mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$. De plus, lorsque le contexte le permet, on écrit $\mapsto_{\mathcal{D}}$ pour désigner le système de réécriture $\mapsto_{\mathcal{D}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}}$.

Les destructeurs peuvent être utilisés dans des contextes d'élimination donnant des termes neutres intéressants.

Définition 11.4.2 (Contextes d'élimination) *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}})$ et $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$. L'ensemble $\mathcal{E}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$ est défini par la grammaire suivante :*

$$E[\,] \in \mathcal{E}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)} \quad ::= \quad [\,] \;\mid\; E[\,]\,t \;\mid\; \pi_1\,E[\,] \;\mid\; \pi_2\,E[\,] \;\mid\; \mathsf{ite}(E[\,], t, u) \;\mid\; d\,t\ ,$$

où $t \in \Lambda(\Sigma_{\mathcal{S}})$ et $d \in \mathcal{D}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$.

Lorsque le contexte le permet on désigne $\mathcal{E}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$ par $\mathcal{E}_{\mathcal{C}}$.

Proposition 11.4.3 *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}})$, $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ et $\mathcal{R}$ un système de réécriture vérifiant les conditions de la définition 11.1.1. Alors $\mathcal{E}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$ est un ensemble de contextes d'élimination pour $\to_{\mathcal{SRD}}$.*

Définition 11.4.4 (Termes neutres) *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow\times}(\mathcal{B}_{0\mathsf{Bool}})$, $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ et $\mathcal{R}$ un système de réécriture vérifiant les conditions de la définition 11.1.1.*

L'ensemble des termes neutres sur $\mathcal{C}(\mathcal{B}_0, \Sigma_0, \tau)$, *noté $\mathcal{N}_{\mathcal{CR}}$, est l'ensemble des termes neutres dans $\mathcal{E}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$ pour $\to_{\mathcal{SRD}}$, au sens de la définition 9.3.13.*

Rappelons que si $t \in \mathcal{N}_{\mathcal{CR}}$ et $E[\ ] \in \mathcal{E}_{\mathcal{C}}$ alors $E[t] \in \mathcal{N}_{\mathcal{CR}}$.

Proposition 11.4.5 *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$, $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ et $\mathcal{R}$ un système de réécriture typé dans $\boldsymbol{\lambda}_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \tau)$.*

Les termes neutres dans $\mathcal{E}_{\mathcal{C}(\mathcal{B}_0,\Sigma_0,\tau)}$ pour $\rightarrow_{\mathcal{SRD}}$ sont exactement les termes qui ne sont pas de la forme

- $\lambda x.t$,
- (t_1, t_2),
- true, false,
- $f\vec{t}$ *tels qu'il existe une règle* $f\vec{l} \mapsto_{\mathcal{R}} r$, *une substitution* σ *et des termes* $\vec{u}$ *avec* $|\vec{u}| > 0$ *tels que* $\vec{l}\sigma = \vec{t}\vec{u}$,
- $c\,t_1 \dots t_n$ *avec* $n \leq a_c$.

Remarque 11.4.6 *Dans le cas des constructeurs, notre notion de terme neutre est proche de celle de [Bla05b] sans être identique. En effet, dans [Bla05b], les termes de la forme $c\,t_1 \dots t_n$ où c est un constructeur d'un type inductif ne sont jamais neutres, alors que dans notre cas ils le sont si $n > a_c$.*

Notons que grâce aux règles $d_{c,0}\,c \mapsto_{\mathcal{D}} \mathcal{U}$, les constructeurs c d'arité nulle ne sont pas neutres. Ceci est conforme à la notion de terme neutre de [Bla05b] et surtout à l'intuition, développée à la section 10.4.2, selon laquelle les termes neutres sont les termes qui ne sont pas des « valeurs ». De plus, en utilisant la définition 9.3.13 avec les contextes d'élimination $\mathcal{E}_{\mathcal{C}}$, ces règles permettent d'obtenir des termes neutres intéressants pour l'interprétation des types inductifs.

Ainsi, le fait d'essayer de définir les termes neutres selon un principe générique et (relativement) indépendant de la syntaxe nous a amené à ajouter des constructions au langage.

Pour les langages de programmation, cette démarche d'essayer de trouver les constructions donnant à la syntaxe un pouvoir de discrimination adapté à certaines notions « sémantiques » n'est pas nouvelle. C'est typiquement le cas de [Plo77] (voir aussi [JM96]), ainsi que de [DCLP96, DCLP98]. Nous nous sommes aussi inspirés des travaux [VM04, Vou04, MV05].

Remarquons enfin que les destructeurs et les règles de réduction $\mathcal{D}$ ne font pas vraiment partie du système $\boldsymbol{\lambda}_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \tau)$ dans le sens où ils ne sont pas typés et où on ne se préoccupe pas de la normalisation forte de $\rightarrow_{\mathcal{SRD}}$ (bien que cette relation soit fortement normalisante sur les termes typés).

11.4.1.(b) Ensembles saturés

Les réductions $\mapsto_{\beta\pi\mathsf{let}}$ sont prises en compte en interprétant les types par des ensembles saturés. Dans cette section, on suppose donnés un ensemble $\mathcal{B}_0$ de types de base, une signature (Σ_0, τ) typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$, un ensemble $\mathcal{C}$ de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ et un système de réécriture $\mathcal{R}$ vérifiant les conditions de la définition 11.1.1. On utilise les contextes d'élimination $\mathcal{E}_{\mathcal{C}}$ présentés à la définition 11.4.2.

Définition 11.4.7 *L'ensemble* $\mathbb{SAT}_{\mathcal{CR}}$ *des ensembles* $\mathcal{CR}$*-saturés est l'ensemble des parties* S *de* $\mathcal{SN}_{\mathcal{SR}}$ *telles que*

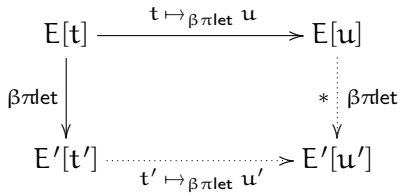
($\mathbb{SAT}0_{\mathcal{C}}$) *si* $t \in S$ *et* $t \to_{\mathcal{SR}} u$ *alors* $u \in S$,

($\mathbb{SAT}1_{\mathcal{C}}$) *si* $E[\,] \in \mathcal{E}_{\mathcal{C}}$, $E[\,] \in \mathcal{SN}_{\mathcal{SR}}$ *et* $x \in \mathcal{X}$ *alors* $E[x] \in S$,

($\mathbb{SAT}2_{\mathcal{C}}$) *si* $E[\,] \in \mathcal{E}_{\mathcal{C}}$, $t \in \mathcal{SN}_{\mathcal{SR}}$, $t \mapsto_{\beta\pi\mathsf{let}} u$ *et* $E[u] \in S$ *alors* $E[t] \in S$.

De même qu'à la section 9.1, on montre que $\mathcal{SN}_{\mathcal{SR}} \in \mathbb{SAT}_{\mathcal{CR}}$, ce qui implique que $\mathbb{SAT}_{\mathcal{CR}}$ n'est pas vide. Le point important est la standardisation faible.

Lemme 11.4.8 (Standardisation faible) *Pour tout* $E[\,] \in \mathcal{E}_{\mathcal{C}}$, *si* $t \mapsto_{\beta\pi\mathsf{let}} u$ *et* $E[t] \to_{\beta\pi\mathsf{let}} v$ *avec* $v \neq E[u]$, *alors* $v = E'[t']$ *avec* $(E[\,], t) \to_{\beta\pi\mathsf{let}} (E'[\,], v')$ *et il existe un terme* u' *tel que* $t' \mapsto_{\beta\pi\mathsf{let}} u'$ *et* $E[u] \to^*_{\beta\pi\mathsf{let}} E'[u']$. *En image :*



Preuve. Comme au lemme 9.1.23. □

Lemme 11.4.9 $\mathcal{SN}_{\mathcal{SR}} \in \mathbb{SAT}_{\mathcal{CR}}$.

Preuve. L'ensemble $\mathcal{SN}_{\mathcal{SR}}$ est évidemment stable par $\mathcal{SR}$-réduction. La clause ($\mathbb{SAT}1_{\mathcal{C}}$) se montre de la même manière qu'à la proposition 9.1.8, en utilisant la propriété suivante, qui suit du fait que $\mathcal{E}_{\mathcal{C}}$ est un ensemble de contextes d'élimination :

$$\forall v.\quad E[x] \to_{\mathcal{SR}} v \quad\Longrightarrow\quad \big(v = E'[x] \quad \text{avec} \quad E'[\,] \in \mathcal{E}_{\mathcal{C}} \quad \text{et} \quad E[\,] \to_{\mathcal{SR}} E'[\,]\big)\ .$$

La clause ($\mathbb{SAT}2_{\mathcal{C}}$) se montre comme au lemme 9.1.24, en utilisant la standardisation faible (lemme 11.4.8). □

Remarque 11.4.10 *D'après le lemme 11.4.8, si* $t[u/x] \in \mathcal{SN}_{\mathcal{SR}}$ *et* $u \in \mathcal{SN}_{\mathcal{SR}}$, *alors* $(\lambda x.t)u \in \mathcal{SN}_{\mathcal{SR}}$ *et* $\mathsf{let}\ x = u\ \mathsf{in}\ t \in \mathcal{SN}_{\mathcal{SR}}$. *De même, on a* $\pi_i(t_1, t_2) \in \mathcal{SN}_{\mathcal{SR}}$ *dès lors que* $t_1, t_2 \in \mathcal{SN}_{\mathcal{SR}}$. *Il s'en suit que la clause* ($\mathbb{SAT}2_{\mathcal{C}}$) *est équivalente à la conjonction des trois clauses suivantes :*

($\mathbb{SAT}2_{\beta}$) *si* $E[t[u/x]] \in S$ *et* $u \in \mathcal{SN}_{\mathcal{SR}}$ *alors* $E[(\lambda x.t)u] \in S$,

($\mathbb{SAT}2_{\pi i}$) *si* $E[t_i] \in S$ *et* $t_{3-i} \in \mathcal{SN}_{\mathcal{SR}}$ *alors* $E[\pi_i(t_1, t_2)] \in S$,

($\mathbb{SAT}2_{\mathsf{let}}$) *si* $E[u[t/x]] \in S$ *et* $t \in \mathcal{SN}_{\mathcal{SR}}$ *alors* $E[\mathsf{let}\ x = t\ \mathsf{in}\ u] \in S$.

Comme à la section 9.3.2, les ensembles $\mathcal{SR}$-saturés peuvent être définis par un opérateur de clôture. Ils forment donc un treillis complet par le lemme 9.3.4, dont les plus grandes bornes inférieures sont les intersections. De plus, il est immédiat d'adapter la preuve du théorème 10.4.3 et de voir que les plus petites bornes supérieures sont les unions.

Enfin, de même qu'à la section 9.1.3, les flèches et les produits préservent les ensembles $\mathcal{SR}$-saturés.

Proposition 11.4.11 *Si* $A_1, A_2 \in \mathbb{SAT}_{\mathcal{CR}}$ *alors,* $A_2 \Rightarrow A_1, A_1 \times A_2 \in \mathbb{SAT}_{\mathcal{CR}}$.

Preuve. Comme pour la proposition 9.1.25. □

On peut maintenant définir l'interprétation des types $T \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$ par des ensembles $\mathcal{SR}$-saturés.

Définition 11.4.12 *Étant donné un assignement* $\mu : \mathcal{V}s \to \mathfrak{D}$*, et une interprétation des types de base* $[\![\_]\!] : \mathcal{B}_0 \to \mathfrak{D} \to \mathbb{SAT}_{\mathcal{CR}}$ *l'interprétation des types* $T \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$ *est définie inductivement de la manière suivante :*

$$\begin{array}{lll}
[\![B^a]\!]\mu & =_{def} & [\![B]\!](\mu(a)) \ , \\
[\![T \ \Xi \ U]\!]\mu & =_{def} & [\![T]\!]\mu \ \Xi \ [\![U]\!]\mu \qquad\qquad si \ \Xi \in \{\Rightarrow, \times\} \ , \\
[\![\forall \vec{\alpha} P.T]\!]\mu & =_{def} & \bigcap_{\vec{a} \in \{\vec{a} \ | \ \mu[\vec{a}/\vec{\alpha}] \models P\}} [\![T]\!]\mu[\vec{a}/\vec{\alpha}] \ , \\
[\![\exists \vec{\alpha} P.T]\!]\mu & =_{def} & \bigcup_{\vec{a} \in \{\vec{a} \ | \ \mu[\vec{a}/\vec{\alpha}] \models P\}} [\![T]\!]\mu[\vec{a}/\vec{\alpha}] \ ,
\end{array}$$

où $\bigcap \emptyset = \mathcal{SN}_{\mathcal{SR}}$ *et* $\bigcup \emptyset = \bigcap \mathbb{SAT}_{\mathcal{CR}}$.

Notons que pour tout $\vec{a} \in \mathcal{T}er(\mathfrak{S}, \mathcal{V}s)$, on a $[\![T[\vec{a}/\vec{\alpha}]]\!]\mu = [\![T]\!]\mu[\mu(\vec{a})/\vec{\alpha}]$. De plus, si $\vec{\alpha} \notin FV(T)$, alors $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] = [\![T]\!]\mu$ pour tout $\vec{a} \in \mathfrak{D}$. Lorsque que $FV_{\mathcal{S}}(T) = \emptyset$, on désigne $[\![T]\!]\mu$ par $[\![T]\!]$. Enfin, étant donné $C \in \mathcal{B}_0$, on écrit $[\![C]\!]$ pour désigner $[\![\exists \alpha.\ B^\alpha]\!]\mu$.

Remarque 11.4.13 *Ainsi, avec l'interprétation des types* $T \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ *issue de* $[\![B]\!] =_{def} [\![\exists \alpha.\ B^\alpha]\!]$*, pour tout type simple contraint* T *on a* $[\![T]\!]\mu = [\![\underline{T}]\!]$.

Lemme 11.4.14 *Pour tout assignement* $\mu : \mathcal{V}s \to \mathfrak{D}$ *et tout* $T \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$*, on a* $[\![T]\!]\mu \in \mathbb{SAT}_{\mathcal{CR}}$.

Preuve. Par induction sur $T \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$. □

11.4.1.(c) Candidats de réductibilité

Voyons maintenant comment prendre en compte les réductions $\to_{\mathcal{R}\mathsf{ite}}$. De même qu'à la section 11.4.1.(b), on suppose donnés un ensemble $\mathcal{B}_0$ de types de base, une signature (Σ_0, τ) typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ et un ensemble $\mathcal{C}$ de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$. On suppose de plus que $\mathcal{R}$ vérifie les conditions de la définition 11.1.1.

On a vu à la section 10.4.3 une condition suffisante pour avoir une famille de réductibilité stable par union : l'existence de réduits principaux (forts). Cependant, nous avons vu aussi à l'exemple 10.3.4 que cette propriété n'est pas assurée pour certains systèmes de réécriture confluents.

On revient donc aux propriétés que nous avons utilisées à la section 9.3.3 pour qu'une interprétation puisse prendre en compte une relation de réécriture comme $\to_{\mathcal{R}\mathsf{ite}}$.

Soit un type $T \in \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$ et un assignement $\mu : \mathcal{V}s \to \mathfrak{D}$. On voudrait que $[\![T]\!]\mu$ satisfasse les deux propriétés ci-dessus :

($\mathbb{SAT}0$) – ($\mathbb{CR}0$) si $t \in [\![T]\!]\mu$ et $t \to_{\mathcal{R}\mathsf{ite}} u$ alors $u \in [\![T]\!]\mu$,

($\mathbb{CR}1_{\mathcal{R}\mathsf{ite}}$) si t est un terme neutre tel que $(t)_{\mathcal{R}\mathsf{ite}} \subseteq [\![T]\!]\mu$, alors $t \in [\![T]\!]\mu$.

D'après ce que l'on a vu à la section 10.4.2, c'est la propriété $(\mathcal{CR}1_{\mathcal{R}\mathsf{ite}})$ qui pose problème vis-à-vis de la stabilité par union. En effet, si $(t)_{\mathcal{R}\mathsf{ite}} \subseteq [\![\exists\vec{\alpha}P.T]\!]\mu$, alors pour tout $u \in (t)_{\mathcal{R}\mathsf{ite}}$ il existe $\vec{a}_u$ tel que $\mu[\vec{a}_u/\vec{\alpha}] \models P$ et $u \in [\![T]\!]\mu[\vec{a}_u/\vec{\alpha}]$. Pour satisfaire $(\mathcal{CR}1_{\mathcal{R}\mathsf{ite}})$, il faut assurer qu'il existe $u \in (t)_{\mathcal{R}\mathsf{ite}}$ tel que $t \in [\![T]\!]\mu[\vec{a}_u/\vec{\alpha}]$.

Le problème est résolut si les ensembles $([\![T]\!]\mu[\vec{a}/\vec{\alpha}])_{\mu[\vec{a}/\vec{\alpha}]\models P}$ sont stables par expansion fortement $\mathcal{SR}$-normalisante :

$$(t \in \mathcal{SN}_{\mathcal{SR}} \quad \wedge \quad t \to_{\mathcal{R}\mathsf{ite}} u \in [\![T]\!]\mu[\vec{a}/\vec{\alpha}]) \quad \Longrightarrow \quad t \in [\![T]\!]\mu[\vec{a}/\vec{\alpha}] \ .$$

En effet, dans ce cas on a $t \in [\![T]\!][\vec{a}_u/\vec{\alpha}]$ pour tout $u \in (t)_R$, donc $t \in [\![\exists\vec{\alpha}P.T]\!]\mu$.

Définition 11.4.15 *Soit $\to_R$ une relation de réécriture. Un ensemble $C \subseteq \mathcal{SN}_R$ est* stable par expansion fortement R-normalisante *si pour tout $t \in \mathcal{SN}_R$,*

$$\forall u. \quad (t \to_R u \quad \wedge \quad u \in C) \quad \Longrightarrow \quad t \in C \ .$$

Cette propriété n'est pas, a priori, préservée par la flèche. En effet, soient C_1, C_2 deux parties de $\mathcal{SN}_R$ stables par expansion fortement R-normalisante. Si $t \to_R u \in C_2 \Rightarrow C_1$ avec $t \in \mathcal{SN}_R$, alors $tv \to_R uv \in C_1$ pour tout $v \in C_2$, mais rien n'assure que $tv \in \mathcal{SN}_R$. Par contre, elle est préservée par les produits.

Proposition 11.4.16 *Soit $\to_R$ une relation de réécriture et C_1, C_2 deux parties de $\mathcal{SN}_R$ stables par expansion fortement R-normalisante. Alors $C_1 \times C_2$ est une partie de $\mathcal{SN}_R$ stable par expansion fortement R-normalisante.*

Preuve. Rappelons que $C_1 \times C_2 =_{\text{def}} \{t \mid \pi_1 t \in C_1 \wedge \pi_2 t \in C_2\}$. Tout d'abord il est clair que $C_1 \times C_2 \subseteq \mathcal{SN}_R$. Soit $t \in \mathcal{SN}_R$ et $u \in (t)_R$ tel que $u \in C_1 \times C_2$. Alors pour tout $i \in \{1,2\}$, on a $\pi_i t \to_R \pi_i u \in C_i$, donc $\pi_i t \in C_i$ car $\pi_i t \in \mathcal{SN}_R$ et C_i est stable par expansion fortement R-normalisante. □

Ainsi, si les types de base sont interprétés par des candidats de réductibilité stables par expansion fortement $\mathcal{SR}$-normalisante, alors les types contraints basés sur les produits sont des candidats de réductibilité stables par expansion fortement $\mathcal{SR}$-normalisante.

