Combination methods for software verification

Daniele Zucchelli

▶ **To cite this version:**

Daniele Zucchelli. Combination methods for software verification. Other [cs.OH]. Université Henri Poincaré - Nancy 1, 2008. English. NNT : 2008NAN10006 . tel-01748341

HAL Id: tel-01748341
https://hal.univ-lorraine.fr/tel-01748341

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Dipartimento di Scienze dell'Informazione

DOTTORATO DI RICERCA IN INFORMATICA
XX CICLO

Settore scientifico-disciplinare INF/01

Combination Methods for Software Verification

Tesi di Dottorato di Ricerca di:
Daniele Zucchelli

Relatori:
Prof. Silvio Ghilardi
Dr. Michaël Rusinowitch

Correlatore:
Dr. Silvio Ranise

Coordinatore del Dottorato:
Prof. Vincenzo Piuri

ANNO ACCADEMICO 2006/2007

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Méthodes de Combinaison pour la Vérification de Logiciels

THÈSE

présentée et soutenue publiquement le 22 janvier 2008

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Daniele Zucchelli

Composition du jury

Président :	*Didier Galmiche*	
Rapporteurs :	*Alessandro Armando*	*Professeur à l'Université de Genova, Italie*
	Ahmed Bouajjani	*Professeur à l'Université Paris 7, France*
	Alessandro Cimatti	*Chercheur à l'ITC, Italie*
	Andreas Herzig	*Directeur de Recherche au CNRS, France*
Examinateurs :	*Didier Galmiche*	*Professeur à l'Université Henri Poincaré, France*
Directeurs :	*Silvio Ghilardi*	*Professeur à l'Université de Milano, Italie*
	Silvio Ranise	*Chargé de Recherche à l'INRIA Lorraine, France*
	Michaël Rusinowitch	*Directeur de Recherche à l'INRIA Lorraine, France*

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

Abstract

The thesis is devoted to the development of formal methods for software verification. Indeed, two are among the most widespread techniques that allow to rigorously specify the possible executions of a system and check whether it contains bugs. On the one hand, the correctness of a program can be guaranteed by showing the unsatisfiability of a formula modulo a theory which usually axiomatizes the involved datatypes; on the other hand, the model checking techniques are used to certify that every possible run of the system satisfies the desired properties. The contributions of the thesis are the following: First of all, we give a decidability result for the constraint satisfiability problem for interesting extensions of the theory of arrays. Secondly, along the lines of Manna and Pnueli, who have shown how a mixture of first-order logic and linear time temporal logic is sufficient to state the verification problems for the class of reactive systems, we draw on the recent literature about the combination of decision procedures to give decidability and undecidability results for the satisfiability problem for logics that allow to plug reasoning modulo first-order theories into a temporal setting. The results obtained in the case of linear flows of time are then generalized to the temporal and modal logics whose relativized satisfiability problem is decidable. The last contribution is the decidability of the model checking problem for linear flows of time under suitable hypothesis over the first-order theories involved. The proofs of the decidability results suggest that efficient Satisfiability Modulo Theories solvers might be successfully employed in the model checking of infinite-state systems.

Keywords: Combination Methods, Model Checking, Decision Procedures.

Résumé

Cette thèse est consacrée au développement de méthodes formelles pour la vérification de logiciels. Parmi les techniques les plus utilisées dans ce contexte, il y en a deux qui permettent une spécification rigoureuse de toutes les exécutions possibles d'un système et le contrôle des bogues cachés. D'un côté, la correction d'un programme peut être garantie en démontrant l'insatisfiabilité d'une formule modulo une théorie qui axiomatise les types de données impliqués ; de l'autre côté, les techniques de model-checking sont utilisées pour certifier que toute exécution possible du système satisfait les propriétés désirées. Les contributions de cette thèse sont les suivantes. Dans un premier temps, nous donnons un résultat de décidabilité pour la satisfiabilité de contraintes pour des extensions intéressantes de la théories des tableaux. Ensuite, nous avons obtenu des résultats dans le prolongement de Manna et Pnueli qui ont montré qu'un mélange de la logique du premier ordre et de la logique temporelle linéaire suffit pour énoncer les problèmes de vérification pour la classe des systèmes réactifs. Ainsi, nous nous inspirons de la récente littérature sur la combinaison des procédures de décision pour établir des résultats de décidabilité et d'indécidabilité pour le problème de satisfiabilité pour des logiques permettant d'intégrer du raisonnement modulo des théories du premier ordre dans un cadre temporel. Les résultats qu'on obtient pour la logiques temporelle linéaire sont ensuite généralisés au cas où le flux temporel est décrit par une logique dont le problème de la satisfiabilité relativisée est décidable. Notre dernière contribution est la décidabilité du problème du model-checking pour le flux temporel linéaire, sous des hypothèses appropriées concernant les théories du premier ordre impliquées. La preuve de ce résultat indique qu'on pourrait employer avec succès des Solveurs Modulo des Théories dans les applications du model-checking aux systèmes ayant un nombre infini d'états.

Mots clés : Méthodes de Combinaison, Model Checking, Procédures de Décisions.

Contents

Résumé Étendu

Le test et la simulation sont les techniques les plus répandues pour identifier les bogues. Ils nécessitent des expérimentations avant de pouvoir déployer le système. Dans le cas des circuits électroniques, ces méthodes injectent typiquement des signaux à certains moments et observent les signaux résultants, alors que pour les logiciels, la simulation et le test impliquent souvent l'entrée de certaines données, et l'observation des résultats correspondants. Ces deux méthodes ne permettent pas cependant de vérifier l'ensemble de *toutes* les interactions possibles, et donc la certification de l'absence de pièges est rarement possible. Comme ces deux méthodes se concentrent sur les chemins probables d'exécution (ou comportements), elles échouent dans la découverte de bogues révélés par des chemins ayant une faible probabilité d'exécution. Ces méthodes seules sont manifestement insuffisantes pour assurer la qualité à la fois des systèmes critiques et des logiciels que nous utilisons chaque jour.