Rappelons que notre notion de terme neutre dépend des destructeurs des types inductifs (voir section 11.4.1.(a)). Ils apparaissent donc dans la définition des candidats de réductibilité. De plus, les contextes d'élimination $\mathcal{E}_{\mathcal{C}}$ sont définis en 11.4.2.

Définition 11.4.17 *Étant donnés un ensemble $\mathcal{B}_0$ de types de base, une signature (Σ_0, τ) typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$, un ensemble $\mathcal{C}$ de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$, on désigne par $\mathcal{CR}_{\mathcal{CR}}$ l'ensemble des candidats de réductibilité pour $\to_{\mathcal{SRD}}$ dans les contextes $\mathcal{E}_{\mathcal{C}}$, au sens de la définition 9.3.16.*

Rappelons que l'ensemble $\mathcal{N}_{\mathcal{CR}}$ des termes neutres pour ces candidats de réductibilité est défini en 11.4.4.

Lemme 11.4.18 *Supposons que pour tout $B \in \mathcal{B}_0$ et tout $a \in \mathfrak{D}$, $[\![B]\!](a)$ soit un candidat de réductibilité stable par expansion fortement $\mathcal{SR}$-normalisante. Alors pour tout type contraint basé sur les produits $T \in \mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$ et tout assignement $\mu : \mathcal{V}s \to \mathfrak{D}$, $[\![T]\!]\mu$ est un candidat de réductibilité stable par expansion fortement $\mathcal{SR}$-normalisante.*

PREUVE. On commence par le montrer pour les types annotés basés sur les produits $\mathsf{T} \in \mathcal{T}_{\times}(\mathcal{B}_0, \mathfrak{S}, \mathcal{V}\mathfrak{s})$. Le cas des types de base suit de l'hypothèse, et le cas des types produits suit des propositions 9.3.31.(ii) et 11.4.16.

Considérons maintenant les types $\mathsf{T} \in \mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$. Le cas des types annotés basés sur les produits vient d'être traité.

Supposons que $\mathsf{T} = \exists\vec{\alpha}\mathsf{P}.\mathsf{U}$. Il est clair que $[\![\mathsf{T}]\!]\mu$ est un ensemble de termes fortement $\mathcal{SR}$-normalisants et clos par réduction. Soit $\mathfrak{t}$ un terme fortement $\mathcal{SR}$-normalisant et $\mathfrak{u} \in (\mathfrak{t})_R$ tel que $\mathfrak{u} \in [\![\mathsf{T}]\!]\mu$. Alors il existe $\vec{\mathfrak{a}}$ tel que $\mu[\vec{\mathfrak{a}}/\vec{\alpha}] \models \mathsf{P}$ et $\mathfrak{u} \in [\![\mathsf{U}]\!]\mu[\vec{\mathfrak{a}}/\vec{\alpha}]$. On en déduit que $\mathfrak{t} \in [\![\mathsf{U}]\!]\mu[\vec{\mathfrak{a}}/\vec{\alpha}]$ car $[\![\mathsf{U}]\!]\mu[\vec{\mathfrak{a}}/\vec{\alpha}]$ est stable par expansion fortement $\mathcal{SR}$-normalisante par hypothèse d'induction.

Il s'en suit que $[\![\mathsf{T}]\!]\mu$ est clos par expansion fortement $\mathcal{SR}$-normalisante et qu'il satisfait la clause ($\mathcal{CR}1$).

Il reste à traiter le cas de $\mathsf{T} = \forall\vec{\alpha}\mathsf{P}.\mathsf{U}$. Le fait que $[\![\mathsf{T}]\!]\mu \in \mathcal{CR}_{\mathcal{CR}}$ provient du lemme 9.3.24 et de la propriété 9.3.3. La stabilité par expansion fortement normalisante se traite de manière similaire au cas $\mathsf{T} = \exists\vec{\alpha}\mathsf{P}.\mathsf{U}$. □

Notons que les candidats de réductibilité sont des ensembles $\mathcal{SR}$-saturés : la clause ($\mathcal{SAT}0$) est triviale et les clauses ($\mathcal{SAT}1_{\mathcal{S}}$), ($\mathcal{SAT}2_{\mathcal{S}}$) suivent du lemme 9.3.18.

Définition 11.4.19 *Un interprétation des types de base* $[\![\_]\!] : \mathcal{B}_0 \to \mathfrak{D} \to \mathcal{SAT}_{\mathcal{SR}}$ *est dite* valide *si pour tout* $\mathsf{B} \in \mathcal{B}_0$ *et tout* $\mathfrak{a} \in \mathfrak{D}$, $[\![\mathsf{B}]\!](\mathfrak{a})$ *est un candidat de réductibilité stable par expansion fortement* $\mathcal{SR}$*-normalisante.*

Ainsi, si les types de base sont interprétés par une interprétation valide, alors les types contraints basés sur les produits sont interprétés par des candidats de réductibilité.

Remarque 11.4.20 *De ce fait, si le type de sortie des fonctions est un type contraint basé sur les produits, on évite le problème, étudié au chapitre 10, des sémantiques de normalisation stables par union.*

Notons que c'est ce qui est prescrit pour le typage du symbole $\mathsf{ite}(\_,\_,\_)$ *:*

$$(\mathsf{Bool})\ \frac{\mathsf{C};\Gamma \vdash \mathsf{b} : \mathsf{Bool} \qquad \mathsf{C};\Gamma \vdash \mathsf{t}_1 : \mathsf{T} \qquad \mathsf{C};\Gamma \vdash \mathsf{t}_2 : \mathsf{T}}{\mathsf{C};\Gamma \vdash \mathsf{ite}(\mathsf{b},\mathsf{t}_1,\mathsf{t}_2) : \mathsf{T}}\ (\mathsf{T} \in \mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s})))$$

11.4.2 Adéquation

On montre maintenant que l'interprétation des types que l'on vient de définir est adéquate vis-à-vis du typage dans $\lambda_{\mathcal{SR}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}), \Sigma_0, \tau)$.

On commence par montrer que l'interprétation des types respecte les contraintes de sous-typage.

Proposition 11.4.21 *Soient* $X_1, X_2, Y_1, Y_2 \subseteq \Lambda(\Sigma)$ *tels que* $X_1 \subseteq Y_1$ *et* $X_2 \subseteq Y_2$*. Alors* $X_1 \times X_2 \subseteq Y_1 \times Y_2$*.*

PREUVE. Si $\mathfrak{t} \in X_1 \times X_2$, alors pour tout $i \in \{1,2\}$ on a $\pi_i\mathfrak{t} \in X_i \subseteq Y_i$, donc $\mathfrak{t} \in Y_1 \times Y_2$. □

Proposition 11.4.22 *Si* $\mu \models \|\mathsf{U} \sqsubseteq \mathsf{T}\|$ *alors* $[\![\mathsf{U}]\!]\mu \subseteq [\![\mathsf{T}]\!]\mu$*.*

PREUVE. Par induction sur la définition de $\|U \sqsubseteq T\|$.

$\|B^a \sqsubseteq B^b\|$. Dans ce cas $\mu \models a = b$, donc $[\![B^a]\!]\mu = [\![B^b]\!]\mu$.

$\|T_2 \Rightarrow T_1 \sqsubseteq U_2 \Rightarrow U_2\|$. On a $\mu \models \|U_2 \sqsubseteq T_2\|$ et $\mu \models \|T_1 \sqsubseteq U_1\|$, donc par hypothèse d'induction, $[\![U_2]\!]\mu \subseteq [\![T_2]\!]\mu$ et $[\![T_1]\!]\mu \subseteq [\![U_1]\!]\mu$, soit $[\![T_2 \Rightarrow T_1]\!]\mu \subseteq [\![U_2 \Rightarrow U_1]\!]\mu$ par la proposition 10.2.4.

$\|T_1 \times T_2 \sqsubseteq U_1 \times U_2\|$. On a $\mu \models \|T_1 \sqsubseteq U_1\|$ et $\mu \models \|T_2 \sqsubseteq U_2\|$, donc par hypothèse d'induction, $[\![T_1]\!]\mu \subseteq [\![U_1]\!]\mu$ et $[\![T_2]\!]\mu \subseteq [\![U_2]\!]\mu$, soit $[\![T_1 \times T_2]\!]\mu \subseteq [\![U_1 \times U_2]\!]\mu$ par la proposition 11.4.21.

$\|T \sqsubseteq \exists \vec{\alpha} P.U\|$ avec $T \neq \exists \vec{\beta} Q.V$. Comme $\mu \models \exists \vec{\alpha}.\, P \wedge \|T \sqsubseteq U\|$, par hypothèse d'induction il existe $\vec{a} \in \mathfrak{D}$ tel que $\mu[\vec{a}/\vec{\alpha}] \models P$ et $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$.

Il s'en suit que $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![\exists \vec{\alpha} P.U]\!]\mu$, soit $[\![T]\!]\mu \subseteq [\![\exists \vec{\alpha} P.U]\!]\mu$ car $\vec{\alpha} \notin FV_{\mathcal{S}}(T)$.

$\|\exists \vec{\alpha} P.T \sqsubseteq U\|$. On doit montrer que $\bigcup_{\vec{a} \in \{\vec{a} \mid \mu[\vec{a}/\vec{\alpha}] \models P\}} [\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu$. S'il n'existe pas de $\vec{a} \in \mathfrak{D}$ tels que $\mu[\vec{a}/\vec{\alpha}] \models P$, alors $[\![\exists \vec{\alpha} P.T]\!]\mu = \bigcap \mathcal{SAT}_{\mathcal{SR}}$, donc $[\![\exists \vec{\alpha} P.T]\!]\mu \subseteq [\![U]\!]\mu$ car $[\![U]\!]\mu \in \mathcal{SAT}_{\mathcal{SR}}$ par le lemme 11.4.14.

Sinon, comme $\mu \models \forall \vec{\alpha}.P \supset \|T \sqsubseteq U\|$, pour tout $\vec{a} \in \mathfrak{D}$ tels que $\mu[\vec{a}/\vec{\alpha}] \models P$, par hypothèse d'induction on a $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$, soit $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu$ car $\vec{\alpha} \notin FV(U)$. Il s'en suit que $\bigcup_{\vec{a} \in \{\vec{a} \mid \mu[\vec{a}/\vec{\alpha}] \models P\}} [\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu$.

$\|T \sqsubseteq \forall \vec{\alpha} P.U\|$. On doit montrer que $[\![T]\!]\mu \subseteq \bigcap_{\vec{a} \in \{\vec{a} \mid \mu[\vec{a}/\vec{\alpha}] \models P\}} [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$. S'il n'existe pas de $\vec{a} \in \mathfrak{D}$ tels que $\mu[\vec{a}/\vec{\alpha}] \models P$, alors $[\![\forall \vec{\alpha} P.U]\!]\mu = \bigcap \mathcal{SN}_{\mathcal{SR}}$, donc $[\![T]\!]\mu \subseteq [\![\forall \vec{\alpha} P.U]\!]\mu$ car $[\![T]\!]\mu \in \mathcal{SAT}_{\mathcal{SR}}$ par le lemme 11.4.14.

Sinon, comme $\mu \models \forall \vec{\alpha}.P \supset \|T \sqsubseteq U\|$, pour tout $\vec{a} \in \mathfrak{D}$ tels que $\mu[\vec{a}/\vec{\alpha}] \models P$, par hypothèse d'induction on a $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$, soit $[\![T]\!]\mu \subseteq [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$ car $\vec{\alpha} \notin FV(T)$. Il s'en suit que $[\![T]\!]\mu \subseteq \bigcap_{\vec{a} \in \{\vec{a} \mid \mu[\vec{a}/\vec{\alpha}] \models P\}} [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$.

$\|\forall \vec{\alpha} P.T \sqsubseteq U\|$ avec $U \neq \forall \vec{\beta} Q.V$. Comme $\mu \models \exists \vec{\alpha}.\, P \wedge \|T \sqsubseteq U\|$, par hypothèse d'induction il existe $\vec{a} \in \mathfrak{D}$ tel que $\mu[\vec{a}/\vec{\alpha}] \models P$ et $[\![T]\!]\mu[\vec{a}/\vec{\alpha}] \subseteq [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$.

Il s'en suit que $[\![\forall \vec{\alpha} P.T]\!]\mu \subseteq [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$, soit $[\![\forall \vec{\alpha} P.T]\!]\mu \subseteq [\![U]\!]\mu$ car $\vec{\alpha} \notin FV_{\mathcal{S}}(U)$. □

On peut maintenant montrer que si l'on dispose d'une interprétation valide, et que tous les symboles $f \in \Sigma_0$ appartiennent à l'interprétation de leur type, alors tous les termes typables appartiennent à l'interprétation de leur type.

Définition 11.4.23 *Étant donnés une substitution σ, un assignement μ et un contexte contraint $C; \Gamma$, on pose $(\mu, \sigma) \models C; \Gamma$, si $\mu \models C$ et $\sigma(x) \in [\![\Gamma(x)]\!]\mu$ pour tout $x \in \mathcal{D}om(\Gamma)$.*

Théorème 11.4.24 (Adéquation) *Soit $[\![\_]\!]$ une interprétation valide des types de base. Supposons que $f \in [\![\tau_{\mathcal{S}}(f)]\!]$ pour tout $f \in \Sigma_0$. Si $C, \Gamma \vdash t : T$ et $(\mu, \sigma) \models C; \Gamma$ alors $t\sigma \in [\![T]\!]\mu$.*

PREUVE. Par induction sur $C; \Gamma \vdash t : T$.

Le cas de (AX) est trivial, celui de (SYMB I) pris en compte par l'hypothèse. Comme les types sont interprétés par des ensembles saturés, les cas de ($\Rightarrow$ I), ($\Rightarrow$ E), ($\times$I) et ($\times$E) sont traités comme au lemme 9.1.26, en utilisant la remarque 11.4.10. Enfin, la règle (SUB) est traitée par la proposition 11.4.22.

Voyons le cas des règles restantes :

(Bool) Soit $(\mu, \sigma) \models C; \Gamma$. Par hypothèse d'induction, on a $b\sigma \in [\![\exists \alpha.\mathsf{Bool}^\alpha]\!]$ et $t_i\sigma \in [\![T]\!]\mu$ pour tout $i \in \{1, 2\}$.
On doit montrer que $\mathsf{ite}(b\sigma, t_1\sigma, t_2\sigma) \in [\![T]\!]\mu$. Pour cela, on montre par induction sur (c, u_1, u_2) ordonné par $\to_{\mathcal{SR}}$ que si $c \in [\![\exists \alpha.\mathsf{Bool}^\alpha]\!]$ et $u_1, u_2 \in [\![T]\!]\mu$ alors $\mathsf{ite}(c, u_1, u_2) \in [\![T]\!]\mu$.
Soit donc $c \in [\![\exists \alpha.\mathsf{Bool}^\alpha]\!]$ et $u_1, u_2 \in [\![T]\!]\mu$.
Comme $T \in \mathcal{T}_{\exists\forall\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$, on a $[\![T]\!]\mu \in \mathcal{CR}_{\mathcal{CR}}$ par le lemme 11.4.18, donc comme $\mathsf{ite}(c, u_1, u_2)$ est neutre, il suffit de montrer que $(\mathsf{ite}(c, u_1, u_2))_{\mathcal{SR}} \subseteq [\![T]\!]\mu$.
Soit $\nu \in (\mathsf{ite}(c, u_1, u_2))_{\mathcal{SR}}$.
— Si $\nu = \mathsf{ite}(c', u_1', u_2')$ avec $(c, u_1, u_2) \to_{\mathcal{SR}} (c', u_1', u_2')$, alors $c' \in [\![\exists \alpha.\mathsf{Bool}^\alpha]\!]$ et $u_1', u_2' \in [\![T]\!]\mu$ par ($\mathfrak{SAT}0$), et on a $\nu \in [\![T]\!]\mu$ par hypothèse d'induction.
— Sinon, $c \in \{\mathsf{true}, \mathsf{false}\}$ et $\nu = u_i \in [\![T]\!]\mu$ par hypothèse.

(let) Soit $(\mu, \sigma) \models C; \Gamma$. On doit montrer que $(\mathsf{let}\ x = t\ \mathsf{in}\ u)\sigma \in [\![U]\!]\mu$.
Par le lemme 1.2.10, on peut supposer que $x \notin \mathsf{FV}(\sigma) \cup \mathcal{D}\mathrm{om}(\sigma)$. On a donc

$$(\mathsf{let}\ x = t\ \mathsf{in}\ u)\sigma \quad = \quad \mathsf{let}\ x = t\sigma\ \mathsf{in}\ u\sigma$$

selon le lemme 1.3.9.(i).
D'après la remarque 11.4.10, comme $t\sigma \in [\![T]\!]\mu \subseteq \mathcal{SN}_{\mathcal{SR}}$ par hypothèse d'induction, il suffit de montrer que $(u\sigma)[t\sigma/x] \in [\![U]\!]\mu$.
Or, $(u\sigma)[t\sigma/x] = u(\sigma[t\sigma/x])$ selon la proposition 1.3.5, et par hypothèse d'induction on a $u(\sigma[t\sigma/x]) \in [\![U]\!]\mu$ car $t\sigma \in [\![T]\!]\mu$. Il s'en suit que $(u\sigma)[t\sigma/x] \in [\![U]\!]\mu$.

($\forall$ I) Soit $(\mu, \sigma) \models \Gamma$. On doit montrer que $t\sigma \in [\![T]\!]\mu[\vec{a}/\vec{\alpha}]$ pour tout $\vec{a}$ tel que $\mu[\vec{a}/\vec{\alpha}] \models P$.
Or, si $\mu[\vec{a}/\vec{\alpha}] \models P$, alors $(\mu[\vec{a}/\vec{\alpha}], \sigma) \models C \wedge P; \Gamma$ car $\vec{\alpha} \notin \mathsf{FV}(C, \Gamma)$, donc par hypothèse d'induction $t\sigma \in [\![T]\!]\mu[\vec{a}/\vec{\alpha}]$.

($\forall$ E) Soit $(\mu, \sigma) \models C; \Gamma$. Par hypothèse d'induction, $t\sigma \in [\![T]\!]\mu[\vec{a}/\vec{\alpha}]$ pour tout $\vec{a}$ tel que $\mu[\vec{a}/\vec{\alpha}] \models P$. Comme $\mu \models C$ et $\vdash C \supset P[\vec{a}/\vec{\alpha}]$, on a $\mu \models P[\vec{a}/\vec{\alpha}]$, donc $\mu[\mu(\vec{a})/\vec{\alpha}] \models P$. Il s'en suit que $t\sigma \in [\![T]\!]\mu[\mu(\vec{a})/\vec{\alpha}] = [\![T[\vec{a}/\vec{\alpha}]]\!]\mu$.

($\exists$ I) Soit $(\mu, \sigma) \models C; \Gamma$. Par hypothèse d'induction, on a $t\sigma \in [\![T[\vec{a}/\vec{\alpha}]]\!]\mu = [\![T]\!]\mu[\mu(\vec{a})/\vec{\alpha}]$.
Comme $\mu \models C$ et $\vdash C \supset P[\vec{a}/\vec{\alpha}]$, on a $\mu \models P[\vec{a}/\vec{\alpha}]$, d'où $\mu[\mu(\vec{a})/\vec{\alpha}] \models P$, et donc $t\sigma \in [\![\exists \vec{\alpha} P.T]\!]\mu$.