Une alternative très attrayante à la simulation et au test est l'approche basée sur la *vérification formelle*. Les méthodes formelles concernent l'utilisation de techniques de logique et des mathématiques discrètes pour la spécification, la conception, la construction et l'analyse des systèmes informatiques et des logiciels. Elles semblent être le complément idéal au test, car elles peuvent examiner tous les comportements possibles du logiciel. La vérification d'un système se fait en fournissant une preuve formelle à l'abstraction du modèle formel du système, la correspondance entre le modèle formel et la nature du système étant connue par construction. Malgré leur promesse, les méthodes formelles ont des difficultés à être largement acceptées par l'industrie, principalement en raison du manque d'outils "prêts à l'usage" qui aident les développeurs de logiciels à les appliquer facilement, sans être noyés par un trop grand nombre de détails concernant les notations mathématiques ou les modèles. Augmenter le degré d'automatisation semble être la clé pour rendre les méthodes formelles économiquement réalisables et pour finalement être utilisées dans l'indus-

trie.

Le Contexte

L'objectif de la vérification formelle consiste à vérifier de manière formelle si un système satisfait certaines propriétés. Par conséquent, en vue de l'application des méthodes et des techniques provenant de ce domaine, le point de départ est de donner une représentation formelle de ce système et des propriétés devant être vérifiées. Le terme *système réactif* identifie généralement un système qui modifie ses actions, ses sorties et ses conditions/états en réponse à des stimuli de l'intérieur ou de l'extérieur. Un cadre commun pour la représentation des systèmes réactifs est fourni par les *systèmes de transition* ; de plus, les *logiques temporelles* sont des langages appropriés pour exprimer formellement les propriétés à vérifier. Les logiques temporelles sont souvent classées en fonction de la structure du temps, si elle est linéaire ou ramifiée. Dans le premier cas, il y a des logiques tels que la logique temporelle à temps linéaire (Linear time Temporal Logic LTL). Dans le second cas, il y a les logiques d'arbre de calcul (Computational Tree Logics CTL et CTL*), le μ-calcul etc. Dans Manna and Pnueli (1995) et de nombreux autres, les auteurs ont abondamment montré qu'un mélange de logique du premier ordre et LTL est suffisant pour décrire précisément les problèmes de vérification pour la catégorie des systèmes réactifs. Les théories en logique du premier ordre modélisent les structures de données (probablement infinies) utilisées par le système réactif alors que LTL spécifie son comportement dynamique. La combinaison de LTL et de la logique du premier ordre permet de spécifier les systèmes à états infinis et la façon subtile selon laquelle leurs flux de données influe le flux de contrôle.

On peut distinguer deux approches différentes pour la vérification formelle. La première approche, parfois appelée *vérification déductive*, consiste à utiliser une version formelle du raisonnement logique du système pour prouver son exactitude ; elle repose généralement sur des outils logiciels (appelés démonstrateurs de théorème). En général, ce processus n'est que partiellement automatisé et est dirigé par la compréhension de l'utilisateur du système de validation, cependant il s'appuie sur des procédures entièrement automatisées (appelées *procédures de décision*) pour certaines sous-problèmes. Bien que les limitations théoriques interdisent l'automatisation complète, un avantage de la vérification déductive est qu'elle peut être utilisé pour raisonner sur les systèmes à états infinis.

La seconde approche, appelée *vérification de modèle* (voir Merz, 2001 pour une vue d'ensemble et Burkart et al., 2001; Clarke et al., 1999 pour de plus amples détails à ce sujet), est une technique qui vérifie les systèmes concurrents à états finis en explorant de façon exhaustive le modèle mathématique utilisé pour représenter le système lui-même. Un des avantages de la vérification de modèle est que la vérification peut être totalement automatisée. Les techniques de vérification de modèle sont largement utilisées pour la vérification de systèmes à états finis, mais dans la dernière décennie les chercheurs ont fait de nombreux efforts pour étendre ces techniques à des systèmes à états infinis.

Le Problème de Satisfiabilité

Une technique standard pour la vérification déductive est basée sur la règle d'invariance déductive qui consiste à réduire la validité de certaines formules temporelles à la validité de formules de premier ordre modulo une théorie d'arrière-plan (voir Manna and Pnueli, 1995). Ceci suggère qu'il est primordial de trouver des fragments décidables de théories du premier ordre.

Les procédures de décision pour des fragments de théories du premier ordre sont typiquement utilisées pour éliminer les sous-buts représentés par exemple comme des séquents modulo une théorie axiomatisant des types de données couramment utilisés dans des programmes tels que des tableaux, des listes, des vecteurs de bits, etc. L'approche déclarative, nécessaire pour que le programmeur puisse exprimer les propriétés à vérifier, a conduit à l'élaboration d'outils (voir, par exemple, Flanagan et al., 2002; Jackson and Vaziri, 2000) basés sur (des extensions de) la logique du premier ordre. Ces outils prennent en entrée un programme avec des annotations écrites en (une extension de la) logique du premier ordre et produisent un ensemble de formules (d'un fragment) de la logique du premier ordre dont la satisfiabilité implique qu'un bogue est présent dans le code. Pour vérifier la satisfiabilité, une procédure capable de gérer des obligations de preuve doit être disponible.

Combinaison Le déchargement d'obligations de preuve engendrées lors la vérification des logiciels et l'élimination des sous-buts dans la vérification avec des assistants de preuve se réduisent au problème de déterminer l'insatisfiabilité d'une formule sans variable libre et sans quantificateur, ayant une structure booléenne complexe modulo une théorie d'arrière-plan. C'est la raison principale pour étudier le problème de satisfiabilité de contraintes et la décidabilité de *fragments* de théories du premier

ordre. En outre, étant donné que les problèmes découlant de la vérification des logiciels concernent des domaines hétérogènes axiomatisés par des mélanges de théories différentes, il est crucial de s'intéresser à la modularité et à la réutilisation d'algorithmes et d'implantations de procédures existantes. La combinaison et l'intégration de procédures de décision existantes sont des tâches non triviales principalement en raison de l'hétérogénéité des techniques utilisées par les procédures de décision composantes.

Les Solveurs de Satisfiabilité Modulo Théories Les solveurs pour la Satisfiabilité Modulo des Théories (Satisfiability Modulo Theories, SMT) sont des systèmes qui traitent du problème de la satisfiabilité de combinaisons booléennes de littéraux clos par rapport à des théories d'arrière-plan pour lesquelles il existe des procédures de décision spécialisées. Ces théories ont des signatures disjointes dans les implémentations existantes. Citons par exemple les théories des listes, des tableaux, des vecteurs de bits, et de l'Arithmetique lineaire. Parmi les solveurs SMT, on peut citer des outils tels que :

- Argo-lib (http://www.matf.bg.ac.yu/~janicic/argo/);
- Barcelogic Tools (http://www.lsi.upc.edu/~oliveras/bclt-main.html);
- CVC3 (http://www.cs.nyu.edu/acsys/cvc3/);
- haRVey (http://harvey.loria.fr/);
- Math-SAT (http://mathsat.itc.it/);
- Simplics (http://fm.csl.sri.com/simplics/);
- Yices (http://yices.csl.sri.com/);
- Z3 (http://research.microsoft.com/projects/z3/).

Pour de plus amples renseignements au sujet des systèmes implantés pour la combinaison et les initiatives dans ce domaine, nous renvoyons le lecteur intéressé à la page Web de la *Satisfiability Modulo Theory Library* (http://combination.cs.uiowa.edu/smtlib/).

Vérification de Modèle

Le terme vérification de modèle représente un ensemble de techniques pour l'analyse automatique des systèmes concurrents. Un vérificateur de modèle prend en entrée la description à analyser du système (habituellement à états finis) et un certain nombre de propriétés souvent exprimées par des formules de logique temporelle,

et vérifie si la propriété est satisfaite ou non ; si elle ne l'est pas, il est censé fournir un contre-exemple, c'est-à-dire une exécution du système qui viole la propriété. Un bref état de l'art de (quelques unes) des techniques principales de vérification de modèle des systèmes à états infinis est fourni dans ce qui suit. Mais, comme la littérature dans le domaine est très vaste, cet état de l'art ne doit pas être considéré comme exhaustif.

Vérification de Modèle De Systèmes à États Infinis De nombreux aspects complexes sont d'une importance cruciale dans les systèmes logiciels modernes, telles que la manipulation de données dans les domaines infinis (entiers, réels, et ainsi de suite), les structures à mémoire dynamique (création et suppression d'objets, manipulation de pointeur), la synchronisation entre les processus concurrents, la paramétrisation, la modélisation en temps réel. La vérification de modèle des systèmes à états infinis étudie des méthodes pour vérifier les modèles abstraits qui impliquent des aspects comme ceux mentionnés ci-dessus. Les modèles bien connus permettant de représenter des systèmes à états infinis sont, par exemple, les processus parallèles élémentaires, les processus hors contexte, les processus "pushdown", les machines à compteur, les réseaux de Petri. Plus récemment, un formalisme de réécriture des termes appelé *système de réécriture de processus* (Process Rewrite système) qui généralise tous ces modèles a été introduit dans Mayr (1998).

De nombreuses techniques pour la vérification de modèle de systèmes à états infinis sont basées sur l'abstraction (voir, par exemple, Abdulla et al., 1996; Bouajjani et al., 2004; Graf and Saïdi, 1997) et de nombreux efforts sont consacrés à des méthodes automatisées d'abstraction de données, suivant l'idée de combiner l'abstraction de prédicat avec le raffinement d'abstraction guidé par les contre-exemples (counterexample-guided abstraction refinement), également connue sous le nom de *CEGAR* (voir, par exemple, Clarke et al., 2000). Les techniques automatisées d'abstraction et de raffinement basées, par exemple, sur l'abstraction paresseuse (lazy abstraction) (Henzinger et al., 2002) ou sur le calcul d'interpolants (voir, par exemple, Henzinger et al., 2004; McMillan, 2005) sont actuellement mises en œuvre dans de nombreux outils, comme par exemple SLAM, développé chez Microsoft Research (voir Ball and Rajamani, 2001), et BLAST (voir Henzinger et al., 2003).

D'autres approches de vérification de modèle de systèmes à états infinis sont représentées par des techniques basées sur les automates. La vérification de modèle régulière (*Regular Model Checking*) (voir Bouajjani et al., 2000) a été développée

pour la vérification algorithmique de plusieurs classes de systèmes à états infinis dont les configurations peuvent être modélisées comme des mots (finis ou infinis) ou des arbres construits sur un alphabet fini. Cette approche a été adoptée pour prendre en compte les différentes classes de systèmes tels que les systèmes à compteur, les systèmes *pushdown*, les systèmes de canaux FIFO, et les réseaux paramétrés de processus. Comme les techniques d'exploration d'espace d'états ne sont plus applicables dans le contexte de la vérification de modèle pour des systèmes à états infinis, les techniques de calcul de l'effet de séquences de transition arbitrairement longues, telles que le calcul de quotient, l'accélération et l'élargissement sont utilisées (voir, par exemple, Bouajjani et al., 1997, 2000, 2004; Esparza and Schwoon, 2001).

Notre Contribution

La principale contribution de cette thèse est double : d'une part, en ce qui concerne les problèmes de satisfiabilité, nous montrons le lien étroit entre les résultats relatifs à la combinaison de procédures de décision pour le problème de satisfiabilité de contrainte dans les théories arbitraires non-disjointes et la "temporalisation" d'une théorie du premier ordre. D'autre part, nous présentons un cadre qui permet d'avoir une approche déclarative pour la vérification de modèle des systèmes à états infinis. De plus, nous présentons un résultat de décidabilité du fragment universel de (extension de) la théorie des tableaux avec dimension.