($\exists$ E) Soit $(\mu, \sigma) \models C; \Gamma$. On doit montrer que $(\mathsf{let}\ x = t\ \mathsf{in}\ u)\sigma$. En raisonnant comme dans le cas de la règle (let), on peut supposer que $x \notin \mathsf{FV}(\sigma) \cup \mathcal{D}\mathrm{om}(\sigma)$ et il suffit de montrer que $u(\sigma[t\sigma/x]) \in [\![U]\!]\mu$.
Du fait que $\mu \models C$ et que $\vdash C \supset \exists \vec{\alpha}.P$, il existe $\vec{a} \in \mathfrak{D}$ tel que $\mu[\vec{a}/\vec{\alpha}] \models P$. Il s'en suit que $t\sigma \in [\![T]\!]\mu[\vec{a}/\vec{\alpha}]$ par hypothèse d'induction.
D'autre part, comme $\vec{\alpha} \notin \mathsf{FV}(C)$, on a $\mu[\vec{a}/\vec{\alpha}] \models C$, et comme $\vec{\alpha} \notin \mathsf{FV}(\Gamma)$, on en déduit que $(\mu[\vec{a}/\vec{\alpha}], \sigma[t\sigma/x]) \models C \wedge P; \Gamma, x : T$. Ainsi, on a $u(\sigma[t\sigma/x]) \in [\![U]\!]\mu[\vec{a}/\vec{\alpha}]$ par hypothèse d'induction, soit $u(\sigma[t\sigma/x]) \in [\![U]\!]\mu$ car $\vec{\alpha} \notin \mathsf{FV}(U)$. □

Corollaire 11.4.25 *Soit* $[\![\_]\!]$ *une interprétation valide des types de base et supposons que* $f \in [\![\tau_{\mathcal{S}}(f)]\!]$ *pour tout* $f \in \Sigma_0$.
Si $C; \Gamma \vdash t : T$ *et s'il existe un assignement* μ *tel que* $\mu \models C$, *alors* $t \in \mathcal{SN}_{\mathcal{SR}}$.

11.4.3 Interprétation singleton des types inductifs

Nous définissons maintenant une interprétation valide des types de base. Pour cela on suppose que tous les types de base sont des types inductifs du premier ordre, dans le sens de la définition 11.2.5. Comme suggéré à la remarque 11.2.13, et d'après nos contraintes de sous-typage (voir exemple 11.3.13), on veut une interprétation « singleton », c'est-à-dire telle que $[\![\mathsf{B}]\!](\mathfrak{a}) = [\![\mathsf{B}]\!](\mathfrak{b})$ si et seulement si $\mathfrak{a} = \mathfrak{b}$. De plus, on veut que les types inductifs soient interprétés par des candidats de réductibilité stables par expansion fortement $\mathcal{SR}$-normalisante, au sens des définitions 11.4.15 et 11.4.17. Pour cela, on suppose que la relation $\rightarrow_{\mathcal{SR}}$ est confluente sur $\mathcal{SN}_{\mathcal{SR}}$.

On commence par voir quels sont les types contraints que l'on assigne aux constructeurs de types inductifs du premier ordre. Soit C un type inductif et $\mathfrak{C}$ sa classe d'équivalence pour $\simeq_{\mathcal{B}_0}$. Rappelons que C est un type inductif du premier ordre si pour tout constructeur c de C on a

$$\tau(\mathsf{c}) \quad = \quad \mathsf{D}_1 \Rightarrow \cdots \Rightarrow \mathsf{D}_k \Rightarrow \mathsf{C}_{k+1} \Rightarrow \cdots \Rightarrow \mathsf{C}_n \Rightarrow \mathsf{C}\ ,$$

où $\mathsf{D}_i \in <^{\mathfrak{C}}_{\mathcal{B}_0}$ pour tout $i \in \{1, \ldots, k\}$ et $\mathsf{C}_i \in \mathfrak{C}$ pour tout $i \in \{k+1, \ldots, n\}$.

Remarque 11.4.26 *Cet ordre sur les arguments des constructeurs n'a rien d'essentiel. Il nous est utile uniquement pour simplifier la présentation.*

Rappelons que les types inductifs sur les booléens sont définis en 11.4.1.

Définition 11.4.27 (Typage des constructeurs) *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ et $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$.*

— *On type dans $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}}, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ les constructeurs* true *et* false *du type* Bool *de la manière suivante :*

$$\tau_{\mathcal{S}}(\mathsf{true}) \quad =_{def} \quad \mathsf{Bool}^{\mathsf{t}} \qquad \textit{et} \qquad \tau_{\mathcal{S}}(\mathsf{false}) \quad =_{def} \quad \mathsf{Bool}^{\mathsf{f}}\ .$$

— *Soit $\mathsf{C} \in \mathcal{C}$ et c un constructeur de C de type*

$$\tau(\mathsf{c}) \quad = \quad \mathsf{D}_1 \Rightarrow \cdots \Rightarrow \mathsf{D}_k \Rightarrow \mathsf{C}_{k+1} \Rightarrow \cdots \Rightarrow \mathsf{C}_n \Rightarrow \mathsf{C}\ ,$$

où $\mathsf{D}_i <_{\mathcal{B}_0} \mathsf{C}$ pour tout $i \in \{1, \ldots, k\}$ et $\mathsf{C}_i \simeq_{\mathcal{B}_0} \mathsf{C}$ pour tout $i \in \{k+1, \ldots, n\}$. On pose

$$\tau_{\mathcal{S}}(\mathsf{c}) \quad =_{def} \quad \mathsf{D}_1 \Rightarrow \cdots \Rightarrow \mathsf{D}_n \Rightarrow \mathsf{C}^0$$

si $k = n$, et sinon,

$$\tau_{\mathcal{S}}(\mathsf{c}) \ =_{def}\ \mathsf{D}_1 \Rightarrow \cdots \Rightarrow \mathsf{D}_k \Rightarrow \forall \alpha_{k+1} \ldots \alpha_n.\ \mathsf{C}_{k+1}^{\alpha_{k+1}} \Rightarrow \cdots \Rightarrow \mathsf{C}_n^{\alpha_n} \Rightarrow \mathsf{C}^{1+\max(\vec{\alpha})}\ .$$

Exemple 11.4.28 *Les types des constructeurs de* Nat *et* List *donné à l'exemple 11.3.8 sont les types spécifiés à la définition 11.4.27 :*

$$\begin{array}{llll} \tau_{\mathcal{S}}(0) & =_{def} & \mathsf{Nat}^0 & \qquad \tau_{\mathcal{S}}(\mathsf{S}) \quad =_{def} \quad \forall\alpha.\ \mathsf{Nat}^{\alpha} \Rightarrow \mathsf{Nat}^{\alpha+1} \\ \tau_{\mathcal{S}}(\mathsf{nil}) & =_{def} & \mathsf{List}^0 & \qquad \tau_{\mathcal{S}}(\mathsf{cons}) \quad =_{def} \quad \mathsf{Nat} \Rightarrow \forall\alpha.\ \mathsf{List}^{\alpha} \Rightarrow \mathsf{List}^{\alpha+1}\ . \end{array}$$

Voyons maintenant comment définir une interprétation singleton pour les types inductifs du premier ordre.

Définition 11.4.29 (Interprétation singleton) *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ et $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$.*

— *On définit l'interprétation du type* Bool *des booléens de la manière suivante :*

$$\begin{array}{lll} [\![\mathsf{Bool}]\!](t) & =_{def} & \{t \in \mathcal{SN}_{\mathcal{SR}} \mid t \not\to^*_{\mathcal{SR}} \mathsf{false}\}\,, \\ [\![\mathsf{Bool}]\!](f) & =_{def} & \{t \in \mathcal{SN}_{\mathcal{SR}} \mid t \not\to^*_{\mathcal{SR}} \mathsf{true}\}\,. \end{array}$$

— *Si $C \in \mathcal{C}$ alors on définit $[\![C]\!](a)$ par induction sur $a \in \mathbb{N}$ comme suit :*

(i) $[\![C]\!](0)$ *est l'ensemble des $t \in \mathcal{SN}_{\mathcal{SR}}$ tels que pour tout $c \in C$ et tout $t_1, \ldots, t_n$ avec $n = a_c$, si $t \to^*_{\mathcal{SR}} c\,t_1 \ldots t_n$ alors $\tau_{\mathcal{S}}(c) = D_1 \Rightarrow \cdots \Rightarrow D_n \Rightarrow C^0$ avec $t_i \in [\![D_i]\!]$ pour tout $i \in \{1, \ldots, n\}$,*

(ii) $[\![C]\!](b+1)$ *est l'ensemble des $t \in \mathcal{SN}_{\mathcal{SR}}$ tels que pour tout $c \in C$ et tout $t_1, \ldots, t_n$ avec $n = a_c$, si $t \to^*_{\mathcal{SR}} c\,t_1 \ldots t_n$ alors il existe $k < n$ tel que*

$$\tau_{\mathcal{S}}(c) = D_1 \Rightarrow \cdots \Rightarrow D_k \Rightarrow \forall \alpha_{k+1} \ldots \alpha_n.\ C_{k+1}^{\alpha_{k+1}} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow C^{1+\max(\vec{\alpha})}$$

et
- *$t_i \in [\![D_i]\!]$ pour tout $i \in \{1, \ldots, k\}$,*
- *il existe $\vec{a}$ tels que $b = \max(\vec{a})$ et $t_j \in [\![C_j]\!](a_j)$ pour tout $j \in \{k+1, \ldots, n\}$.*

On vérifie maintenant que l'on a bien définit une interprétation valide au sens de la définition 11.4.19.

Lemme 11.4.30 *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ et $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ tels que tout $C \in \mathcal{B}_0$ appartient à $\mathcal{C}$. Soit de plus $\mathcal{R}$ un système de réécriture conditionnelle vérifiant les conditions de la définition 11.1.1 et tel que la relation $\to_{\mathcal{SR}}$ soit confluente sur $\mathcal{SN}_{\mathcal{SR}}$.*

Alors l'interprétation $[\![\_]\!] : \mathcal{B}_{0\mathsf{Bool}} \to \mathfrak{D} \to \mathcal{SAT}_{\mathcal{CR}}$ définie en 11.4.29 est une interprétation valide des types de base.

Preuve. On commence par le cas des booléens. On ne traite que le cas de $[\![\mathsf{Bool}]\!](t)$, celui de $[\![\mathsf{Bool}]\!](f)$ étant similaire.

Tout d'abord, il est clair que $[\![\mathsf{Bool}]\!](t) \subseteq \mathcal{SN}_{\mathcal{SR}}$. Vérifions maintenant que $[\![\mathsf{Bool}]\!](t)$ est bien un candidat de réductibilité.

($\mathcal{CR}0$) Si $t \in [\![\mathsf{Bool}]\!](t)$ et $t \to_{\mathcal{SR}} u$, alors $u \not\to^*_{\mathcal{SR}} \mathsf{false}$ car $t \not\to^*_{\mathcal{SR}} \mathsf{false}$.

($\mathcal{CR}1$) Soit t un terme neutre tel que $(t)_{\mathcal{SR}} \subseteq [\![\mathsf{Bool}]\!](t)$. Donc $t \neq \mathsf{false}$, et t ne se réduit pas vers false car aucun $u \in (t)_{\mathcal{SR}}$ ne se réduit vers false. Il s'en suit que $t \in [\![\mathsf{Bool}]\!](t)$.

Il reste à voir que $[\![\mathsf{Bool}]\!](t)$ est stable par expansion fortement $\mathcal{SR}$-normalisante. Soit $t \in [\![\mathsf{Bool}]\!](t)$ et $u \in \mathcal{SN}_{\mathcal{SR}}$ tel que $u \to^*_{\mathcal{SR}} t$. Alors si $u \to^*_{\mathcal{SR}} \mathsf{false}$, comme false est un terme $\mathcal{SR}$-normal, par confluence de $\to_{\mathcal{SR}}$ sur $\mathcal{SN}_{\mathcal{SR}}$ on a $t \to^*_{\mathcal{SR}} \mathsf{false}$, ce qui est impossible. Il s'en suit que $u \in [\![\mathsf{Bool}]\!](t)$.

Abordons maintenant le cas des types de base $C \in \mathcal{B}_0$. On raisonne par induction sur les paires (C, a) ordonnées par $(>_{\mathcal{B}_0}, >)_{\text{lex}}$, où $>$ est l'ordre standard sur $\mathbb{N}$. Soit donc $C \in \mathcal{B}_0$. On ne traite que le cas de $[\![C]\!](a+1)$, celui de $[\![C]\!](0)$ étant similaire et plus simple.

On a $[\![C]\!](a+1) \subseteq \mathcal{SN}_{\mathcal{SR}}$ par définition. Considérons les clauses ($\mathcal{CR}0$) et ($\mathcal{CR}1$).

($\mathcal{CR}0$) Si $t \in [\![C]\!](a+1)$ et $t \to_{\mathcal{SR}} u$, alors tout réduit de u de la forme $c\vec{t}$ avec $c \in C$ est un réduit de t, donc satisfait les conditions de la définition 11.4.29. Il s'en suit que $u \in [\![C]\!](a+1)$

($\mathcal{CR}1$) Soit t un terme neutre tel que $(t)_{\mathcal{SR}} \subseteq [\![C]\!](a+1)$. Alors comme t est neutre, ce n'est pas un terme de la forme $c\vec{t}$, donc tout réduit de t de cette forme est un réduit d'un $u \in (t)_{\mathcal{SR}}$. Il s'en suit que tout réduit de t de la forme $c\vec{t}$ satisfait les conditions de la définition 11.4.29, et donc que $t \in [\![C]\!](a+1)$.

Montrons maintenant que $[\![C]\!](a+1)$ est stable par expansion fortement normalisante. Soit $t \to_{\mathcal{SR}} u$ avec $u \in [\![C]\!](a+1)$ et $t \in \mathcal{SN}_{\mathcal{SR}}$. Supposons que $t \to^*_{\mathcal{SR}} ct_1 \dots t_k t_{k+1} \dots t_n$ avec

$$\tau_{\mathcal{S}}(c) = D_1 \Rightarrow \cdots \Rightarrow D_k \Rightarrow \forall \vec{\alpha}.\ C_{k+1}^{\alpha_{k+1}} \Rightarrow \dots C_n^{\alpha_n} \Rightarrow C^{\max(\vec{\alpha})+1}\ .$$

Alors par confluence de $\to_{\mathcal{SR}}$ sur $\mathcal{SN}_{\mathcal{SR}}$, il existe $u_1 \dots u_n$ tels que $(t_1, \dots, t_n) \to^*_{\mathcal{SR}} (u_1, \dots, u_n)$ et $u \to^*_{\mathcal{SR}} cu_1 \dots u_n$. Par définition, on a $u_i \in [\![D_i]\!]$ pour tout $i \in \{1, \dots, k\}$, et il existe $a_{k+1}, \dots, a_n$ tels que $a = \max(\vec{a})$ et et $u_j \in [\![C_j]\!](a_j)$ pour tout $j \in \{k+1, \dots, n\}$.

Par hypothèse d'induction, pour tout $i \in \{1, \dots, k\}$ et tout b, $[\![D_i]\!](b)$ est un candidat de réductibilité stable par expansion fortement normalisante, donc $[\![D_i]\!]$ est un candidat de réductibilité stable par expansion fortement normalisante par le lemme 11.4.18, et il s'en suit que $t_i \in [\![D_i]\!]$.

De même, par hypothèse d'induction, pour tout $j \in \{k+1, \dots, n\}$, $[\![C_j]\!](a_j)$ est stable par expansion fortement normalisante, donc $t_j \in [\![C_j]\!](a_j)$.

On en déduit que $t \in [\![C]\!](a+1)$. □

Vérifions que les constructeurs appartiennent à l'interprétation de leur type.

Lemme 11.4.31 *Soit $\mathcal{B}_0$ un ensemble de types de base, (Σ_0, τ) une signature typée sur $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_{0\mathsf{Bool}})$ et $\mathcal{C}$ un ensemble de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ tels que tout $C \in \mathcal{B}_0$ appartient à $\mathcal{C}$. Soit de plus $\mathcal{R}$ un système de réécriture conditionnelle vérifiant les conditions de la définition 11.1.1 et tel que la relation $\to_{\mathcal{SR}}$ soit confluente sur $\mathcal{SN}_{\mathcal{SR}}$.*

Considérons un type de base $C \in \mathcal{B}_0$, un constructeur c de C et des termes $t_1, \dots, t_n$ avec $n = a_c$.

(i) On a $ct_1 \dots t_n \in [\![C]\!](0)$ si et seulement si $\tau_{\mathcal{S}}(c) = D_1 \Rightarrow \cdots \Rightarrow D_n \Rightarrow C^0$ et $t_i \in [\![D_i]\!]$ pour tout $i \in \{1, \dots, n\}$.

(ii) Pour tout $b \in \mathbb{N}$*, on a* $ct_1 \dots t_n \in [\![C]\!](b+1)$ *si et seulement s'il existe* $k < n$ *tel que*

$$\tau_{\mathcal{S}}(c) \quad = \quad D_1 \Rightarrow \cdots \Rightarrow D_k \Rightarrow \forall \alpha_{k+1} \dots \alpha_n.\ C_{k+1}^{\alpha_{k+1}} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow C^{1+\max(\vec{\alpha})}$$

et
— $t_i \in [\![D_i]\!]$ *pour tout* $i \in \{1, \dots, k\}$,
— *il existe* $\vec{a}$ *tels que* $b = \max(\vec{a})$ *et* $t_j \in [\![C_j]\!](a_j)$ *pour tout* $j \in \{k+1, \dots, n\}$.

Preuve. Dans les deux cas, le sens « seulement si » suit directement de la définition 11.4.29.

Pour l'autre direction, soient $c \in C$ et $t_1, \dots, t_n$ avec $n = a_c$. Tout d'abord, notons que si $ct_1 \dots t_n \to_{\mathcal{SR}} v$ alors $v = ct'_1 \dots t'_n$ avec $(t_1, \dots, t_n) \to_{\mathcal{SR}} (t'_1, \dots, t'_n)$. Il s'en suit que $ct_1 \dots t_n \in \mathcal{SN}_{\mathcal{SR}}$ lorsque $t_1, \dots, t_n \in \mathcal{SN}_{\mathcal{SR}}$.

Les cas 11.4.31.(i) et 11.4.31.(ii) étant similaires, on ne traite que le second.

Cas 11.4.31.(ii). Supposons qu'il existe $k < n$ tel que

$$\tau_{\mathcal{S}}(c) \quad = \quad D_1 \Rightarrow \cdots \Rightarrow D_k \Rightarrow \forall \alpha_{k+1} \dots \alpha_n.\ C_{k+1}^{\alpha_{k+1}} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow C^{1+\max(\vec{\alpha})}$$

et
— $t_i \in [\![D_i]\!]$ pour tout $i \in \{1, \dots, k\}$,
— il existe $\vec{a}$ tels que $b = \max(\vec{a})$ et $t_j \in [\![C_j]\!](a_j)$ pour tout $j \in \{k+1, \dots, n\}$.
Alors comme c est un constructeur, si $ct_1 \dots t_n \to^*_{\mathcal{SR}} c't'_1 \dots t'_{n'}$ on a $c' = c$, $n' = n$ et $(t_1, \dots, t_n) \to_{\mathcal{SR}} (t'_1, \dots, t'_n)$. Le lemme 11.4.30 implique que $t'_i \in [\![D_i]\!]$ pour tout $i \in \{1, \dots, k\}$ et que $t'_j \in [\![C_j]\!][a_j]$ pour tout $j \in \{k+1, \dots, n\}$. On en déduit que $ct_1 \dots t_n \in [\![C]\!](b+1)$. □

11.4.4 Critère de terminaison

On présente maintenant notre critère de terminaison basé sur les types contraints.