Temporalisation et Satisfiabilité Les logiques temporelles sont largement utilisées pour raisonner sur des programmes concurrents, car elles offrent des primitives pour exprimer les relations temporelles de façon concise. Par conséquent, pour aider les ingénieurs en logiciel dans l'écriture de spécifications abstraites, concises et capables d'exprimer l'évolution des systèmes réactifs, le problème consiste à "ajouter une dimension temporelle" (en un sens, semblable à celle étudiée dans Finger and Gabbay, 1992) au fragment decidable d'une théorie du premier ordre avec égalité. Ce problème est étudié dans la première partie de cette thèse.

L'indécidabilité des logiques modales quantifiées sur un flux discret a été découverte par D. Scott dans les années soixante. Des travaux récents ont isolé des fragments de LTL très intéressants qui se comportent mieux en terme de calcul (voir Gabbay et al., 2003 pour plus de détails). Toutefois, ces fragments sont souvent insuffisants pour la vérification ; à cet égard, une restriction plus prometteuse

est d'interdire l'interaction entre les quantificateurs et les opérateurs temporels (voir Manna and Pnueli, 1995). Nous avons adopté une approche similaire enrichissant la partie extensionnelle du langage afin de modéliser les structures de données infinies manipulées par les systèmes. Cela nous amène à considérer la satisfiabilité de formules LTL sans quantificateur construites à partir d'une signature du premier ordre Σ et des modèles de domaine constant consistant en une séquence $\{\mathcal{M}_i\}_i$ de modèles du premier ordre d'une Σ-théorie T. De plus, les symboles de Σ et les variables libres sont divisés en deux groupes. Les premiers sont interprétés de façon rigide alors que les seconds le sont de façon flexible dans les modèles $\mathcal{M}_i$. Cette approche a été déjà adoptée dans le papier fondateur de Plaisted (1986), où l'auteur a établi un résultat de décidabilité lorsque le fragment sans quantificateurs de T est décidable et les symboles flexibles sont considérés comme des symboles libres par la théorie T. En utilisant des techniques récentes et des résultats de la littérature sur la combinaison, nous avons été en mesure de nous attaquer au problème dans toute sa généralité et de montrer à la fois l'indécidabilité dans le cas général (voir Bonacina et al., 2006) et à la décidabilité sous des hypothèses de 'combinabilité' pour T (voir Ghilardi, 2004). Ces hypothèses reposent sur la décidabilité du fragment universel du premier ordre et la compatibilité par rapport à une sous-théorie localement finie dans la sous-signature rigide.

L'hypothèse de finitude locale est ensuite affaiblie en la noethérianité. La procédure du combinaison est plus complexe que dans le cas localement fini, puisque l'énumération exhaustive par "guessing" ne peut plus être utilisée pour abstraire l'échange des littéraux qui sont maintenant possibles en nombre infini entre les théories composantes et de ce fait les résultats de combinaison de Ghilardi (2004) ne s'appliquent pas. Le mécanisme d'échange est formalisée par les *énumérateurs de résidu*, qui sont des fonctions calculables retournant des clauses positives déduites de la théorie partagée. Ceci nous a permis de montrer la décidabilité du problème de satisfiabilité des formules LTL sans quantificateur modulo une théorie du premier ordre T, lorsque T est une extension T_r-compatible, où T_r est effectivement noethérienne. Le résultat de décidabilité est ensuite étendu à toute les logiques temporelles/modales dont le problème de satisfiabilité propositionnelle relativisé est décidable. Enfin, nous montrons que nos exigences de 'combinabilité' liées à la noethérianité sont satisfaites par toute les extensions d'une théorie stablement infinie avec un symbole de fonction unaire libre.

Une Approche Déclarative du Model Checking La deuxième contribution de cette thèse est un cadre qui assure une approche déclarative de la vérification de modèle des systèmes à états infinis ; un tel cadre est basé sur des techniques provenant du domaine de la combinaison. Nous enrichissons le cadre conçu pour la satisfiabilité "temporalisée" en y ajoutant la capacité d'encoder des systèmes de transition. Nous dérivons des résultats d'indécidabilité et de décidabilité pour la vérification de modèle de propriétés de sûreté en appliquant des méthodes de combinaison non-disjointe pour des théories en logique du premier ordre. L'indécidabilité de ce problème est montrée (sous des hypothèses légères) par une réduction bien connue en un problème d'accessibilité pour les machines de Minsky (voir Minsky, 1961). Sous les mêmes hypothèses de compatibilité et de finitude locale, le problème de vérification de modèle pour les propriétés de sûreté sans quantificateur est montré décidable. La preuve de ce résultat indique comment des procédures de décision pour la problème de satisfiabilité de contrainte dans des théories en logique du premier ordre peuvent être intégrées à des algorithmes de contrôle de la satisfiabilité de formules LTL propositionnelles. Ceci permet l'emploi de solveurs SMT efficaces pour la vérification de modèle des systèmes à états infinis. Le résultat de décidabilité pour les propriétés de sûreté est ensuite généralisé à toutes les propriétés LTL.

Tableaux avec Dimension Depuis son introduction en McCarthy (1962), la théorie des tableaux a joué un rôle très important en informatique. Malheureusement, comme beaucoup des travaux précédents (voir, par exemple, Bradley, 2007; Bradley et al., 2006; Jaffar, 1981; Mateti, 1981; Suzuki and Jefferson, 1980) l'ont déjà observé, la théorie des tableaux seule ou même étendue avec l'égalité extensionnelle entre tableaux (comme dans Armando et al., 2003; Stump et al., 2001) n'est pas suffisante pour de nombreuses applications en vérification.

Comme dernière contribution de cette thèse nous considérons la théorie des tableaux avec extensionalité dont les indices ont la structure algébrique de l'arithmétique de Presburger et que nous étendons avec d'autres (fonction ou prédicat) symboles additionnels exprimant des caractéristiques importantes des tableaux (par exemple, la dimension d'un tableau ou un tableau trié). Nous donnons une méthode pour intégrer deux procédures de décision pour le problème de satisfiabilité de contrainte, une pour la théorie des tableaux et une pour l'arithmétique de Presburger, avec des stratégies d'instanciation qui nous permettent de réduire le problème de la satisfiabilité de contrainte pour (des extensions de) la théorie des tableaux avec

dimension au problème décidé par les deux procédures existantes. Notre démarche pour prouver la correction d'une version non-déterministe de la procédure de décision pour le problème de satisfiabilité de contrainte pour la théorie des tableaux avec dimension est inspirée par des méthodes de combinaison pour le problèmes de satisfiabilité (voir Ghilardi, 2004).

Bien que les procédures non-déterministes sont utiles pour montrer la correction, elles ne sont pas adaptées pour l'implantation. Nous abordons les questions de l'implantation de deux façons. D'abord, pour certaines extensions de la théorie de base, il est possible de réduire sensiblement le non-déterminisme à l'aide de méthodes fondées sur la réécriture pour construire des procédures de décision (voir, par exemple, Armando et al., 2003, 2007). Cependant, comme les méthodes basées sur la réécriture sont sensibles à l'axiomatisation des théories, et comme elles ne sont pas applicables à toutes les extensions considérées dans ce travail, nous adaptons des idées développées par la communauté SMT pour concevoir des procédures de décision intéressantes en pratique pour toutes les extensions de la théorie de tableaux avec dimension. En particulier, nous exploitons l'idée de Bozzano et al. (2006) d'utiliser un solveur booléen pour implanter de manière efficace la phase de "guessing" requise par les procédures non déterministes. Ceci ouvre la voie à la réutilisation des optimisations déjà disponibles dans les solveurs SMT et c'est notre deuxième (et principale) façon de résoudre le non-déterminisme.

Introduction

Nowadays, the presence of hardware and software systems in our lives becomes pervasive. These systems are often used in applications where failure is unacceptable, such as digital controllers supervising critical functions of cars, airplanes, medical instruments, or even software platforms guaranteeing privacy and reliability of electronic commerce applications, just to mention some examples. Ensuring reliability has effects in reducing the cost of software systems: the National Institute of Standards and Technologies (NIST) has estimated that bugs in software cost the U.S. economy about 59.5 billion dollars and that 80% of the cost of developing software goes into identifying and correcting defects (see NIST, 2002).

Testing and simulation are the most widespread techniques to identify bugs. They both involve making experiments before deploying the system in the field. In the case of hardware circuits, these methods typically inject signals at certain points and observe the resulting signals, whereas, for software, simulation and testing usually involve providing certain inputs and observing the corresponding outputs. Both of these methods, however, lack in terms of checking *all* the possible interactions, hence certifying the absence of pitfalls is rarely possible. Since these methods both focus on probable execution paths (or behaviors), they are incapable of spotting bugs which are revealed by paths with a low probability of execution. Such methods alone, as a matter of fact, are clearly inadequate for ensuring the quality not just of critical systems, but also of the software we use every day. Paradoxically, the more successful a software is, the more it will be used, and the more probable bugs not detected by testing will be reported by users, determining a decrease of the customer satisfaction.

A very attractive and increasingly appealing alternative to simulation and testing is the approach of *formal verification*. Formal methods refer to the use of techniques from logic and discrete mathematics to the specification, design, construction, and analysis of computer systems and software. They seem to be the ideal complement to

testing since they can consider all the possible behaviors of software. The verification of a system is done by providing a formal proof on an abstract formal model of the system, the correspondence between the formal model and the nature of the system being otherwise known by construction. Despite their promise, the formal methods have difficulties in being widely accepted by industry, mainly because of the lack of "off-the-shelf" tools which assist the software developer to apply the formal methods in a convenient way, without being overwhelmed with too many details concerning the mathematical notations or models. Augmenting the degree of automation seems to be the key to make the formal methods economically feasible and to ultimately meet the industrial standards.

The Context

The aim of formal verification is to verify in a formal way whether a system satisfies certain properties; hence, the starting point in order to apply methods and techniques coming from this field is to give a formal representation of the system and of the properties to be checked. The term *reactive system* usually identifies a system that changes its actions, outputs and conditions/status in response to stimuli from within or outside it. A common framework for the representation of the reactive systems is provided by *transition systems*; furthermore, the *temporal logics* are convenient languages to formally express the properties to be checked. The temporal logics are often classified according to whether the time is assumed to have a linear or a branching structure; in the former case there are logics such as propositional Linear time Temporal Logic LTL, whereas in the latter Computational Tree Logics CTL and CTL*, μ-calculus and so on. In Manna and Pnueli (1995) and many other writings, the authors have extensively shown how a mixture of first-order logic and LTL is sufficient to precisely state the verification problems for the class of the reactive systems. The theories in first-order logic model the (possibly infinite) data structures used by a reactive system while LTL specifies its (dynamic) behavior. The combination of LTL and first-order logic allows one to specify infinite-state systems and the subtle ways in which their data flow influences the control flow.

We can distinguish two different approaches in formal verification; a first approach, sometimes called *deductive verification*, consists of using a formal version of a logical reasoning about the system to prove its correctness; it usually relies on tools (called *theorem provers*) such as Coq (Coq, 2006), Isabelle/HOL (Nipkow et al.,

2002), Nqthm (Boyer and Moore, 1997), PVS (Owre et al., 1992), and STeP (Manna et al., 1994). This process is in general only partially automated and is driven by the user's understanding of the system to validate, but relies on fully automated procedures (from now on called *decision procedures*) for some subproblems. Although theoretical limitations forbid a complete automatization, an advantage of deductive verification is that it can be used for reasoning about infinite-state systems.