Les deux points importants de ce critère sont que l'on peut formuler la fermeture calculable directement dans le système de types contraints, sans passer par un système de types annexe comme dans [BJO02, Bla05b], et que l'on obtient la normalisation forte non seulement des termes typés sur $\boldsymbol{\lambda}_{\mathcal{SR}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s), \Sigma_0, \tau_{\mathcal{S}})$ mais aussi des termes typés dans le système *non contraint* $\boldsymbol{\lambda}_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$.

Dans toute cette section on suppose donnés un ensemble de types de base $\mathcal{B}_0$, une signature Σ_0, une fonction $\tau_{\mathcal{S}} : \Sigma_0 \to \mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}s))$ et un ensemble $\mathcal{C}$ de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ tels que tout $C \in \mathcal{B}_0$ est un type inductif du premier ordre basé sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$. Rappelons que $\mathcal{F} = \Sigma_0 \setminus \mathcal{C}$.

On suppose de plus que $\mathcal{R}$ est un système de réécriture conditionnelle avec constructeurs inductifs, au sens de la définition 11.2.9, qui vérifie les conditions de la définition 11.1.1, et tel que la relation $\to_{\mathcal{SR}}$ soit confluente sur $\mathcal{SN}_{\mathcal{SR}}$. En particulier, les règles de $\mathcal{R}$ sont de la forme

$$d_1 = c_1 \ \wedge \ \dots \ \wedge \ d_m = c_m \ \supset \ f\ l_1 \dots l_n \ \mapsto_{\mathcal{R}} \ r$$

où $f \in \mathcal{F}$ et $c_j \in \{\mathsf{true}, \mathsf{false}\}$ pour tout $j \in \{1, \ldots, m\}$, et où pour tout $i \in \{1, \ldots, n\}$, le terme l_i est généré par la grammaire :

$$p \in \mathcal{PC} \quad ::= \quad x \quad | \quad c\, p_1 \ldots p_m$$

avec $x \in \mathcal{X}$, $c \in \mathcal{C}$ et $m = a_c$.

D'après le théorème 11.4.24, la normalisation forte dépend du fait que les symboles $f \in \Sigma_0$ appartiennent à l'interprétation de leur type. Le cas des constructeurs a été traité au lemme 11.4.31.

En ce qui concerne les symboles $f \in \mathcal{F}$, on a vu à la section 11.2 que pour la réécriture non-conditionnelle, il faut, pour toute règle $f\vec{l} \mapsto_{\mathcal{R}} r$, que toutes les occurrences de symboles $g \in \mathcal{F}$ dans r appartiennent à l'interprétation de leur type. Ainsi, les règles de réécriture induisent une relation de dépendance entre les symboles de $\mathcal{F}$, qui est appelée *précédence*. Rappelons qu'un préordre est une relation réflexive et transitive.

Définition 11.4.32 Une précédence *est un préordre dont la partie stricte est une relation bien fondée.*

Si $\leq_{\mathcal{F}}$ est une précédence, nous dénotons par $<_{\mathcal{F}}$ sa partie stricte et par $\simeq_{\mathcal{F}}$ la plus petite relation d'équivalence telle que $f \simeq_{\mathcal{F}} g$ si $f \leq_{\mathcal{F}} g \leq_{\mathcal{F}} f$.

Une précédence n'est utile que si elle respecte le système de réécriture $\mathcal{R}$. Dans le cas non conditionnel, cela veut dire que pour toute règle $f\vec{l} \mapsto_{\mathcal{R}} r$, on a $g \leq_{\mathcal{F}} f$ pour tout $g \in \mathcal{F}$ apparaissant dans r. Dans le cas conditionnel, on suppose que ceci est aussi vérifié pour les conditions des règles.

Définition 11.4.33 *On dit qu'une précédence $\leq_{\mathcal{F}}$ sur $\mathcal{F}$* respecte un système conditionnel $\mathcal{R}$ *si pour tout $f \in \mathcal{F}$ et toute règle $\vec{d} = \vec{c} \supset f\vec{l} \mapsto_{\mathcal{R}} r$ on a $g \leq_{\mathcal{F}} f$ pour tout $g \in \mathcal{F}$ apparaissant dans $\vec{d}$, $\vec{c}$ ou r.*

On suppose donnée une précédence $\leq_{\mathcal{F}}$ sur $\mathcal{F}$ qui respecte $\mathcal{R}$. De plus, on fait l'hypothèse que pour tout $c \in \mathcal{C}$, $\tau_{\mathcal{S}}(c)$ est le type défini en 11.4.27. D'autre part, on pose

$$\mathsf{true}^* =_{\text{def}} t \qquad t^* =_{\text{def}} t \qquad \text{et} \qquad \mathsf{false}^* =_{\text{def}} f \qquad f^* =_{\text{def}} f\,.$$

Remarque 11.4.34 *Notons que pour tout $b \in \{t, f\}$, si $t \in [\![\mathsf{Bool}]\!](b)$ et $t \to^*_{\mathcal{SR}} c$ avec $c \in \{\mathsf{true}, \mathsf{false}\}$, alors $b = c^{**}$.*

Le dernier outil technique à introduire avant de présenter le critère de terminaison est notre relation d'accessibilité contrainte. Comme nous allons le voir au lemme 11.4.37, elle doit être lue de la manière suivante : si $\alpha = a\ ;\ \Gamma \rightsquigarrow t : C^{\alpha}$ et $t\sigma \in [\![C]\!](a)$, alors il existe un assignement μ tel que $(\mu[a/\alpha], \sigma) \models \alpha = a\ ;\ \Gamma$.

Rappelons que deux contextes Γ_1 et Γ_2 sont compatibles si $\Gamma_1(x) = \Gamma_2(x)$ pour tout $x \in \mathcal{D}om(\Gamma_1) \cap \mathcal{D}om(\Gamma_2)$.

Définition 11.4.35 *Soit une injection $\epsilon : \mathcal{X} \to \mathcal{V}s$. On dit que deux contextes contraints $C_1\ ;\ \Gamma_1$ et $C_2\ ;\ \Gamma_2$ sont* strictement compatibles pour ϵ, *notation*

$$C_1\ ;\ \Gamma_1 \quad \uparrow_{\epsilon} \quad C_2\ ;\ \Gamma_2\ ,$$

si Γ_1 et Γ_2 sont compatibles et si pour tout $\beta \in \mathsf{FV}_{\mathcal{S}}(C_1, \Gamma_1) \cap \mathsf{FV}_{\mathcal{S}}(C_2, \Gamma_2)$, il existe $x \in \mathcal{X}$ tel que $\beta = \epsilon(x)$ avec $x \in \mathcal{D}om(\Gamma_1) \cap \mathcal{D}om(\Gamma_2)$ et $\Gamma_1(x) = \Gamma_2(x) = C^{\beta}$ où $C \in \mathcal{B}_{0\mathsf{Bool}}$.

Définition 11.4.36 (Accessiblité contrainte) *La relation* d'accessibilité contrainte $\alpha = a\ ;\ \Gamma \rightsquigarrow t : C^{\alpha}$ *est la plus petite relation satisfaisant aux règles de la figure 11.3, où* ϵ *est un injection de* $\mathcal{X}$ *dans* $\mathcal{V}s$.

$$(\textsc{Var})\ \frac{}{\alpha = \epsilon(x)\ ;\ x : C^{\epsilon(x)} \ \rightsquigarrow\ x : C^{\alpha}}$$

$$(\textsc{Cons}_0)\ \frac{\tau_{\mathcal{S}}(c) = \vec{D} \Rightarrow C^0 \qquad C \neq \mathsf{Bool}}{\alpha = 0\ ;\ \vec{x} : \vec{D} \ \rightsquigarrow\ c\vec{x} : C^{\alpha}} \qquad (\textsc{Cons}_{\mathsf{Bool}})\ \frac{\tau_{\mathcal{S}}(c) = \mathsf{Bool}^{c^*}}{\alpha = c^*\ ;\ \emptyset \ \rightsquigarrow\ c : \mathsf{Bool}^{\alpha}}$$

$$(\textsc{Cons})\ \frac{\begin{array}{c}\tau_{\mathcal{S}}(c) \ = \ \vec{D} \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow C^{1+\max(\vec{\alpha})} \\ \alpha_1 = a_1\ ;\ \Gamma_1 \ \rightsquigarrow\ u_1 : C_1^{\alpha_1} \quad \ldots \quad \alpha_n = a_n\ ;\ \Gamma_n \ \rightsquigarrow\ u_n : C_n^{\alpha_n}\end{array}}{\beta = 1 + \max(\vec{a})\ ;\ \vec{x} : \vec{D}, \vec{\Gamma} \ \rightsquigarrow\ c\vec{x}\vec{u} : C^{\beta}}$$

si $\quad \forall i \neq j.\quad \alpha_i = a_i\ ;\ \Gamma_i \ \uparrow_{\epsilon}\ \alpha_j = a_j\ ;\ \Gamma_j$,
$\beta \notin \mathsf{FV}(\vec{\alpha}, \vec{a}, \vec{\Gamma})$ et pour tout i, Γ_i est compatible avec $\vec{x} : \vec{D}$.

FIG. 11.3: Accessibilité contrainte

La correction de l'accessibilité contrainte, que l'on montre au lemme 11.4.37 suivant, repose sur le lemme 11.4.31.

Lemme 11.4.37 (Correction de l'accessibilité contrainte) *Si* $\alpha = a\ ;\ \Gamma \rightsquigarrow t : C^{\alpha}$ *et* $t\sigma \in [\![C]\!](a)$ *alors il existe* μ *tel que* $(\mu[a/\alpha], \sigma) \models \alpha = a\ ;\ \Gamma$.

PREUVE. Par induction sur $\alpha = a\ ;\ \Gamma \rightsquigarrow t : C^{\alpha}$.

(VAR) On prend μ tel que $\mu(\epsilon(x)) = a$.

($\textsc{Cons}_0$) et ($\textsc{Cons}_{\mathsf{Bool}}$). L'assignement $\mu = \emptyset$ convient.

(CONS) On a $(c\vec{x}\vec{u})\sigma \in [\![C]\!](a)$ avec $\alpha_i = a_i\ ;\ \Gamma_i \rightsquigarrow u_i : C_i^{\alpha_i}$ pour tout $i \in \{1, \ldots, n\}$. Par le lemme 11.4.31 il existe $\vec{a}$ tels que $a = 1 + \max(\vec{a})$ et $u_i\sigma \in [\![C_i]\!](a_i)$ pour tout $i \in \{1, \ldots, n\}$. Par hypothèse d'induction, pour tout $i \in \{1, \ldots, n\}$ il existe μ_i tel que $(\mu_i[a_i/\alpha_i], \sigma) \models \alpha_i = a_i\ ;\ \Gamma_i$.

D'autre part, par hypothèse pour tout $i \neq j$, les contextes contraints $\alpha_i = a_i\ ;\ \Gamma_i$ et $\alpha_j = a_j\ ;\ \Gamma_j$ sont strictement compatibles. Il s'en suit que pour tout $i \neq j$ et tout $\alpha \in \mathcal{V}s$, si $\gamma \in \mathsf{FV}(\alpha_i, a_i, \Gamma_i) \cap \mathsf{FV}(\alpha_j, a_j, \Gamma_j)$, alors $\mu_i[a_i/\alpha_i](\gamma) = \mu_j[a_j/\alpha_j](\gamma)$.

On peut donc définir

$$\mu \quad =_{\text{def}} \quad [\vec{a}/\vec{\alpha}] \ \uplus\ [\mu_i(\gamma)/\gamma \mid \gamma \in \mathsf{FV}(\alpha_i, a_i, \Gamma_i) \ \wedge\ i \in \{1, \ldots, n\}]$$

et on a $(\mu, \sigma) \models \vec{\alpha} = \vec{a}\ ;\ \vec{\Gamma}$. Il s'en suit que $(\mu[a/\alpha], \sigma) \models \alpha = 1 + \max(\vec{a})\ ;\ \vec{x} : \vec{D}, \vec{\Gamma}$ car $\alpha \notin \mathsf{FV}(\vec{\alpha}, \vec{a}, \vec{\Gamma})$. □

On aborde maintenant le critère de terminaison à proprement parler. Avant de le présenter formellement, essayons de voir quels en sont les éléments principaux.

Tout d'abord, nous faisons l'hypothèse que le type des symboles $f \in \mathcal{F}$ est de la forme

$$\tau_{\mathcal{S}}(f) = T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow T\ ,$$

où T est un type contraint basé sur les produits avec $\mathsf{FV}_{\mathcal{S}}(T) \subseteq \vec{\alpha}$, et où $\mathsf{FV}_{\mathcal{S}}(\vec{T}) = \emptyset$.

— Comme T est un type contraint basé sur les produits, d'après le lemme 11.4.18 il est interprété par un candidat de réductibilité. Ainsi, pour tous termes $t_1, \ldots, t_k$, $u_1, \ldots, u_n$ et tous $a_1, \ldots, a_n \in \mathfrak{D}$ tels que $t_i \in [\![T_i]\!]$ pour tout $i \in \{1, \ldots, k\}$ et $u_j \in [\![C_j]\!](a_j)$ pour tout $j \in \{1, \ldots, n\}$, on a $f\vec{t}\vec{u} \in [\![T]\!][\vec{a}/\vec{\alpha}]$ si et seulement si $(f\vec{t}\vec{u})_{\mathcal{SR}} \subseteq [\![T]\!][\vec{a}/\vec{\alpha}]$.

— La forme de $\tau_{\mathcal{S}}(f)$ permet de séparer ses *arguments récursifs*, qui ont des types inductifs et sur lesquels on peut faire du filtrage constructeur, de ses autres arguments qui sont nécessairement des variables. Nous supposons que les arguments sont donnés dans cet ordre là pour faciliter la présentation.

— Les conditions $\mathsf{FV}_{\mathcal{S}}(T) = \vec{\alpha}$ et $\mathsf{FV}_{\mathcal{S}}(\vec{T}) = \emptyset$ impliquent que le type de f est clos et que son type de sortie dépend uniquement de la taille de ses arguments récursifs.

Notons que ces conditions sont respectées par le type contraint du symbole pivot donné à l'exemple 11.3.8 :

$$\tau_{\mathcal{S}}(\mathsf{pivot}) \quad = \quad \mathsf{Nat} \Rightarrow \forall \alpha.\ \mathsf{List}^{\alpha} \Rightarrow \exists \beta_1 \beta_2\ (\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ .$$

Ensuite, nous supposons que f est muni d'une *contrainte de terminaison* $<_f$ telle que la relation $\prec_f$ définie par

$$\vec{b} \prec_f \vec{a} \qquad \text{si et seulement si} \qquad [\vec{b}/\vec{\beta}, \vec{a}/\vec{\alpha}] \models \vec{\beta} <_f \vec{\alpha}$$

est bien fondée. C'est cette relation qui sert d'argument de terminaison. Par exemple, pour $<_{\mathsf{pivot}}$, on prend l'ordre strict standard sur $\mathbb{N}$.

D'autre part, si $g \simeq_{\mathcal{F}} f$ alors f et g peuvent définir des fonctions mutuellement récursives : g peut apparaître dans le membre droit ou les conditions d'une règle définissant f et inversement, f peut apparaître dans le membre droit ou les conditions d'une règle définissant g. Les arguments de terminaison utilisés pour f et pour g doivent donc être identiques : f et g ont le même nombre d'arguments récursifs, de mêmes types, et ont même contrainte de terminaison.

Considérons maintenant une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ définissant f. Rappelons que l est un terme sur la grammaire

$$\begin{array}{rlll} & l & ::= & f\, p_1 \ldots p_{k+n} \\ \text{avec} & p \in \mathcal{PC} & ::= & x \ \mid\ c\, p_1 \ldots p_m \end{array}$$

où $x \in \mathcal{X}$, $c \in \mathcal{C}$ et $m = a_c$. Comme on sépare les arguments récursifs de f de ses autres arguments, on suppose de plus que l est de la forme $f x_1 \ldots x_k l_1 \ldots l_n$ où $l_j \in \mathcal{PC}$ pour

tout $j \in \{1, \dots, n\}$. Notons que le membre gauche de la règle (11.2) définissant pivot satisfait cette condition :

$$\begin{array}{lll} \mathsf{pivot}\ x\ (\mathsf{cons}\ y\ l) & \mapsto_{\mathsf{pivot}} & \mathsf{let}\ z = (\mathsf{pivot}\ x\ l)\ \mathsf{in} \\ & & \mathsf{ite}(>\ y\ x,\ (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ & & (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \end{array}$$

On impose que $l_j \in \mathcal{PC}$ pour tout $j \in \{1, \dots, n\}$ car on sait faire du filtrage constructeur pour les arguments récursifs, c'est-à-dire récupérer des informations de réductibilité sur les arguments des constructeurs.

Ceci est assuré par la relation d'accessibilité. Ainsi, on suppose qu'il existe des contextes contraints deux à deux strictement compatibles $(\alpha_i = a_i\ ;\ \Gamma_i)_{i\in\{1,\dots,n\}}$ tels que pour tout $i \in \{1, \dots, n\}$,

$$\alpha_i = a_i\ ;\ \Gamma_i \quad \rightsquigarrow \quad l_i : C_i^{\alpha_i}\ .$$

Par exemple, pour la règle de réduction de pivot on a

$$(\textsc{Cons})\ \frac{(\textsc{Var})\ \dfrac{}{l : \mathsf{List}^{\epsilon(l)} \ \rightsquigarrow\ l : \mathsf{List}^{\epsilon(l)}}}{\alpha = 1 + \epsilon(l)\ ;\ y : \mathsf{Nat}, l : \mathsf{List}^{\epsilon(l)} \ \rightsquigarrow\ \mathsf{cons}\ y\ l : \mathsf{List}^{\alpha}}$$

Enfin, on essaye de typer les conditions et le membre droits des règles en utilisant le contexte contraint généré par l'accessibilité. Considérons le cas d'une règle non conditionnelle $f\,\vec{x}\vec{l} \mapsto_{\mathcal{R}} r$ (un cas de réécriture conditionnelle est développé à l'exemple 11.4.42).

Comme nous l'avons déjà vu, l'important est ici d'assurer, pour toute substitution σ, une décroissance entre les arguments récursifs de f dans le membre gauche instantié et les arguments récursifs des symboles $g \simeq_{\mathcal{F}} f$ dans le membre droit instantié.

Pour ce faire, on type le membre droit r en utilisant certaines hypothèses sur le types des symboles. Rappelons que comme $\leq_{\mathcal{F}}$ est compatible avec $\mathcal{R}$, on a $g \leq_{\mathcal{F}} f$ pour tout $g \in \mathcal{F}$ apparaissant dans r. Le type $\tau^{<}_{\mathcal{S}}(g)$ des symboles $g \in \Sigma_0$ pouvant apparaître dans le membre droit est défini comme suit :

— $\tau^{<}_{\mathcal{S}}(c) =_{\text{def}} \tau_{\mathcal{S}}(c)$ pour tout $c \in \mathcal{C}$,
— $\tau^{<}_{\mathcal{S}}(g) =_{\text{def}} \tau_{\mathcal{S}}(g)$ pour tout $g <_{\mathcal{F}} f$,
— pour tout $g \simeq_{\mathcal{F}} f$ tel que

$$\tau_{\mathcal{S}}(g) = U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}'.\ C_1^{\alpha'_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha'_n} \Rightarrow U\ ,$$

on pose

$$\tau^{<}_{\mathcal{S}}(g) \quad =_{\text{def}} \quad U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}'(\vec{\alpha}' <_f \vec{\alpha}).\ C_1^{\alpha'_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha'_n} \Rightarrow U[\vec{\alpha}'/\vec{\alpha}]\ ,$$

où $\vec{\alpha} \cap \vec{\alpha}' = \emptyset$.