On the other hand, the second approach, called *model checking* (see Merz, 2001 for an overview and Burkart et al., 2001; Clarke et al., 1999 for further details on this topic), is a technique for verifying finite-state concurrent systems by, roughly speaking, exploring exhaustively the mathematical model used to represent the system itself. One of the benefit of the model checking is that the verification can be done completely automatically. The model checking techniques are widely used for verifying the finite-state systems, but in the last decade researchers have made many efforts to extend these techniques also to the infinite-state systems.

The approaches of deductive verification and model checking are however not necessarily disjoint. For example, in Sipma et al. (1999), an integration between classic tableaux and automated deduction techniques is shown, whereas Pnueli et al. (2001) presents a method for the automatic verification of a class of parametrized systems by using both model checking and deductive techniques, and finally Saïdi and Shankar (1999) shows how to define within a single framework proof strategies combining deductive proof construction, model checking, and abstraction. More details about the interplay between deductive verification and model checking techniques can be found also in Bjørner (1998).

The Satisfiability Problem

In deductive verification, the process of proving correctness of a system is usually only partially automated; however, it relies on decision procedures for some subproblems. A standard technique of deductive verification is based on the deductive invariance rule that consists in reducing the validity of certain temporal formulae to the validity of first-order sentences modulo a background theory (see Manna and Pnueli, 1995). This suggests that finding decidable fragments of first-order theories is of paramount importance.

The decision procedures for fragments of first-order theories are typically used for eliminating subgoals represented, for instance, as sequents modulo a background first-order theory which usually axiomatize the standard datatypes commonly used

in programs such as arrays, lists, bit-vectors and so on. The declarative approach, required to enable the programmer to express the properties to be checked, led to the development of tools (see, e.g., Flanagan et al., 2002; Jackson and Vaziri, 2000) based on (extensions of) first-order logic. These tools take in input a program with some annotations written in (an extension of) first-order logic and produce a set of formulae of (a fragment of) first-order logic whose satisfiability implies that a bug is present in the code. In order to check for satisfiability, a procedure capable of handling the generated proof obligations must be available.

Combination Discharging proof obligations arising in software verification and eliminating subgoals in verification with proof assistants reduce to the problem of proving the unsatisfiability of a quantifier-free sentence with a complex Boolean structure modulo a background theory. This is the main reason to study the constraint satisfiability problem and the decidability of *fragments* of first-order theories. Moreover, since problems deriving from software verification involve heterogeneous domains which are axiomatized by different theories, modularity in combining and re-using algorithms and concrete implementation of already developed decision procedures becomes crucial. The combination and integration of existing decision procedures are non trivial tasks mainly because of the heterogeneity of the techniques used by the component decision procedures. If we consider the theories which are suitable for software verification, decision procedures are obtained in many different ways: sometimes (e.g., when dealing with the empty theory, the theories of lists or of arrays) Superposition Calculus decides constraint satisfiability (Armando et al., 2003), but in many other cases ad hoc procedures are needed. In this context the problem of combining decision procedures naturally arises.

Satisfiability Modulo Theories Solvers The Satisfiability Modulo Theories solvers (SMT solvers for short) are systems dealing with the problem of the satisfiability of Boolean combinations of ground literals with respect to background theories for which a specialized decision procedure exists. Such background theories have disjoint signatures in the existing implementations, and among them there are the theories of lists, arrays, bit-vectors, and Linear Arithmetic; among the SMT solvers we can find tools such as:

- Argo-lib (`http://www.matf.bg.ac.yu/~janicic/argo/`);
- Barcelogic Tools (`http://www.lsi.upc.edu/~oliveras/bclt-main.html`);

- CVC3 (http://www.cs.nyu.edu/acsys/cvc3/);
- haRVey (http://harvey.loria.fr);
- Math-SAT (http://mathsat.itc.it/);
- Simplics (http://fm.csl.sri.com/simplics/);
- Yices (http://yices.csl.sri.com/);
- Z3 (http://research.microsoft.com/projects/z3/).

The general idea is to integrate a Boolean solver (usually based on DPLL algorithm) with a constraint satisfiability procedure for a first-order theory T (see, e.g., Nieuwenhuis et al., 2006). The systems are based on a loop consisting of the following steps: (i) the input formula φ which has to be tested for satisfiability modulo T is "abstracted" into a propositional formula φ^p; (ii) the Boolean solver enumerates the propositional assignment satisfying φ^p that can be "re-instantiated" as a conjunction of literals; (iii) each conjunction of literals is checked for T-satisfiability. The advantage of this idea is that the satisfiability procedure is not invoked whenever the inconsistency can be detected at a propositional level.

Conflict sets are used to refine the above schema and to minimize the (generally unavoidable) exponential blow-up determined by the exponentially many calls to the decision procedure for the involved theory. The conflict sets allow to lead the DPLL procedure to prune all the satisfiable propositional assignments that are eventually unsatisfiable modulo the involved theory; this technique is very useful in practice, and makes these systems well-performing. Many techniques arose recently to overcome the difficulties of dealing with the conflict sets, in particular in a combination context (see, e.g., Bozzano et al., 2006).

For further information about the implemented systems for combination and initiatives on this area, we refer the interested reader to the web page of the *Satisfiability Modulo Theory Library* (http://combination.cs.uiowa.edu/smtlib/).

Model Checking

The term model checking represents a collection of techniques for the automatic analysis of concurrent systems. A model checker takes in input a description of the (usually finite-state) system to be analyzed and a certain number of properties, often expressed as formulae of temporal logic, and either confirms that the property holds or not; in the latter case it is expected to show a counterexample, i.e. a

run of the system that violates the property. A brief review of (some of) the main techniques for the model checking of finite and infinite-state systems is provided in the following; however, since the literature in the field is extremely vast, this review should not be regarded as exhaustive.

Model Checking of Finite-State Systems The main disadvantage of the model checking is the state explosion problem that can occur if the system being verified has many components that can make transitions in parallel. Many efforts have been done by researchers to overcome this issue and to avoid the enumeration of all the possible states, and different techniques are currently used for the (finite-state) model checking.

The close connection between temporal logics and *automata* is the basis for one of the decision procedures for the satisfiability and the model checking for propositional LTL. The theory of automata over infinite words and trees was initiated by Büchi (1962); Rabin (1969), whereas Vardi (1991); Vardi and Wolper (1986) first proposed the use of ω-automata for automated verification, showing how the model checking problem for LTL could be formulated in terms of language containment between ω-automata. Automata-theoretic characterizations of branching-time logics (see Bernholtz et al., 1994) are based on tree automata; in this context, alternating automata allow for a rather uniform presentation of decision procedures for both linear and branching-time logics.

The so-called *global* model checking techniques for branching-time logics such as CTL rely on the evaluation of the (greatest or least) fixed points (see, e.g., Clarke et al., 1986). One can associate to a given formula the set of states in which the formula is true; in this way, the Boolean connectives are replaced by set-theoretic operations, and the (greatest or least) fixed points (of operators obtained by combining Boolean operators and inverse images) take the place of temporal connectives; in the finite-state case, these fixed points can be effectively computed.

In the original implementation of the model checking algorithm, the transition relations were represented explicitly by adjacency lists. In systems with many concurrent parts, however, the number of states in the global state transition graph was too large to handle. In contrast with the *explicit* model checking techniques, the *symbolic* model checking technique arose to deal with the state explosion problem (see, e.g., Burch et al., 1992). The idea is to exploit the relationship between temporal operators and fixed points to build a Boolean formula φ^B out of the temporal

property φ such that the underlying sets of states associated to the two formulae coincide. In this way, using suitable data structures, such as BDDs, that efficiently handle the operations needed to compute φ^B, the model checking problem is reduced to a tautology test in the propositional logic. The symbolic model checking techniques allow to verify much larger systems then the explicit model checking ones (see Burch et al., 1992; McMillan, 1992b).

Although symbolic model checking has been traditionally associated with BDDs, other representations of Boolean functions have been considered. This is the case of the *bounded* model checking technique (Biere et al., 1999), that relies on the observation that state sequences of some fixed length k can be represented by using k copies of the variables used to represent a single state. The existence of a state sequence of a fixed length k that represents a run of the transition system satisfying the propositional LTL property φ is reduced to the satisfiability of a certain propositional formula, which can be efficiently decided by using a SAT solver.

Other approaches to the state explosion problems are represented by the *partial-order reduction* techniques (see, e.g., Katz and Peled, 1988; McMillan, 1992a; Overman, 1981). A common model for representing concurrent software is the interleaving model, in which all of the events in a single execution are arranged in a linear order. The partial-order reduction techniques make it possible to decrease the number of interleaving sequence relying on the consideration that often specifications cannot distinguish between interleaving sequences in which two independent events are executed in different orders. The arguments for correctness are often simplified by appealing to some form of symmetry in the system: if a transition system is invariant under permutation (i.e., permuting individual values does not affect the overall behavior), techniques of *symmetry reductions* can be applied to obtain an equivalent transition system that is much smaller than the original (see, e.g. Clarke et al., 1993; Ip and Dill, 1993).

Although techniques such as symbolic model checking, partial-order reduction, and symmetry reduction attempt to cope with the state explosion problem, the dimension of the state space can be easily greater that 10^{100} states even if few hundred Boolean variables are involved. Model checking must therefore be performed on rather abstract models. The idea behind the *abstraction* techniques (see, e.g. Bensalem et al., 1992; Clarke et al., 1994; Cousot and Cousot, 1977; Loiseaux et al., 1995; Long, 1993) is that it is rarely necessary to consider the system in full detail in order to check some given property. This idea can be formalized as an abstraction

relation that induces an abstract model in which failure of the property implies the failure of the property in the original model. The abstraction-based approaches are not entirely automatic methods; on the other hand, an interesting form of abstraction, called *predicate abstraction*, where the predicates of interest at the concrete level are mapped to Boolean variables at the abstract level, is the base for many approaches to software model checking.

Model Checking of Infinite-State Systems Many complex aspects are of crucial importance in modern software systems, such as manipulation of data over unbounded domains (integers, reals, and so on), dynamic memory structures (creation and deletion of objects, pointer manipulations), synchronization between concurrent processes, parametrization, real-time modeling. The infinite-state model checking studies methods for the verification of abstract models that involve features such as those mentioned above. Widely known models for representing infinite-state systems are, e.g., Basic Parallel Processes, context-free processes, pushdown processes, counter machines, Petri nets. More recently, a term rewriting formalism called *Process Rewrite System* that generalizes all these models has been introduced in Mayr (1998).

Many techniques for the model checking of infinite-state systems are based on the abstraction (see, e.g., Abdulla et al., 1996; Bouajjani et al., 2004; Graf and Saïdi, 1997) and many efforts are devoted to the development of automated data abstraction methods, following the idea to combine the predicate abstraction with *counterexample-guided abstraction refinement*, also known as *CEGAR* (see, e.g., Clarke et al., 2000). The automated abstraction-refinement techniques are currently implemented in many tools, including SLAM, developed at Microsoft Research (see Ball and Rajamani, 2001), and BLAST (see Henzinger et al., 2003). CEGAR consists of the following steps:

abstraction a finite set of predicates is chosen, and an abstract model is built automatically out of the "concrete" model as a finite or push-down automaton whose states represent truth assignments for the chosen predicates;

verification the abstract model is checked for the desired property. If the property holds on the abstract model (i.e., the abstract model is "error-free"), so it is the original model, thus the procedure stops; otherwise an abstract counterexample is produced;

refinement it is checked automatically if the abstract counterexample corresponds to a concrete counterexample in the original model. If so, then the property does not hold on the original model and the procedure stops; otherwise, the abstract model is too coarse, and the failure to concretize the abstract counterexample automatically guides the selection of new predicates to "refine" the abstract model; the procedure continues to step **verification**.