Ainsi, si $g \simeq_{\mathcal{F}} f$, le type $\tau^{<}_{\mathcal{S}}(g)$ oblige à utiliser g dans le membre droit avec des arguments récursifs strictement plus petit que ceux de f dans le membre gauche. En utilisant ce typage des symboles, on suppose que r est typable de la manière suivante :

$$\vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma} \vdash_{\tau^{<}_{\mathcal{S}}} r\ :\ T\ .$$

Nous allons voir à l'exemple 11.4.41 comment obtenir ceci pour la règle (11.2) de pivot.

Passons maintenant à la formulation formelle de notre critère de terminaison, et à la preuve de sa correction.

Théorème 11.4.38 (Normalisation forte dans $\lambda_{\mathcal{SR}}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}), \Sigma_0, \tau_{\mathcal{S}})$) *Soit un ensemble de types de base $\mathcal{B}_0$, une signature Σ_0, une fonction de typage $\tau_{\mathcal{S}}$ de Σ_0 dans $\mathcal{T}_{\Rightarrow,\times}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\mathfrak{s}))$ et un ensemble $\mathcal{C}$ de types inductifs basés sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \tau)$ tels que tout $C \in \mathcal{B}_0$ est un type inductif du premier ordre basé sur les booléens sur $(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$. Rappelons que $\mathcal{F} = \Sigma_0 \setminus \mathcal{C}$.*

Soit de plus $\mathcal{R}$ un système de réécriture conditionnelle avec constructeurs inductifs, au sens de la définition 11.2.9, qui vérifie les conditions de la définition 11.1.1, et tel que la relation $\rightarrow_{\mathcal{SR}}$ soit confluente sur $\mathcal{SN}_{\mathcal{SR}}$.

Soit enfin un précédence $\leq_{\mathcal{F}}$ sur $\mathcal{F}$ qui respecte $\mathcal{R}$ au sens de la définition 11.4.33.

Supposons que pour tout $f \in \mathcal{F}$,

— *$\tau_{\mathcal{S}}(f) = T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow T$ où T est un type contraint basé sur les produits avec $\mathsf{FV}_{\mathcal{S}}(T) = \vec{\alpha}$ et où $\mathsf{FV}_{\mathcal{S}}(\vec{T}) = \emptyset$.*

— *il existe une contrainte $\vec{\beta} <_f \vec{\alpha}$ telle que la relation $\prec_f$ définie par*

$$\vec{b} \prec_f \vec{a} \qquad \textit{si et seulement si} \qquad [\vec{b}/\vec{\beta}, \vec{a}/\vec{\alpha}] \models \vec{\beta} <_f \vec{\alpha}$$

est bien fondée ;

— *pour tout $g \simeq_{\mathcal{F}} f$, $\tau_{\mathcal{S}}(g)$ est de la forme*

$$U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow U$$

avec $<_g = <_f$.

De plus, supposons que pour toute règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ définissant f, le membre gauche l est de la forme $f x_1 \dots x_k l_1 \dots l_n$, et qu'il existe des contextes contraints deux à deux strictement compatibles $(\alpha_i = a_i\ ;\ \Gamma_i)_{i \in \{1,\dots,n\}}$ tels que pour tout $i \in \{1, \dots, n\}$,

$$\alpha_i = a_i\ ;\ \Gamma_i \quad \rightsquigarrow \quad l_i : C_i^{\alpha_i}\ .$$

Enfin, on suppose que les conditions et le membre droit de la règle peuvent être typés de la manière suivante :

$$\begin{aligned} \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma} \ &\vdash_{\tau_{\mathcal{S}}^{<}} \ \vec{d} : \mathsf{Bool}^{\vec{b}} \\ \vec{b} = \vec{c}^*,\ \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma} \ &\vdash_{\tau_{\mathcal{S}}^{<}} \ r : T\ , \end{aligned}$$

où

— *$\tau_{\mathcal{S}}^{<}(c) =_{def} \tau_{\mathcal{S}}(c)$ pour tout $c \in \mathcal{C}$,*

— *$\tau_{\mathcal{S}}^{<}(g) =_{def} \tau_{\mathcal{S}}(g)$ pour tout $g <_{\mathcal{F}} f$,*

— *pour tout $g \simeq_{\mathcal{F}} f$ tel que*

$$\tau_{\mathcal{S}}(g) = U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}'.\ C_1^{\alpha'_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha'_n} \Rightarrow U\ ,$$

on pose

$$\tau_{\mathcal{S}}^{<}(g) \quad =_{def} \quad U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}'(\vec{\alpha}' <_f \vec{\alpha}).\ C_1^{\alpha'_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha'_n} \Rightarrow U[\vec{\alpha}'/\vec{\alpha}]\ ,$$

où $\vec{\alpha} \cap \vec{\alpha}' = \emptyset$.

Si $C;\Gamma \vdash t : T$ *et si* C *est satisfiable, alors* t *est fortement* $\mathcal{SR}$*-normalisant.*

PREUVE. D'après le corollaire 11.4.25, il suffit de montrer que $f \in [\![\tau_{\mathcal{S}}(f)]\!]$ pour tout $f \in \Sigma_0$. Le cas des constructeurs est traité au lemme 11.4.31. Il reste donc le cas des symboles $f \in \mathcal{F}$.

On montre que pour tout $f \in \mathcal{F}$ avec

$$\tau_{\mathcal{S}}(f) \quad = \quad T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow T\ ,$$

pour tout $t_1, \ldots, t_k, u_1, \ldots, u_n$ et tout μ tels que $t_i \in [\![T_i]\!]\mu$ pour tout $i \in \{1, \ldots, k\}$ et $u_j \in [\![C_j]\!](\mu(\alpha_j))$ pour tout $j \in \{1, \ldots, n\}$, on a $f\vec{t}\vec{u} \in [\![T]\!]\mu$. On raisonne par induction sur les tuples $(f, \mu(\vec{\alpha}), (t_1, \ldots, t_k, u_1, \ldots, u_n))$ ordonnés par $(>_{\mathcal{F}}, \succ_f, \to_{\mathcal{SR}})_{\text{lex}}$.

Soit un symbole f, des termes $t_1, \ldots, t_k$, $u_1, \ldots, u_n$ et un assignement μ vérifiant ces conditions. On doit montrer que $f\vec{t}\vec{u} \in [\![T]\!]\mu$. Comme c'est un terme neutre et comme T est un type contraint basé sur les produits, par le lemme 11.4.18 il suffit de montrer que $(f\vec{t}\vec{u})_{\mathcal{SR}} \subseteq [\![T]\!]\mu$.

Soit donc ν tel que $f\vec{t}\vec{u} \to_{\mathcal{SR}} \nu$. Si $\nu = f t'_1 \ldots t'_k u'_1 \ldots u'_n$ avec

$$(t_1, \ldots, t_k, u_1, \ldots, u_n) \ \to_{\mathcal{SR}}\ (t'_1, \ldots, t'_k, u'_1, \ldots, u'_n)\ ,$$

alors par ($\mathbb{SAT}0$) on a $t'_i \in [\![T_i]\!]\mu$ pour tout $i \in \{1, \ldots, k\}$ et $u'_j \in [\![C_j]\!](\mu(\alpha_j))$ pour tout $j \in \{1, \ldots, n\}$. Il s'en suit que $\nu \in [\![T]\!]\mu$ par hypothèse d'induction.

Sinon il existe une règle $\vec{d} = \vec{c} \supset l \mapsto_{\mathcal{R}} r$ et une substitution σ telles que $l\sigma = f\vec{t}\vec{u}$ et $\nu = r\sigma$.

Pour tout $i \in \{1, \ldots, n\}$, posons $a_i =_{\text{def}} \mu(\alpha_i)$. Comme $\alpha_i = a_i\ ;\ \Gamma_i \rightsquigarrow l_i : C_i^{\alpha_i}$, par le lemme 11.4.37 il existe ξ_i tel que $(\xi_i[a_i/\alpha_i], \sigma) \models \alpha_i = a_i\ ;\ \Gamma_i$. De plus, comme les contextes $(\alpha_i = a_i\ ;\ \Gamma_i)_{i \in \{1,\ldots,n\}}$ sont deux à deux compatibles, en posant

$$\xi \quad =_{\text{def}} \quad [\xi_i(\gamma)/\gamma \mid \gamma \in \mathsf{FV}(\alpha_i, a_i, \Gamma_i) \ \wedge\ i \in \{1, \ldots, n\}]$$

on a $(\xi[\vec{a}/\vec{\alpha}], \sigma) \models \vec{\alpha} = \vec{a}\ ;\ \vec{\Gamma}$. De plus on a $(\xi[\vec{a}/\vec{\alpha}], \sigma) \models \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}$ car les types $\vec{T}$ sont clos.

Soit maintenant $g \leq_{\mathcal{F}} f$ avec

$$\tau_{\mathcal{S}}(g) \quad = \quad \tau_{\mathcal{S}}(g) = U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}'.\ D_1^{\alpha'_1} \Rightarrow \cdots \Rightarrow D_m^{\alpha'_m} \Rightarrow U$$

— Si $g <_{\mathcal{F}} f$ alors on a $\tau_{\mathcal{S}}(g) = \tau_{\mathcal{S}}^{<}(g)$.
De plus, pour tout $\xi' : \mathcal{V}s \to \mathfrak{D}$ et tout $\nu_1, \ldots, \nu_p, w_1, \ldots, w_m$ tels que $\nu_i \in [\![U_i]\!]\xi'$ pour tout $i \in \{1, \ldots, p\}$ et $w_j \in [\![D_j]\!](\xi'(\alpha'_j))$ pour tout $j \in \{1, \ldots, m\}$, on a

$$(f, \vec{a}, (\vec{t}, \vec{u}))\ \ (>_{\mathcal{F}}, \succ_f, \to_{\mathcal{SR}})_{\text{lex}}\ \ (g, \xi'(\vec{\alpha}'), (\vec{v}, \vec{w}))$$

donc $g\vec{v}\vec{w} \in [\![\tau_{\mathcal{S}}(g)]\!]\xi'$ par hypothèse d'induction.
Il s'en suit que $g \in [\![\tau_{\mathcal{S}}^{<}(g)]\!]$.

— Sinon, on a

$$\tau^{<}_{\mathcal{S}}(g) \quad = \quad U_1 \Rightarrow \cdots \Rightarrow U_p \Rightarrow \forall \vec{\alpha}'(\vec{\alpha}' <_f \vec{\alpha}).\ C_1^{\alpha'_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha'_n} \Rightarrow U\ ,$$

avec $\vec{\alpha} \cap \vec{\alpha}' = \emptyset$. Notons que l'on peut supposer que $\vec{\alpha}, \vec{\alpha}' \notin FV_{\mathcal{S}}(\vec{U})$.
Ainsi, pour tout $\vec{a}' \prec_f \vec{a}$, tout $\xi' : \mathcal{V}s \to \mathfrak{D}$ et tout $v_1, \ldots, v_p$, $w_1, \ldots, w_n$ tels que $v_i \in [\![U_i]\!]\xi'[\vec{a}'/\vec{\alpha}']$ pour tout $i \in \{1, \ldots, p\}$ et $w_j \in [\![C_j]\!](\vec{a}')$ pour tout $j \in \{1, \ldots, n\}$, on a

$$(f, \vec{a}, (\vec{t}, \vec{u}))\ \ (>_{\mathcal{F}}, \succ_f, \to_{\mathcal{SR}})_{lex}\ \ (g, \vec{a}', (\vec{v}, \vec{w}))$$

donc $g\vec{v}\vec{w} \in [\![\tau_{\mathcal{S}}(g)]\!]\xi'[\vec{a}'/\vec{\alpha}']$ par hypothèse d'induction.
Il s'en suit que $g \in [\![\tau^{<}_{\mathcal{S}}(g)]\!]$.

Du fait que

$$\vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}\ \ \vdash_{\tau^{<}_{\mathcal{S}}}\ \ \vec{d}\ :\ Bool^{\vec{b}}\ ,$$

et que $(\xi[\vec{a}/\vec{\alpha}], \sigma) \models \vec{\alpha} = \vec{a}; \vec{x} : \vec{T}, \vec{\Gamma}$, en appliquant le théorème 11.4.24 aux symboles $\mathcal{C} \cup \{g \mid g \leq_{\mathcal{F}} f\}$ typés par $\tau^{<}_{\mathcal{S}}$, on obtient $\vec{d}\sigma \in [\![Bool]\!](\xi[\vec{a}/\vec{\alpha}](\vec{b}))$. Par la remarque 11.4.34 on a alors $\xi[\vec{a}/\vec{\alpha}](\vec{b}) = \vec{c}^{**}$ et on en déduit que $(\xi[\vec{a}/\vec{\alpha}], \sigma) \models \vec{b} = \vec{c}^{*}, \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}$.

De plus, comme on a

$$\vec{b} = \vec{c}^{*}, \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}\ \ \vdash_{\tau^{<}_{\mathcal{S}}}\ \ r\ :\ T\ ,$$

en appliquant le théorème 11.4.24 aux symboles $\mathcal{C} \cup \{g \mid g \leq_{\mathcal{F}} f\}$ typés par $\tau^{<}_{\mathcal{S}}$, il vient $r\sigma \in [\![T]\!]\xi[\vec{a}/\vec{\alpha}]$.

D'autre part, on a $[\![T]\!]\xi[\vec{a}/\vec{\alpha}] = [\![T]\!][\vec{a}/\vec{\alpha}] = [\![T]\!]\mu$ car $FV_{\mathcal{S}}(T) = \vec{\alpha}$. Il s'en suit que $r\sigma \in [\![T]\!]\mu$. □

Remarque 11.4.39 *D'après la remarque 11.3.22, pour vérifier que*

$$\begin{array}{r} \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}\ \ \vdash_{\tau^{<}_{\mathcal{S}}}\ \ \vec{d}\ :\ Bool^{\vec{b}} \\ \vec{b} = \vec{c}^{*},\ \vec{\alpha} = \vec{a}\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}\ \ \vdash_{\tau^{<}_{\mathcal{S}}}\ \ r\ :\ T\ , \end{array}$$

on peut essayer de générer des contraintes C_1 et C_2 telles que

$$\begin{array}{l} C_1\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}\ \ \vdash_{\tau^{<}_{\mathcal{S}}}\ \ \vec{d}\ \downarrow\ Bool^{\vec{b}} \\ C_2\ ;\ \vec{x} : \vec{T}, \vec{\Gamma}\ \ \vdash_{\tau^{<}_{\mathcal{S}}}\ \ r\ \downarrow\ T\ , \end{array}$$

et tester si $\vdash (\vec{\alpha} = \vec{a} \supset C_1) \wedge ([\vec{\alpha} = \vec{a} \wedge \vec{b} = \vec{c}^{*}] \supset C_2)$.

Si les types des arguments non récursifs des symboles $f \in \mathcal{F}$ sont des types contraints simples, au sens de la définition 11.3.5.(ii), alors on obtient la normalisation forte des termes typés dans le système *non contraint* $\lambda_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$.

Théorème 11.4.40 (Normalisation forte dans $\lambda_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$) *Sous les hypothèses du théorème 11.4.38, si pour tout $f \in \mathcal{F}$, on a*

$$\tau_{\mathcal{S}}(f) \quad = \quad T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow T\ ,$$

où T_i est un type simple contraint pour tout $i \in \{1, \ldots, k\}$, alors les termes typables de $\lambda_{\underline{\mathcal{SR}}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$ sont fortement $\mathcal{SR}$-normalisants.

PREUVE. On raisonne par induction sur $\Gamma \vdash t : T$, comme au lemme 9.1.26. Les règles (Bool) et (let) sont traitées comme au théorème 11.4.24. Voyons enfin le cas de la règle (SYMB I). Soit $f \in \Sigma_0$.

— Supposons que $f = c \in \mathcal{C}$. On ne considère que le cas où

$$\tau_{\mathcal{S}}(c) = D_1 \Rightarrow \cdots \Rightarrow D_k \Rightarrow \forall \vec{\alpha}.\ C_{k+1}^{\alpha_{k+1}} \Rightarrow \dots C_n^{\alpha_n} \Rightarrow C^{max(\vec{\alpha})+1}\ ,$$

les autres étant similaires et plus simples. D'après la remarque 11.4.13, pour tout μ, on a $[\![C^{max(\vec{\alpha})+1}]\!]\mu \subseteq [\![C]\!]$ et de plus $[\![C^{\alpha_i}]\!]\mu \subseteq [\![C]\!]$ pour tout $i \in \{k+1, \dots, n\}$. On en déduit que $[\![\tau_{\mathcal{S}}(c)]\!] \subseteq [\![\underline{\tau_{\mathcal{S}}(c)}]\!]$ par la proposition 10.2.4, d'où $c \in [\![\underline{\tau_{\mathcal{S}}(c)}]\!]$ d'après le lemme 11.4.31.

— Sinon, $f \in \Sigma$, et $\tau_{\mathcal{S}}(f) = T_1 \Rightarrow \cdots \Rightarrow T_k \Rightarrow \forall \vec{\alpha}.\ C_1^{\alpha_1} \Rightarrow \cdots \Rightarrow C_n^{\alpha_n} \Rightarrow T$, où T_i est un type simple contraint pour tout $i \in \{1, \dots, k\}$, et où T est un type contraint basé sur les produits.
Par la remarque 11.4.13, on a donc $[\![T_i]\!] = [\![\underline{T_i}]\!]$ pour tout $i \in \{1, \dots, k\}$ et $[\![C_j^{\alpha_j}]\!]\mu \subseteq [\![C]\!]$ pour tout $j \in \{1, \dots, n\}$ et tout μ.
Enfin, comme T est un type contraint basé sur les produits, $\underline{T}$ est un type basé sur les produits, et il suit de la remarque 11.4.13 et de la proposition 11.4.21 que l'on a $[\![T]\!]\mu \subseteq [\![\underline{T}]\!]$ pour tout μ.
On en déduit que $f \in [\![\underline{\tau_{\mathcal{S}}(f)}]\!]$ car $f \in [\![\tau_{\mathcal{S}}(f)]\!]$ par le théorème 11.4.38. □

Exemple 11.4.41 *On revient sur la règle (11.2) définissant le symbole* pivot. *Rappelons que cette règle est définie à l'exemple 11.1.7, et que l'on a commencé à l'étudier juste avant le théorème 11.4.38. D'autre part, la règle (11.2) utilise le symbole* $>$ *défini à l'exemple 11.1.4. On désigne par* pivot *le système de réécriture* $\mapsto_{\mathsf{pivot}} \cup \mapsto_{>}$. *Rappelons que pour les règles de* pivot, *on utilise* $>$ *avec le type* $\mathsf{Nat} \Rightarrow \mathsf{Nat} \Rightarrow \mathsf{Bool}$.

On montre maintenant que

$$\mathsf{pivot} \in [\![\mathsf{Nat} \Rightarrow \forall\alpha.\ \mathsf{List}^{\alpha} \Rightarrow \exists\beta_1\beta_2\ (\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}]\!]\ . \tag{11.12}$$

Notons que le type contraint de pivot *satisfait les hypothèse du théorème 11.4.40. Ainsi, si les règles définissant* $>$ *sont calculables, on obtient la* $\mathcal{S}$pivot*-normalisation pour les termes typables de* $\boldsymbol{\lambda}_{\underline{\mathcal{S}}\mathsf{pivot}}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$.

Pour obtenir (11.12), il reste à montrer qu'avec $\Gamma =_{def} x : \mathsf{Nat}, y : \mathsf{Nat}, l : \mathsf{List}^{\epsilon(l)}$, *on a*

$$\begin{array}{l} \alpha = 1 + \epsilon(l)\ ;\ \Gamma \vdash \mathsf{let}\ z = (\mathsf{pivot}\ x\ l)\ \mathsf{in} \\ \mathsf{ite}(>\ y\ x,\ (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \qquad\qquad : \ \exists\beta_1\beta_2\ (\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2} \end{array}$$

Pour ce faire on va générer une contrainte C *telle que* $\vdash (\alpha = \epsilon(l) + 1) \supset C$ *et*

$$\begin{array}{l} C\ ;\ \Gamma \vdash \mathsf{let}\ z = (\mathsf{pivot}\ x\ l)\ \mathsf{in} \\ \mathsf{ite}(>\ y\ x,\ (\pi_1\ z,\ \mathsf{cons}\ y\ (\pi_2\ z)), \\ (\mathsf{cons}\ y\ (\pi_1\ z),\ \pi_2\ z)) \qquad\qquad \downarrow\ \exists\beta_1\beta_2\ (\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2} \end{array} \tag{11.13}$$

D'après ce que l'on a vu à l'exemple 11.3.23, en utilisant le type

$$\tau_{\mathcal{S}}^{<} \quad = \quad \mathsf{Nat} \Rightarrow \forall\gamma(\gamma < \alpha).\ \mathsf{List}^{\alpha} \Rightarrow \exists\gamma_1\gamma_2\ (\gamma = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}\ ,$$

on a

$$\epsilon(\mathsf{l}) < \alpha \,;\, \Gamma \vdash \mathsf{pivot}\ \mathsf{x}\ \mathsf{l} \uparrow \exists\gamma_1\gamma_2\,(\epsilon(\gamma) = \gamma_1 + \gamma_2).\ \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}\ .$$

On peut appliquer la règle $(\downarrow \exists\,\mathrm{E})$ *si on arrive à générer une contrainte* D *telle que*

$$\begin{array}{l}\mathsf{D}\,;\,\Gamma, \mathsf{z} : \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2} \vdash \\ \mathsf{ite}(> \mathsf{y}\ \mathsf{x},\ (\pi_1\ \mathsf{z},\ \mathsf{cons}\ \mathsf{y}\ (\pi_2\ \mathsf{z})), \\ (\mathsf{cons}\ \mathsf{y}\ (\pi_1\ \mathsf{z}),\ \pi_2\ \mathsf{z})) \qquad\qquad \downarrow\ \ \exists\beta_1\beta_2\,(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ ,\end{array}$$

On va utiliser la règle $(\downarrow \mathsf{Bool})$. *Posons* $\Gamma_z =_{def} \Gamma, \mathsf{z} : \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}$. *Tout d'abord, il est clair que* $;\Gamma_z \vdash\ > \mathsf{y}\ \mathsf{x} \uparrow \mathsf{Bool}$. *D'autre part, comme*

$$;\Gamma_z \vdash \mathsf{y} \downarrow \mathsf{Nat} \qquad \textit{et} \qquad ;\Gamma_z \vdash \mathsf{cons} \uparrow \mathsf{Nat} \Rightarrow \forall\gamma_2.\ \mathsf{List}^{\gamma_2} \Rightarrow \mathsf{List}^{\gamma_2+1}\ ,$$

on a

$$;\Gamma_z \vdash \mathsf{cons}\ \mathsf{y} \uparrow \forall\gamma_2.\ \mathsf{List}^{\gamma_2} \Rightarrow \mathsf{List}^{\gamma_2+1}\ ,$$

soit

$$;\Gamma_z \vdash \mathsf{cons}\ \mathsf{y} \uparrow \mathsf{List}^{\gamma_2} \Rightarrow \mathsf{List}^{\gamma_2+1}\ .$$

Du fait que

$$(\uparrow \times \mathrm{E})\ \frac{;\Gamma_z \vdash \mathsf{z} \uparrow \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2}}{;\Gamma_z \vdash \pi_2\,\mathsf{z} \uparrow \mathsf{List}^{\gamma_2}}$$

on en déduit que

$$;\Gamma_z \vdash \mathsf{cons}\ \mathsf{y}\ (\pi_2\,\mathsf{z}) \uparrow \mathsf{List}^{\gamma_2+1}\ .$$

On a aussi $;\Gamma_z \vdash \pi_1\,\mathsf{z} \uparrow \mathsf{List}^{\gamma_1}$, *d'où*

$$;\Gamma_z \vdash (\pi_1\,\mathsf{z}, \mathsf{cons}\ \mathsf{y}\ (\pi_2\,\mathsf{z})) \uparrow \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2+1}\ .$$

Comme $\|\mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2+1} \sqsubseteq \exists\beta_1\beta_2(\alpha = \beta_1 + \beta_2).\mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\|$ *est la formule*

$$\Pi_1 \quad =_{def} \quad \exists\beta_1\beta_2.\ \alpha = \beta_1 + \beta_2 \ \wedge\ \gamma_1 = \beta_2 \ \wedge\ \gamma_2 + 1 = \beta_2\ ,$$

il s'en suit que

$$\Pi_1\,;\,\Gamma_z \vdash (\pi_1\,\mathsf{z}, \mathsf{cons}\ \mathsf{y}\ (\pi_2\,\mathsf{z})) \downarrow \exists\beta_1\beta_2(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ .$$

De manière similaire, avec

$$\Pi_2 \quad =_{def} \quad \exists\beta_1\beta_2.\ \alpha = \beta_1 + \beta_2 \ \wedge\ \gamma_1 + 1 = \beta_2 \ \wedge\ \gamma_2 = \beta_2\ ,$$

on a

$$\Pi_2\,;\,\Gamma_z \vdash (\mathsf{cons}\ \mathsf{y}\ (\pi_1\,\mathsf{z}), \pi_2\,\mathsf{z}) \downarrow \exists\beta_1\beta_2\,(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ .$$

La règle $(\downarrow \mathsf{Bool})$ *donne alors*

$$\begin{array}{l}\Pi_1 \wedge \Pi_2\,;\,\Gamma, \mathsf{z} : \mathsf{List}^{\gamma_1} \times \mathsf{List}^{\gamma_2} \vdash \\ \mathsf{ite}(> \mathsf{y}\ \mathsf{x},\ (\pi_1\ \mathsf{z},\ \mathsf{cons}\ \mathsf{y}\ (\pi_2\ \mathsf{z})), \\ (\mathsf{cons}\ \mathsf{y}\ (\pi_1\ \mathsf{z}),\ \pi_2\ \mathsf{z})) \qquad\qquad \downarrow\ \ \exists\beta_1\beta_2\,(\alpha = \beta_1 + \beta_2).\ \mathsf{List}^{\beta_1} \times \mathsf{List}^{\beta_2}\ .\end{array}$$

Enfin, en posant

$$\mathsf{C} =_{def} \epsilon(\mathsf{l}) < \alpha \ \wedge\ \exists\gamma_1\gamma_2(\epsilon(\mathsf{l}) = \gamma_1 + \gamma_2) \ \wedge\ \forall\gamma_1\gamma_2\,(\epsilon(\mathsf{l}) = \gamma_1 + \gamma_2 \supset (\Pi_1 \wedge \Pi_2))\ ,$$

on obtient (11.13). D'autre part, il est aisé de voir que $\vdash (\alpha = \epsilon(\mathsf{l}) + 1) \supset \mathsf{C}$.

Exemple 11.4.42 (McCarthy) *On s'intéresse maintenant à la fonction* 91 *de McCarthy, présentée à l'exemple 11.1.8. On considère un symbole* f *de type non contraint* $\mathsf{Nat} \Rightarrow \mathsf{Nat}$ *défini par les règles conditionnelles :*

$$\begin{array}{lllllll} > \ x\ 100 & = & \mathsf{true} & \supset & f\ x & \mapsto_{91} & \mathsf{minus}\ x\ 10\ , \\ > \ x\ 100 & = & \mathsf{false} & \supset & f\ x & \mapsto_{91} & f\ (f\ (\mathsf{plus}\ x\ 11))\ , \end{array}$$

où, pour tout $n \in \mathbb{N}$, *le terme* $\mathsf{S}^n 0$ *est représenté par* n. *Il est clair que l'on a* $; \vdash \mathsf{S}^n 0 : \mathsf{Nat}^n$ *Rappelons que* $f\ n \to^+_{91} \mathsf{minus}\ n\ 10$ *si* $n > 100$ *et* $f\ n \to^+_{91} 91$ *sinon.*

On suppose que $\mathfrak{S}$ *contient un symbole* $\_ > \_ : \mathsf{nat} \times \mathsf{nat} \to \mathsf{bool}$, *interprété dans* $\mathfrak{D}$ *par la fonction booléenne du même nom, ainsi que de la soustraction entière* $\_ - \_$ *de sorte* $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$.

Le système $\mapsto_{91}$ *utilise les symboles* $>$, plus *et* minus, *définis à l'exemple 11.1.4. On les type ici de la manière suivante :*

$$\begin{array}{rcl} \tau_{\mathcal{S}}(>) & =_{def} & \forall \alpha_1.\ \mathsf{Nat}^{\alpha_1} \Rightarrow \forall \alpha_2.\ \mathsf{Nat}^{\alpha_2} \Rightarrow \mathsf{Bool}^{\alpha_1 > \alpha_2}\ , \\ \tau_{\mathcal{S}}(\mathsf{plus}) & =_{def} & \forall \alpha_1.\ \mathsf{Nat}^{\alpha_1} \Rightarrow \forall \alpha_2.\ \mathsf{Nat}^{\alpha_2} \Rightarrow \mathsf{Nat}^{\alpha_1 + \alpha_2}\ , \\ \tau_{\mathcal{S}}(\mathsf{minus}) & =_{def} & \forall \alpha_1.\ \mathsf{Nat}^{\alpha_1} \Rightarrow \forall \alpha_2.\ \mathsf{Nat}^{\alpha_2} \Rightarrow \forall \gamma\ Q_{\mathsf{minus}}(\alpha_1, \alpha_2, \gamma).\ \mathsf{Nat}^{\gamma}\ , \end{array}$$

où

$$Q_{\mathsf{minus}}(\alpha_1, \alpha_2, \gamma) \quad =_{def} \quad ((\alpha_1 < \alpha_2) \supset \gamma = 0) \quad \wedge \quad ((\alpha_2 \leq \alpha_1) \supset \alpha_1 = \gamma + \alpha_2)\ .$$

Notons que ce type pour le symbole $>$ *diffère de celui utilisé aux exemples 11.3.23 et 11.4.41. On suppose que* $>$, plus *et* minus *appartiennent à l'interprétation de leurs types respectifs et on se concentre sur les règles définissant* f.

Le type contraint de f *est le suivant :*

$$\tau_{\mathcal{S}}(f) \quad =_{def} \quad \forall \alpha.\ \mathsf{Nat}^{\alpha} \Rightarrow \forall \gamma\ Q_f(\alpha, \gamma).\ \mathsf{Nat}^{\gamma}\ ,$$

où

$$Q_f(\alpha, \gamma) \quad =_{def} \quad (\alpha \leq 100 \ \supset \ \gamma = 91) \ \wedge \ (100 < \alpha \ \supset \ \alpha = \gamma + 10)\ .$$

De plus, on utilise la contrainte de terminaison suivante :

$$\alpha <_f \beta \quad =_{def} \quad \max(0, 101 - \alpha) < \max(0, 101 - \beta)\ ,$$

où $<$ *est l'ordre standard sur* $\mathbb{N}$.

Pour les deux règles, la relation d'accessibilité est

$$\alpha = \epsilon(x)\ ;\ x : \mathsf{Nat}^{\epsilon(x)} \ \rightsquigarrow \ x : \mathsf{Nat}^{\alpha}\ .$$

Il est aisé de dériver que

$$\alpha = \epsilon(x)\ ;\ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ > x\ 100 \ : \ \mathsf{Bool}^{\alpha > 100}\ ,$$

et il s'en suit que les contraintes générées par les conditions des deux règles de f *sont respectivement*

$$(\alpha > 100) = t \qquad et \qquad (\alpha > 100) = f\ .$$

Dans le cas de la première règle, on a

$$(\alpha > 100) = \mathsf{t} \ \wedge \ \alpha = \epsilon(x) \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \vdash \mathsf{minus}\ x\ 10 \ : \ \forall\gamma\, Q_{\mathsf{minus}}(\epsilon(x), 10, \gamma)\mathsf{Nat}^{\gamma} \ .$$

Or, comme

$$\vdash ((\alpha > 100) = \mathsf{t} \ \wedge \ \alpha = \epsilon(x)) \supset \|\forall\gamma\, Q_{\mathsf{minus}}(\epsilon(x), 10, \gamma).\ \mathsf{Nat}^{\gamma} \sqsubseteq \forall\delta\, Q_f(\alpha, \delta).\ \mathsf{Nat}^{\delta}\| \ ,$$

on en déduit que

$$(\alpha > 100) = \mathsf{t} \ \wedge \ \alpha = \epsilon(x) \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \vdash \mathsf{minus}\ x\ 10 \ : \ \forall\delta\, Q_f(\alpha, \delta).\ \mathsf{Nat}^{\delta} \ .$$

On doit typer la seconde règle en utilisant f *avec le type*

$$\tau^{<}_{\mathcal{S}}(f) \quad = \quad \forall\alpha'(\alpha' <_f \alpha).\ \mathsf{Nat}^{\alpha'} \Rightarrow \forall\delta\, Q_f(\alpha', \delta).\ \mathsf{Nat}^{\delta} \ .$$

En posant $D =_{def} (\alpha > 100) = f \ \wedge \ \alpha = \epsilon(x)$, *on a*

$$D \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ \mathsf{plus}\ x\ 11 \ : \ \mathsf{Nat}^{\epsilon(x)+11} \ .$$

D'autre part,

$$(\forall\,\mathrm{E})\ \frac{D \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f \ : \ \forall\alpha'(\alpha' <_f \alpha).\ \mathsf{Nat}^{\alpha'} \Rightarrow \forall\delta\, Q_f(\alpha', \delta).\ \mathsf{Nat}^{\delta}}{D \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f \ : \ \mathsf{Nat}^{\epsilon(x)+11} \Rightarrow \forall\delta\, Q_f(\epsilon(x) + 11, \delta).\ \mathsf{Nat}^{\delta}}$$

car

$$\vdash \ D \ \supset \ \mathrm{max}(0, 101 - (\epsilon(x) + 11)) < \mathrm{max}(0, 101 - \alpha) \ .$$

On a alors

$$D \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f\,(\mathsf{plus}\ x\ 11) \ : \ \forall\delta\, Q_f(\epsilon(x) + 11, \delta).\ \mathsf{Nat}^{\delta} \ .$$

D'après le lemme 11.3.16, on obtient

$$D \ \wedge \ Q_f(\epsilon(x) + 11, \delta) \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f\,(\mathsf{plus}\ x\ 11) \ : \ \forall\delta\, Q_f(\epsilon(x) + 11, \delta).\ \mathsf{Nat}^{\delta} \ ,$$

et en appliquant la règle $(\forall\,\mathrm{E})$ *on a*

$$D \ \wedge \ Q_f(\epsilon(x) + 11, \delta) \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f\,(\mathsf{plus}\ x\ 11) \ : \ \mathsf{Nat}^{\delta} \ .$$

Or, on vérifie facilement que

$$\vdash \ (D \ \wedge \ Q_f(\epsilon(x) + 11, \delta)) \ \supset \ \mathrm{max}(0, 101 - \delta) < \mathrm{max}(0, 101 - \alpha) \ ,$$

et on obtient

$$D \ \wedge \ Q_f(\epsilon(x) + 11, \delta) \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f \ : \ \mathsf{Nat}^{\delta} \Rightarrow \forall\gamma\, Q_f(\delta, \gamma).\ \mathsf{Nat}^{\gamma} \ ,$$

d'où

$$(\forall\,\mathrm{I})\ \frac{D \ \wedge \ Q_f(\epsilon(x) + 11, \delta) \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f\,(f\,(\mathsf{plus}\ x\ 11)) \ : \ \forall\gamma\, Q_f(\delta, \gamma).\ \mathsf{Nat}^{\gamma}}{D \ ; \ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash \ f\,(f\,(\mathsf{plus}\ x\ 11)) \ : \ \forall\delta\, Q_f(\epsilon(x) + 11, \delta).\ \forall\gamma\, Q_f(\delta, \gamma).\ \mathsf{Nat}^{\gamma}}$$

On en déduit que

$$\mathsf{D}\ ;\ x : \mathsf{Nat}^{\epsilon(x)} \ \vdash\ \mathsf{f}\,(\mathsf{f}\,(\mathsf{plus}\ x\ 11))\ :\ \forall\zeta\, Q_{\mathsf{f}}(\alpha,\zeta).\ \mathsf{Nat}^{\zeta}$$

car

$$\vdash\ \mathsf{D}\ \supset\ \|\forall\delta\, Q_{\mathsf{f}}(\epsilon(x)+11,\delta).\ \forall\gamma\, Q_{\mathsf{f}}(\delta,\gamma).\ \mathsf{Nat}^{\gamma}\ \sqsubseteq\ \forall\zeta\, Q_{\mathsf{f}}(\alpha,\zeta).\ \mathsf{Nat}^{\zeta}\| \ .$$

Ainsi, par le théorème 11.4.38 (resp. le théorème 11.4.40), la relation $\rightarrow_{\mathcal{S}91}$ est fortement normalisante sur les termes typables de $\boldsymbol{\lambda}_{\mathcal{S}91}(\mathcal{B}_0, \Phi(\mathcal{P}, \mathfrak{S}, \mathcal{V}\!s), \Sigma_0, \tau_{\mathcal{S}})$ (resp. les termes typables de $\boldsymbol{\lambda}_{\underline{\mathcal{S}}91}(\mathcal{B}_0, \Sigma_0, \underline{\tau_{\mathcal{S}}})$).

Quatrième partie

Épilogue

Conclusion

Dans cette thèse, nous avons étudié la combinaison de la réécriture et du lambda-calcul. Nous nous sommes concentrés sur deux propriétés, la confluence et la terminaison, qui permettent d'assurer respectivement la cohérence équationnelle du calcul et la cohérence déductive de systèmes de types.

Les questions auxquelles nous amènent nos travaux peuvent être regroupées selon trois axes de recherche.

— Étudier le lien (sémantique) entre les algèbres de termes du premier ordre et les termes applicatifs et algébriques, c'est à dire les termes construits avec un symbole d'application binaire. C'est un point important vis-à-vis des questions traitées à la partie II sur la confluence, mais aussi pour mieux comprendre ce qui est en jeu dans la normalisation de la combinaison typée de la réécriture du premier ordre et du lambda-calcul (voir section 9.4).

— Étudier la signification logique de notre travail sur la réductibilité et comprendre ses liens avec la sémantique dénotationnelle. Cela concerne essentiellement nos travaux sur la stabilité par union de familles de réductibilité (chapitre 10) et notre définition des candidats de réductibilité de Girard (section 9.3.4).

— Approfondir l'étude du critère de terminaison avec types contraints présenté à la section 11.

Partie II Confluence

Chapitre 5 Outils pour la confluence

Concernant la réécriture conditionnelle, nous avons rappelé au théorème 5.2.6 que la réécriture semi-équationnelle orthogonale est confluente et que la réécriture normale orthogonale est confluente par niveaux. Comme nous l'avons dit à la remarque 5.2.7, il nous semble possible que la réécriture conditionnelle normale soit de plus confluente en profondeur. De plus, il est connu que la réécriture conditionnelle par joignabilité orthogonale n'est en général pas confluente. Il serait intéressant de savoir si la condition de semi-clôture permet d'avoir la confluence de la réécriture conditionnelle par joignabilité orthogonale.

D'autre part, nous avons évoqué à la section 5.3.4 un lien possible entre la non modularité de la confluence pour les combinaisons non disjointes et la non préservation de la confluence par ajout aux termes d'éléments issus d'une algèbre non libre. Il nous semble intéressant de voir précisément en quoi la non modularité de la confluence pour des systèmes non disjoints est liée au fait qu'un des deux systèmes de réécriture simule des

termes infinis, via par exemple la notion d'arbres de Böhm pour la réécriture développée dans [KOV99].

Chapitre 7 Réécriture conditionnelle combinée au lambda-calcul

Les travaux présentés dans ce chapitre posent un certain nombre de questions sur le lien entre les termes du premier ordre et les termes algébriques (c.-à-d. construits avec un symbole d'application).

En effet, hormis pour la combinaison du lambda-calcul à la réécriture conditionnelle linéaire à gauche et semi-close (sans β-réduction dans les conditions), tous nos résultats ont été montrés pour des termes vérifiant des conditions liées à l'arité applicative des symboles (c.-à-d. au nombre d'arguments qui leurs sont appliqués). Ces conditions sont un moyen de récupérer de l'information algébrique au niveau des lambda-termes. Notons que les résultats de [BTG94], qui sont dans un cadre typé, utilisent le fait que les systèmes de réécriture sont algébriquement typés, et qu'on a vu à la remarque 3.6.6 que les termes curryfiées algébriquement typés peuvent être vus comme des termes du premier ordre.

Il serait donc intéressant d'étudier de manière précise la condition de $(\mathcal{R}, \mathfrak{a})$-stabilité, en particulier en termes de typage, et en ce qui concerne la confluence de la réécriture β-conditionnelle par joignabilité, d'essayer de se passer de la condition que les termes respectent héréditairement une arité applicative.

Chapitre 8 Préservation de la confluence par curryfication

Nos résultats sur la préservation de la confluence par curryfication pour la réécriture conditionnelle ne sont pas totalement satisfaisants. En effet, ils n'étendent pas le résultat de Kahrs [Kah95] à la réécriture conditionnelle car nous supposons que les règles conditionnelle sont non effondrantes.

De manière plus générale, il nous semble important d'essayer de mieux comprendre la curryfication et les liens sémantiques entre les algèbres de termes du premier ordre et les lambda-termes algébriques.

Partie III Normalisation forte et réductibilité

Chapitre 9 Normalisation forte dans le lambda-calcul typé

La question de l'interaction entre les algèbres de termes du premier ordre et les lambda-termes se pose aussi pour la normalisation forte. Elle se manifeste au niveau des preuves de préservation de la normalisation forte pour la combinaison de la réécriture du premier ordre (ou curryfiée et algébriquement typée) au système F [BTG89, Oka89]. Il nous semble qu'il serait intéressant de voir si on peut reformuler ces preuves en utilisant de manière directe la notion de fermeture calculable [BJO02, Bla07]. Cela permettrait de mieux comprendre (et de manière plus abstraite), du point de vue de la normalisation forte, l'interaction entre les réductions « du premier ordre » et la β-réduction.

Chapitre 10 Stabilité par union de familles de réductibilité

Nous avons vu que la stabilité par union d'une famille de réductibilité prenant en compte une relation de réécriture $\rightarrow_{\mathsf{R}}$ correspond à un comportement calculatoire particulier de $\rightarrow_{\mathsf{R}}$. Cependant, nous ne savons pas quelle est l'interprétation logique de cette propriété. Cette question pourrait être posée à plusieurs niveaux : en termes de réalisabilité, au niveau des « truth values algebras » de Dowek [Dow06], et au niveau des lambda-calculs pour la logique classique, comme par exemple [Par97, CH00].

D'autre part, il serait intéressant d'étudier plus en profondeur la signification « sémantique » de la stabilité par union de familles de réductibilité. Pour cela, le système de type $\boldsymbol{\lambda}_{\sqcap\sqcup\mathcal{R}}(\Sigma)$ induit naturellement un modèle de filtres. Mais il faudrait essayer de comprendre quels sens ont nos conditions du réduit principal et du réduit principal par rapport aux modèles d'extensions du lambda-calcul, tels qu'étudiés par exemple dans [Bou94, DCLP96, DCLP98].

La question principale concernant la stabilité par union de familles de réductibilité est d'avoir une condition suffisante pour les systèmes de réécriture non simples (c.-à-d. avec filtrage) qui soit facilement vérifiable. Nous avons vu avec la réécriture simple qu'il existe des systèmes confluents qui n'admettent pas de famille de réductibilité stable par union. Par contre, il nous semble tout à fait possible que les systèmes de réécriture orthogonaux admettent des familles de réductibilité stables par union (les systèmes de réécriture simples orthogonaux ont des candidats de Girard stables par union, voir la remarque 10.5.7). Cela demanderait probablement une étude précise du théorème de standardisation de [HL91]. Un problème lié nous semble être l'obtention d'un système de types complet pour la normalisation forte de la réécriture orthogonale. Ce sujet pourrait être intéressant aussi pour faire le lien entre les modèles de filtres et les notions sémantiques issues de la standardisation [HL91, Bou85, Mel98].

Enfin, notre étude de la stabilité par union des candidats de Girard nous a permis de mettre en évidence une structure intéressante de ceux-ci : les candidats de Girard sont stables par union exactement lorsque ce sont les ensembles non vides de termes fortement normalisants et clos par le bas pour un ordre d'observation faible.

Cette caractérisation pose au moins deux questions. La première est de voir si une telle structure « algébrique » simple existe pour les ensembles saturés de Tait. La seconde est de comprendre ce que cette caractérisation veut dire en termes d'identification de preuves (ou bien en termes de « sémantique dénotationnelle » pour les preuves).

Cette seconde question est sûrement en rapports étroits avec la question du lien entre notre définition des termes neutres par interaction entre termes et contextes et les sémantiques dénotationnelles pour preuves basées « officiellement » sur l'interaction, comme par exemple la sémantique des jeux [HO00].

Chapitre 11 Types contraints pour la normalisation forte

Une première question importante est l'étude précise de la complexité de la résolution des contraintes générées par le processus de vérification du typage.

De plus nous avons travaillé sur une ébauche de prototype pour ce critère de ter-

minaison. Nous avons implanté la vérification du typage presque dans son intégralité. Cependant, nous n'avons pas encore traité le critère de terminaison en tant que tel.

Bien que nous ayons pensé notre système de types contraints comme une version implicite du système de types dépendants de [Xi98], nous n'avons pas étudié précisément cette question. En particulier, il est serait intéressant de comparer notre vérification du typage avec le processus d'élaboration de [Xi98].

Il pourrait aussi être intéressant d'étudier des restrictions du système. Nous ne sommes pas sûr que la gestion des types contraints dans toute leur généralité soit réellement utile. Des restrictions sur la forme des types contraints pourraient être intéressantes. En particulier, il est possible que certains travaux sur la programmation logique dans des systèmes avec preuves uniformes [MNPS91] puissent donner des restrictions intéressantes sur la forme des types pour simplifier la forme des contraintes générées par la vérification du typage.

D'autre part, le système ne traite que les types inductifs du premier ordre, et il serait intéressant de l'étendre aux types inductifs positifs. Cela nécessiterait probablement une extension de la syntaxe et de la sémantique des contraintes, en particulier si on veut une interprétation singleton des types inductifs (ce qui semble toutefois difficile à obtenir).

La façon dont est traitée la réécriture conditionnelle est relativement intéressante car le système permet de manipuler l'information issue de la satisfaction de conditions booléennes. Cela nous permet de traiter le cas de la fonction 91 de McCarthy. Cependant, la réécriture conditionnelle offre des perspectives plus larges, qu'il serait intéressant d'étudier. Citons l'utilisation de conditions sur des types autres que les booléens et l'utilisation dans les conditions de variables « extra » (c.-à-d. instantiées par filtrage lors de l'évaluation des conditions).

Enfin, il pourrait être intéressant d'un point de vue théorique de voir ce que l'on pourrait faire en utilisant des contraintes dans un domaine issu de la « sémantique dénotationnelle ». En particulier, il se pourrait que l'argument utilisé dans les preuves de terminaison de [BBC98, Ber05] puisse être formulé au niveau du typage, en utilisant un système de contraintes approprié (mais pas forcément décidable).

Bibliographie

[Abe03] A. Abel – « Termination and Productivity Checking with Continuous Types », *Proceedings of TLCA'03*, 2003, p. 1–15. 250

[Abe04] — , « Termination Checking with Types », *RAIRO – Theoretical Informatics and Applications* **38** (2004), no. 4, p. 277–319, Special Issue (FICS'03). 160, 164, 244, 250, 262

[Abe06] — , « Semi-Continuous Sized Types and Termination », *Proceedings of CSL'06*, LNCS, vol. 4207, Springer, 2006, p. 72–88. 218, 250

[AC98] R. M. Amadio et P.-L. Curien – *Domains and Lambda-Calculi*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1998. 15, 202, 205, 220

[ALS94] J. Avenhaus et C. Loría-Sáenz – « Higher Order Conditional Rewriting and Narrowing », *Proceedings of the 1st International Conference on Constraints in Computational Logics*, LNCS, vol. 845, Springer Verlag, 1994, p. 269–284. 84

[Alt93] T. Altenkirch – « Constructions, Inductive Types and Strong Normalization », Thèse, University of Edinburgh, 1993. 160, 165

[Bar84] H.P. Barendregt – *The Lambda-Calculus, its Syntax and Semantics*, Studies in Logic and the Foundation of Mathematics, North Holland, 1984, Second edition. 12, 27, 37, 46, 68, 77, 78, 79, 83, 102

[Bar92] — , « Lambda Calculi with Types », Handbook of Logic in Computer Science (S. Abramsky, D. Gabbay et T. Maibaum, éds.), vol. 2, Oxford University Press, 1992. 68, 74, 163, 164, 179

[BBC98] S. Berardi, M. Bezem et T. Coquand – « On the Computational Content of the Axiom of Choice », *Journal of Symbolic Logic* **63** (1998), no. 2, p. 600–622. 14, 296

[BBF96] S. van Bakel, F. Barbanera et M. Fernández – « Rewrite Systems with Abstraction and β-rule : Types, Approximants and Normalisation », *Proceedings of ESOP'96*, 1996, p. 387–403. 203

[BCD+06] B. Borovanský, H. Cirstea, E. Deplagne, H. Dubois, C. Kirchner, H. Kirchner, P.-E. Moreau, Q.-H. Nguyen, C. Ringeissen et M. Vittek – *Elan User Manual*, 2006, `http://elan.loria.fr/`. 14

[BDCL95] F. Barbanera, M. Dezani-Ciancaglini et U. de' Liguoro – « Intersection and Union Types : Syntax and Semantics », *Information and Computation* **119** (1995), p. 202–230. 202, 212, 213

[Ber05] U. Berger – « Continuous Semantics for Strong Normalization », *Proceedings of CiE'05*, vol. 3526, LNCS, 2005, p. 23–34. 296

[BF96] F. Barbanera et M. Fernández – « Intersection Types Assignment Systems with Higher-Order Algebraic Rewriting », *Theoretical Computer Science* **170** (1996), no. 1–2, p. 173–207. 114, 198, 203

[BF97] S. van Bakel et M. Fernández – « Normalization Results for Typeable Rewrite Systems », *Information and Computation* **133** (1997), no. 2, p. 73–116. 203

[BFG97] F. Barbanera, M. Fernández et H. Geuvers – « Modularity of Strong Normalization and Confluence in the Algebraic-lambda-Cube », *Journal of Functional Programming* **7** (1997), no. 6, p. 613–660. 114, 124, 126, 198

[BFG+04] G. Barthe, M. J. Frade, E. Giménez, L. Pinto et T. Uustalu – « Type-Based Termination of Recursive Definitions », *Mathematical Structures in Computer Science* **14** (2004), no. 1, p. 97–141. 250

[BGP05] G. Barthe, B. Grégoire et F. Pastawski – « Practical Inference ofr Type-Based Termination in a Polymorphic Setting », *Proceedings of TLCA'05*, 2005, p. 71–85. 250

[BGP06] — , « Type-Based Termination of Recursives Definitions in the Calculus of Inductive Constructions », *Proceedings of LPAR'06*, 2006, p. 257–271. 250

[BJO02] F. Blanqui, J.-P. Jouannaud et M. Okada – « Inductive-Data-Types Systems », *Theoretical Computer Science* **271** (2002), p. 41–68. 171, 172, 173, 244, 248, 249, 278, 294

[BJR07] F. Blanqui, J.-P. Jouannaud et A. Rubio – « HORPO with Computability Closure : A Reconstruction », *Proceedings of LPAR'07*, 2007, p. 138–150. 244

[BK86] J.A. Bergstra et J.W. Klop – « Conditional rewrite rules : Confluence and termination », *Journal of Computer and System Sciences* **32** (1986), no. 3, p. 323–362. 104

[BKR05a] E. Bonelli, D. Kesner et A. Ríos – « de Bruijn Indices for Metaterms », *Journal of Logic and Computation* **15** (2005), no. 6, p. 855–899. 92

[BKR05b] — , « Relating Higher-Order and First-Order Rewriting », *Journal of Logic and Computation* **15** (2005), no. 6, p. 901–947. 92

[BKR06] F. Blanqui, C. Kirchner et C. Riba – « On the Confluence of Lambda-Calculus with Conditional Rewriting », *Proceedings of FoSSaCS'06*, LNCS, vol. 3921, 2006. 17, 119, 130

[Bla04] F. Blanqui – « A Type-Based Termination Criterion for Dependently-Types Higher-Order Rewrite Systems », *Proceedings of RTA'04*, 2004, p. 24–39. 239, 250, 251

[Bla05a] — , « Decidability and Type-Checking in the Calculus of Algebraic Constructions with Size Annotations », *Proceedings of CSL'05*, 2005, p. 135–150. 250

[Bla05b] — , « Definitions by Rewriting in the Calculus of Constructions », *Mathematical Structures in Computer Science* **15** (2005), no. 1, p. 37–92. 114, 188, 194, 198, 266, 268, 278

[Bla06a] — , « Higher-Order Dependancy Pairs », *Proceedings of WST'06*, disponible sur HAL, 2006. 244

[Bla06b] — , « (HO)RPO Revisited », Tech. Report 5972, INRIA, 2006. 244

[Bla07] — , « Computability Closure : Ten Years Later », *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of his 60th Birthday* (H. Comon, C. Kirchner et H. Kirchner, éds.), LNCS, vol. 4600, Springer, 2007, p. 68–88. 171, 172, 173, 248, 294

[BN98] F. Baader et T. Nipkow – *Term Rewriting and All That*, Cambridge University Press, 1998. 27

[Bou85] G. Boudol – « Computationel Semantics of Term Rewriting Systems », Algebraic Methods in Semantics (M. Nivat et J. Reynolds, éds.), Cambridge University Press, 1985. 295

[Bou94] — , « Lambda-Calculi for (Strict) Parallel Functions », *Information and Computation* **108** (1994), no. 1, p. 51–127. 295

[BR06] F. Blanqui et C. Riba – « Combining Typing and Size Constraints for Checking the Termination of Higher-Order Conditional Rewrite Systems », *Proceedings of LPAR'06*, LNAI, vol. 4246, 2006. 18, 172, 173, 240, 244, 256

[BT88] V. Breazu-Tannen – « Combining Algebra and Higher-Order Types », *Proceedings of LiCS'88*, IEEE Computer Society, 1988. 16, 111, 114, 117

[BTG89] V. Breazu-Tannen et J. Gallier – « Polymorphic Rewriting Conserves Algebraic Strong-Normalization and Confluence », *Proceedings of ICALP'89*, 1989. 16, 111, 114, 117, 118, 198, 294

[BTG94] — , « Polymorphic Rewriting Conserves Algebraic Confluence », *Information and Computation* **114** (1994), no. 1, p. 1–29. 16, 111, 117, 118, 147, 294

[BTM87] V. Breazu-Tannen et A. Meyer – « Computable Values Can Be Classical », *Proceedings of POPL'87*, ACM, 1987. 111, 114, 119

[CD78] M. Coppo et M. Dezani – « A New Type-Assignement for λ-terms », *Archive of Mathematical Logic* **19** (1978), p. 139–156. 13, 202

[CDE+07] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer et C. Talcott – *Maude Manual (Version 2.3)*, 2007, http://maude.cs.uiuc.edu/. 14

[CH88] T. Coquand et G. Huet – « The Calculus of Constructions », *Information and Computation* **76** (1988), no. 2/3, p. 95–120. 14

[CH00] P.-L. Curien et H. Herbelin – « The Duality of Computation », *Proceedings of ICFP'00*, ACM, 2000, p. 233–243. 295

[CJ96] H. Comon et J.-P. Jouannaud – « Les Termes en Logique et en Programmation », 1996, Notes de cours de DEA. 27, 30

[Coq] C. Coquand – *Agda*, http://www.cs.chalmers.se/~catarina/agda/. 13

[CP88] T. Coquand et C. Paulin – « Inductively defined types », *Conference on Computer Logic*, LNCS, vol. 417, Springer, 1988, p. 50–66. 14

[CS06] T. Coquand et A. Spiwack – « A Proof of Strong Normalisation using Domain Theory », *Proceedings of LiCS'06*, IEEE Computer Society, 2006, p. 307–316. 202, 203, 205

[DCK94] R. Di Cosmo et D. Kesner – « Combining First Order Algebraic Rewriting Systems, Recursion and Extensional Lambda Calculi », *Proceedings of ICALP*, 1994, p. 462–472. 111, 117

[DCK96] — , « Combining First Order Algebraic Rewriting Systems, Recursion and Extensional Lambda Calculi », *Theoretical Computer Science* **169** (1996), no. 2, p. 201–220. 111, 117

[DCLP96] M. Dezani-Ciancaglini, U. de' Liguoro et P. Piperno – « Filter Models for Conjunctive-Disjunctive Lambda-Calculi », *Theoretical Computer Science* **170** (1996), no. 1-2, p. 83–128. 208, 268, 295

[DCLP98] — , « A Filter Model for Concurrent Lambda-Calculus », *Siam Journal on Computing* **27** (1998), no. 5, p. 1376–1419. 208, 268, 295

[DHK00] G. Dowek, T. Hardin et C. Kirchner – « Higher Order Unification via Explicit Substitutions », *Information and Computation* **157** (2000), no. 1-2, p. 183–235. 47, 49

[DHK03] — , « Theorem Proving Modulo », *Journal of Automated Reasoning* **31** (2003), no. 1, p. 33–72. 14

[DJ90] N. Dershowitz et J.-P. Jouannaud – « Rewrite Systems », Handbook of Theoretical Computer Science, Volume B : Formal Models and Sematics (B) (J. van Leeuwen, éd.), North-Holland, 1990, p. 243–320. 12, 32, 94

[DK00] V. Danos et J.-L. Krivine – « Disjunctive Tautologies as Synchronisation Schemes », *Proceedings of CSL'00*, LNCS, vol. 1862, 2000, p. 292–301. 194

[DO90] N. Dershowitz et M. Okada – « A rationale for conditional equational programming », *Theoretical Computer Science* **75** (1990), p. 111–138. 88

[Dou92] D.J. Dougherty – « Adding Algebraic Rewriting to the Untyped Lambda Calculus », *Information and Computation* **101** (1992), no. 2, p. 251–267. 17, 111, 115, 116, 122, 125, 147

[Dow06] G Dowek – « Truth Values Algebras and Proof Normalization », *Post Proceedings of TYPES'06*, LNCS, Springer, 2006, p. 110–124. 295

[DP00] R. Davis et F. Pfenning – « Intersection Types and Computational Effects », *Proceedings of ICFP'00*, ACM, 2000, p. 198–208. 262

[FR74] M. J. Fisher et M. O. Rabin – « Super-Exponential Complexity of Presburger Arithmetic », *Proceedings on the SIAM-AMS Symposium in Applied Mathematics*, vol. 7, 1974, p. 27–41. 265

[Gal89] J.H. Gallier – « On Girard's "Candidats de Reducibilité" », Logic and Computer Science (P. Odifredi, éd.), Academic Press, 1989. 70, 74, 164, 179, 187

[Gal91] — , « What's So Special About Kruskal's Theorem and the Ordinal Γ_0 ? A Survey of Some Results in Proof Theory », *Annals of Pure and Applied Logic* **53** (1991), no. 3, p. 199–260. 22, 190

[Gal98] — , « Typing Untyped Lambda-Terms, or Reducibility Strikes Again ! », *Annals of Pure and Applied Logic* **91** (1998), p. 231–270. 202, 205, 210

[Gir72] J.-Y. Girard – « Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur », Thèse, Université Paris 7, 1972. 17, 174, 175, 178, 179, 187

[Gir87] — , « Linear Logic », *Theoretical Computer Science* **50** (1987), p. 1–102. 194

[GKK05] J. Glauert, D. Kesner et Z. Khasidashvili – « Expression Reduction Systems and Extensions : An Overview », *Processes, Terms and Cycles : Steps to the Road of Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion os His 60th Birthday*, LNCS, vol. 3838, Springer, 2005, p. 496–553. 84

[GLT89] J.-Y. Girard, Y. Lafont et P. Taylor – *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1989. 68, 74, 75, 83, 163, 170, 179, 187, 188

[GM88] E. Giovannetti et C. Moiso – « Notes on the Elimination of Conditions », *Proceedings of CTRS'87*, LNCS, vol. 308, Springer, 1988, p. 91–97. 140

[GTWW77] J. A. Goguen, J. W. Thatcher, E. G. Wagner et J. B. Wright – « Initial Algebra Semantics and Continuous Algebras », *Journal of the Association for Computing Machinery* **4** (1977), no. 1, p. 68–95. 27, 30

[HL91] G. Huet et J.-J. Lévy – « Computations in Orthogonals Systems, I & II », Computational Logic, Essays in Honour of Alan Robinson (J. L. Lassez et G. Plotkin, éds.), MIT Press, 1991. 13, 99, 295

[HM99] R. Harper et J.C. Mitchell – « Parametricity and Variants of Girard's *J* Operator », *Information Processing Letters* **70** (1999), no. 1, p. 1–5. 174, 175

[HO00] J. M. E. Hyland et C.-H. Ong – « On Full Abstraction for PCF : I, II, and III », *Information and Computation* **163** (2000), no. 2, p. 285–408. 220, 295

[HPS96] J. Hughes, L. Pareto et A. Sabry – « Proving the Correctness of Reactive Systems Using Sized Types », *Proceedings of POPL'96*, ACM, 1996, p. 410–423. 250

[Hue80] G. Huet – « Confluent Reductions : Abstract Properties and Applications to Term Rewriting Systems », *Journal of the Association for Computing Machinery* **27** (1980), no. 4, p. 797–821. 94, 99, 101

[Hue86] — , « Formal Structures for Computation and Deduction », Technical report, INRIA, Rocquencourt, 1986. 91

[JK86] J.-P. Jouannaud et H. Kirchner – « Completion of a Set of Rules Modulo a Set of Equations », *Siam Journal on Computing* **25** (1986), no. 4, p. 1155–1194. 94

[JM96] T. Jim et A. Meyer – « Full Abstraction and the Context Lemma », *Siam Journal on Computing* **25** (1996), no. 3, p. 663–696. 220, 268

[JO91] J.-P. Jouannaud et M. Okada – « Executable Higher-Order Algebraic Specification Languages », *Proceedings of LiCS'91*, IEEE Computer Society, 1991. 114, 244, 249

[JO97] J.-P. Jouannaud et M. Okada – « Abstract Data Type Systems », *Theoretical Computer Science* **173** (1997), no. 2, p. 349–391. 244

[Jou06] J.-P. Jouannaud – « Modular Church-Rosser Modulo », *Proceedings of RTA'06*, LNCS, no. 4098, Springer, 2006, p. 96–107. 106, 108

[JR99] J.-P. Jouannaud et A. Rubio – « The Higher-Order Recursive Path Ordering », *Proceedings of LiCS'99*, IEEE Computer Society, 1999, p. 402–411. 244

[JR07] — , « Polymorphic Higher-Order Recursive Path Orderings », *Journal of the Association for Computing Machinery* **54** (2007), no. 1, p. 1. 244

[Kah95] S. Kahrs – « Confluence of Curried Term-Rewriting Systems », *Journal of Symbolic Computation* **19** (1995), p. 601–623. 16, 17, 145, 147, 152, 153, 294

[Kap84] S. Kaplan – « Conditional Rewrite Rules », *Theoretical Computer Science* **33** (1984), p. 175–193. 89

[KKSV96] R. Kennaway, J.W. Klop, R. Sleep et F.-J. de Vries – « Comparing Curried and Uncurried Rewriting », *Journal of Symbolic Computation* **21** (1996), no. 1, p. 15–39. 199

[Klo80] J.W. Klop – *Combinatory Reduction Systems*, Mathematical Center Tracts, vol. 127, CWI, 1980, PhD Thesis. 83, 84, 111

[KMTV94] J.W. Klop, A. Middeldorp, Y. Toyama et R.C. de Vrijer – « Modularity of Confluence : A Simplified Proof », *Information Processing Letters* **49** (1994), p. 101–109. 106, 148

[KOR93] J.W. Klop, V. van Oostrom et F. van Raamsdonk – « Combinatory Reduction Systems : Introduction and Survey », *Theoretical Computer Science* **121** (1993), p. 279–308. 84

[KOV99] R. Kennaway, V. van Oostrom et F.-J. de Vries – « Meaningless Terms in Rewriting », *Journal of Functional and Logic Programming* **1999** (1999), no. 1, p. 1–35. 106, 109, 112, 294

[Kri90] J.-L. Krivine – *Lambda-Calcul, Types et Modèles*, Masson, 1990. 12, 15, 27, 36, 37, 39, 68, 75, 79, 102, 163, 164, 179, 202, 205

[Kri04] — , « Realizability in Classical Logic », à paraître dans *Panoramas et synthèses*, Société Mathématique de France, disponible sur HAL, 2004. 194

[KS07] K. Kusakari et M. Sakai – « Enhancing Dependency Pair Method Using Strong Computability in Simply-Typed Rewriting », *Applicable Algebra in Engineering, Communication and Computing* **18** (2007), p. 407–431. 244

[KW97] U. Kühler et C.-P. Wirth – « Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving », *Proceedings of RTA'97*, LNCS, vol. 1232, Springer, 1997, p. 38–52. 140

[LDG+07] X. Leroy, D. Doligez, J. Garrique, D. Rémy et J. Vouillon – *The Objective Caml System Release 3.10*, 2007, `http://caml.inri.fr/`. 13

[Luo90] Z. Luo – « An Extended Calculus of Constructions », Thèse, University of Edinburgh, 1990. 160

[Mat98] R. Matthes – « Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types », Thèse, Ludwig-Maximilians-Universität München, 1998. 163

[Mat00] — , « Lambda Calculus : A Case for Inductive Definitions », Lecture notes for ESSLLI'00, 2000. 251

[Mat05] — , « Non-Strictly Positive Fixed-Points for Classical Natural Deduction », *Annals of Pure and Applied Logic* **133** (2005), p. 205–230. 160, 164

[Mel98] P.-A. Melliès – « A Stability Theorem in Rewriting Theory », *Proceedings of LiCS'98*, IEEE Computer Society, 1998, p. 287–299. 295

[Men87] N. P. Menler – « Recursive Types and Type Constraints in Second Ordel Lambda-Calculus », *Proceedings of LiCS'87*, IEEE Computer Society, 1987, p. 30–36. 245, 247, 248

[Mid89] A. Middeldorp – « A Sufficient Condition for the Termination of the Direct Sum of Term Rewriting Systems », *Proceedings of LiCS'89*, IEEE Computer Society, 1989, p. 396–401. 199

[Mid91] — , « Confluence of the Disjoint Union of Conditional Term Rewriting Systems », *Proceedings of CTRS'91*, LNCS, vol. 516, 1991, p. 295–306. 106

[Mil77] R. Milner. – « Fully Abstract Models of Typed λ-Calculi », *Theoretical Computer Science* **4** (1977), p. 1–22. 220

[Miq01] A. Miquel – « Le Calcul des Constructions Implicite : Syntaxe et Sémantique », Thèse, Université Paris 7, 2001. 68, 70

[ML84] P. Martin-Löf – *Intuitionistic Type Theory*, Bibliopolis, Napoli, 1984. 11

[MNPS91] D. Miller, G. Nadathur, F. Pfenning et A. Scedrov – « Uniform Proofs as a Foundation for Logic Programming », *Annals of Pure and Applied Logic* **52** (1991), no. 1/2, p. 125–157. 296

[MPS86] D. MacQueen, G. Plotkin et R. Sethi – « An Ideal Model for Recursive Polymorphic Types », *Information and Control* **71** (1986), no. 1-2, p. 95–130. 202, 212, 256, 260

[MT92] K. Meinke et J.V. Tucker – « Universal Algebra », Handbook of Logic in Computer Science (S. Abramsky, D. Gabbay et T. Maibaum, éds.), vol. 1, Oxford University Press, 1992. 27, 29

[MTHM97] R. Milner, M. Tofte, R. Harper et D. MacQueen – *The Defintion of Standard ML, Revised Edition*, MIT Press, 1997. 13

[Mül92] F. Müller – « Confluence of the Lambda Calculus with Left Linear Algebraic Rewriting », *Information Processing Letters* **41** (1992), p. 293–299. 111, 112, 113, 121, 122

[MV05] P.-A. Melliès et J. Vouillon – « Recursive Polymorphic Types and Parametricity in an Operational Framework », *Proceedings of LiCS'05*, IEEE Computer Society, 2005, p. 82–91. 194, 268

[Nip91] T. Nipkow – « Higher-Order Critical Pairs », *Proceedings of LiCS'91*, IEEE Computer Society, 1991. 84

[NP98] T. Nipkow et C. Prehofer – « Higher-Order Rewriting and Equational Reasoning », Automated Deduction — A Basis for Applications. Volume I : Foundations (W. Bibel et P. Schmitt, éds.), Kluwer, 1998. 84

[Ohl02] E. Ohlebusch – *Advanced Topics in Term Rewriting*, Springer, April 2002. 88, 89, 91, 94, 104, 109, 139, 141, 239

[Oka89] M. Okada – « Strong Normalizability for the Combined System of the Typed Lambda-Calculus and an Arbitrary Convergent Term Rewrite System », *Proceedings of ISSAC'89*, ACM, 1989, p. 357–363. 111, 198, 294

[Oos94] V. van Oostrom – « Confluence for Abstract and Higher-Order Rewriting », Thèse, Vrije Universiteit, Amsterdam, 1994. 84, 109

[Oos02] J. van Oosten – « Realizability : A Historical Essay », *Mathematical Structures in Computer Science* **12** (2002), no. 3, p. 239–263. 12

[OR94] V. van Oostrom et F. van Raamsdonk – « Weak Orthogonality Implies Confluence : the Higher-Order Case », *Proceedings of LFCS'94*, LNCS, vol. 813, 1994. 84, 99, 113

[Par89] M. Parigot – « On the Representation of Data in Lambda-Calculus », *Proceedings of CSL'89*, LNCS, vol. 440, 1989, p. 309–321. 78

[Par97] — , « Proofs of Strong Normalization for Second Order Classical Natural Deduction », *Journal of Symbolic Logic* **62** (1997), no. 4, p. 1461–1479. 194, 295

[Pit00] A. M. Pitts – « Parametric Polymorphism and Operational Equivalence », *Mathematical Structures in Computer Science* **10** (2000), p. 321–359. 194

[PJ03] S. Peyton-Jones – « Special Issue : Haskell 98 Language and Libraries », *Journal of Functional Programming* **13** (2003), no. 1, p. 0–255. 13

[Plo77] G. Plotkin – « LCF Considered as a Programming Language », *Theoretical Computer Science* **5** (1977), p. 223–256. 268

[Pol93] J. van de Pol – « Termination Proofs for Higher-Order Rewrite Systems », *HOA*, 1993, p. 305–325. 244

[Pol96] — , « Termination of HIgher-Order Rewrite Systems », Thèse, Universiteit Utrecht, 1996. 244

[Pot80] G. Pottinger – « A Type Assignement to the Strongly Normalizable λ-Terms », To H. B. Curry : Essays on Combinatory Logic, Lambda-Calculus and Formalism (J. P. Seldin et J. R. Hindley, éds.), Academic Press, 1980, p. 561–577. 205

[Pre29] M. Presburger – « Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt », *Comptes Rendus du 1er congrès de Mathématiciens des Pays Slaves, Warszawa*, 1929, p. 92–101. 265

[Raa96] F. von Raamsdonk – « Confluence and Normalisation for Higher-Order Rewriting », Thèse, Vrije Universiteit, Amsterdam, 1996. 84

[Rib07a] C. Riba – « On the Stability by Union of Reducibility Candidates », *Proceedings of FoSSaCS'07*, LNCS, vol. 4423, 2007. 18, 201, 202

[Rib07b] — , « Strong Normalization as Safe Interaction », *Proceedings of LiCS'07*, IEEE Computer Society, 2007, p. 13–22. 18, 194, 201, 202

[Ros73] B. Rosen – « Tree-Manipulating Systems and Church-Rosser Theorems », *Journal of the Association for Computing Machinery* **20** (1973), no. 1, p. 160–187. 54

[RS95] F. von Raamsdonk et P. Severi – « On Normalisation », Tech. Report CS-R9545, CWI, 1995. 236

[Sco75] D. Scott – « Lambda Calculus and Recursion Theory », *Proc. of the third Scandinavian Logic Symposium* (S. Kanger, éd.), North Holland, 1975, p. 154–193. 83

[SP07] V. Simonet et F. Pottier – « A Constraint-Based Approach to Guarded Algebraic Data Types », *ACM Transactions on Programming Languages and Systems* **29** (2007), no. 1, p. 1. 262

[Tak93] M. Takahashi – « Lambda-Calculi with Conditional Rules », *Proceedings of TLCA'93*, LNCS, Springer-Verlag, 1993, p. 406–417. 140

[Tak95] — , « Parallel Reductions in λ-Calculus », *Information and Computation* **118** (1995), p. 120–127. 102, 130

[Tat07] M. Tatsuta – « Simple Saturated Sets for Disjunction and Second-Order Existential Quantification », *Proceedings of TLCA'07*, LNCS, vol. 4583, Springer, 2007, p. 366–380. 227

[Tea06] The Coq Developement Team – *The Coq Proof Assistant Reference Manual*, 2006, `http://coq.inria.fr/`. 13

[Ter03] Terese – *Term Rewriting Systems*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2003, M. Bezem, J.W. Klop and R.C. de Vrijer, eds. 12, 51, 58, 109, 110

[Toy87] Y. Toyama – « On the Church-Rosser Property for the Direct Sum of Term Rewriting Systems », *Journal of the Association for Computing Machinery* **34** (1987), no. 1, p. 128–143. 106, 148

[VM04] J. Vouillon et P.-A. Melliès – « Semantic Types : A Fresh Look at the Ideal Model for Types », *Proceedings of POPL'04*, ACM, 2004. 182, 194, 216, 232, 268

[Vou04] J. Vouillon – « Subtyping Union Types », *Proceedings of CSL'04*, LNCS, vol. 3210, Springer Verlag, 2004, p. 415–429. 194, 268

[Vri89] R.C. de Vrijer – « Extending the Lambda Calculus with Surjective Pairing is Conservative », *Proceedings of LiCS'89*, IEEE Computer Society, 1989, p. 204–215. 83, 92

[WC03a] D. Walukiewicz-Chrzaszcz – « Termination of rewriting in the calculus of constructions », Thèse, Warsaw University, Poland et Université Paris-Sud, France, 2003. 244

[WC03b] — , « Termination of rewriting in the calculus of constructions », *Journal of Functional Programming* **13** (2003), no. 2, p. 339–414. 244

[Wel99] J. Wells – « Typability and Type Checking in System F are Equivalent and Undecidable », *Annals of Pure and Applied Logic* **98** (1999), no. 1-3, p. 111–156. 70

[Wer94] B. Werner – « Une Théorie des Contructions Inductives », Thèse, Université Paris 7, 1994. 186, 218

[Wol93] D.A. Wolfram – *The Clausal Theory of Types*, Cambridge Tracts in Theoretical Computer Science, vol. 21, Cambridge University Press, 1993. 84

[Xi96] H. Xi – « Combining Algebraic Rewriting with Second Order Extentional Polymorphic Lambda Calculus », Unpublished Manuscript, 1996. 111, 117

[Xi98] — , « Dependent Types in Practical Programming », Thèse, Carnegie Mellon University, 1998. 239, 253, 258, 262, 296

[Xi02] — , « Dependent Types for Program Termination Verification », *Higher-Order and Symbolic Computation* **15** (2002), no. 1, p. 91–131. 240, 253, 258

Résumé

Cette thèse concerne la combinaison du lambda-calcul et de la réécriture, dont nous étudions principalement deux propriétés : la confluence et la normalisation forte.

Nous commençons par étudier sous quelles conditions la combinaison d'une relation de réécriture conditionnelle confluente au lambda-calcul donne une relation de réécriture confluente.

Ensuite nous nous intéressons aux preuves de normalisation forte de lambda-calculs typés utilisant la technique de réductibilité. Notre contribution la plus importante est une comparaison de diverses variantes de cette technique, utilisant comme outil de comparaison la manière dont ces variantes s'étendent à la réécriture et dont elles prennent en compte les types unions et les types existentiels implicites.

Enfin, nous présentons un critère de normalisation forte pour la réécriture conditionnelle combinée au lambda-calcul basé un système de types contraints. Notre approche, qui étend des critères de terminaison existants utilisant des annotations de taille, est à notre connaissance le premier critère de terminaison pour la réécriture conditionnelle avec membres droits d'ordre supérieur qui prenne en compte, dans l'argument de terminaison, de l'information issue de la satisfaction des conditions des règles de réécriture.

Mots clefs : Lambda-calcul, réécriture, réécriture conditionnelle, confluence, typage, types union, normalisation forte, candidats de réductibilité, ensembles saturés, biorthogonalité.

Abstract

This thesis is about the combination of lambda-calculus with rewriting. We mainly study two properties: confluence and strong normalization.

We begin by studying under which conditions the combination of a confluent conditional rewrite relation to the lambda-calculus leads to a confluent relation.

Next, we study strong normalization proofs of typed lambda-calculi that use the reducibility technique. Our main contribution is a comparison of variants of this technique, with respect to how they extend to rewriting and how they handle union and existential types.

Finally, we present a termination criterion for the combination of conditional rewriting and lambda-calculus based on a constrained type system. Our approach, which extend known criteria that use sized types, is to our knowledge the first termination criterion for conditional rewriting with higher-order right-hand sides that takes into account in the termination argument some information generated by the satisfaction of the conditions of the rewrite rules.

Keywords: Lambda-calculus, rewriting, conditional rewriting, typing, union types, strong normalization, reducibility candidates, saturated sets, biorthogonality.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur Thierry COQUAND, Professeur, Chalmers University of Technology, Göteborg, Sweden

Madame Délia KESNER, Professeur, Université Diderot, Paris 7, Paris

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur RIBA Colin

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Définitions par réécriture dans le *lambda*-calcul : confluence, réductibilité et typage"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **« Informatique »**

Fait à Vandoeuvre, le 03 décembre 2007
Le Président de l'I.N.P.L.,
F. LAURENT

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

TEL 33/03.83.59.59.59
FAX. 33/03.83.59.59.55