The refinement phase might lead to the generation of a completely new abstract model; the *lazy abstraction* techniques (Henzinger et al., 2002) are used to refine the abstract model "locally". In the verification phase, spurious counterexamples (i.e., abstract runs falsifying the property that are not feasible in the original model) can be generated because the set of predicates chosen for the abstraction does not contain enough information on the original model. The information obtained by such counterexamples is used to refine the abstract model in order to avoid such spurious runs. Recently, it has been shown that the techniques for computing interpolants can be used to guide the refinement phase (see, e.g., Henzinger et al., 2004; McMillan, 2005).

Other approaches to the infinite-state model checking are represented by the techniques based on automata. *Regular model checking* (see Bouajjani et al., 2000) is being developed for algorithmic verification of several classes of infinite-state systems whose configurations can be modeled as (finite or infinite) words or trees over a finite alphabet. This approach has been adopted for dealing with various classes of systems such as counter systems, pushdown systems, FIFO channels systems, and parametrized networks of process. Since the state-space exploration techniques are no more applicable in the context of the infinite-state model checking, techniques for computing the effect of arbitrarily long sequences of transition such as quotienting, acceleration, and widening arose (see, e.g., Bouajjani et al., 1997, 2000, 2004; Esparza and Schwoon, 2001).

Many other techniques are developed in the field of the model checking of infinite-state systems. To name but a few challenging tasks, in order to reason about *programs with pointers* and dynamic management of the memory, the approaches are mainly based on the use of fragments of separation logic (see, e.g., Brookes, 2004; O'Hearn, 2007), translation to counter automata (see, e.g., Bouajjani et al., 2005, 2006), and graph rewriting (see, e.g., Heckel, 1998). Several groups are developing approaches for the *termination analysis*, i.e. the automatic verification of program termination (see, e.g., Bradley et al., 2005; Cook et al., 2005; Tiwari, 2004). *Paramet-*

ric verification intends to verify systems comprising a network of arbitrary number of identical or similar components running concurrently; typical examples of such systems are the mutual exclusion, the cache coherence, and the broadcast protocols (see, e.g., Abdulla and Jonsson, 1998; Abdulla et al., 1999; Delzanno et al., 1999; Esparza et al., 1999; German and Sistla, 1992).

Our Contribution

The main contribution of this thesis is twofold: on the one hand, as far as satisfiability problems are concerned, we show the close connection between combination results for the constraint satisfiability problem for non-disjoint theories and the "temporalization" of a first-order theory. On the other hand, we present a framework that allows for a declarative approach to the model checking of infinite-state systems. Finally, a decidability result for the universal fragment of (extension of) the theory of arrays with dimension is presented.

Temporalization and Satisfiability Temporal logics are widely used for reasoning about concurrent programs because they offer primitives for expressing time relationships concisely. Hence, to the aim of helping software engineers in writing concise and abstract specifications capable of expressing the evolution of reactive systems, the problem of "adding a temporal dimension" (in a sense similar to that investigated in Finger and Gabbay, 1992) to a decidable fragment of a first-order theory with identity immediately originates. This problem is considered in the first part of this thesis.

The undecidability of quantified modal logics over a discrete flow was discovered by D. Scott already in the sixties. Recent works isolated quite interesting fragments of quantified LTL which are computationally better behaved (see Gabbay et al., 2003 for a survey). However such fragments are often insufficient for the verification purposes; in this respect, a more promising restriction is to prohibit the interplay between quantifiers and temporal operators (see Manna and Pnueli, 1995). We have taken a similar approach by enriching the extensional part of the language so to be able to model the infinite data structures manipulated by systems. This leads us to consider the satisfiability of quantifier-free LTL formulae built up from a first-order signature Σ and models with constant domain consisting of a sequence $\{\mathcal{M}_i\}_i$ of first-order models of a Σ-theory T. Furthermore, the symbols in Σ and the free

variables were divided into two groups. The former are interpreted rigidly whereas the latter flexibly in the $\mathcal{M}_i$'s. This approach was already taken in the seminal paper Plaisted (1986), where the author established a decidability result when the quantifier-free fragment of T is decidable and the flexible symbols are considered as free symbols by the theory T. By using recent techniques and results from the combination literature, we were able to attack the problem in its full generality and derive both the undecidability in the unrestricted case (see Bonacina et al., 2006) and the decidability under the 'combinability' hypotheses for T (see Ghilardi, 2004). These hypotheses, besides the decidability of the universal first-order fragment, were the compatibility over a locally finite subtheory in the rigid subsignature.

The local finiteness requirement is then weakened to Noetherianity. The combination procedure is more complex than in the locally finite case, since the exhaustive enumeration of guessings cannot be used anymore to abstract away the exchange of now (possibly) infinitely many literals between the component theories, hence the combination results in Ghilardi (2004) do not apply. The exchange mechanism is formalized by *residue enumerators*, i.e. computable functions returning entailed positive clauses in the shared theory. This leads us to show the decidability of the satisfiability problem for the quantifier-free LTL formulae modulo a first order theory T, when T is an effectively Noetherian and T_r-compatible extension of T_r. The decidability result is then extended to any modal/temporal logic whose propositional relativized satisfiability problem is decidable. Finally, we show that our 'combinability' requirements related to Noetherianity are met by any extension with a free unary function symbol of a stably infinite theory.

A Declarative Approach to Model Checking The second contribution of this thesis is a framework that allows for a declarative approach to the model checking of infinite-state systems; such a framework is based on techniques coming from the combination field. We enrich the framework built for "temporalized" satisfiability by adding on top of it the capability to encode the transition systems. We derive undecidability and decidability results for the model checking of safety properties by lifting combination methods for (non-disjoint) theories in first-order logic. The undecidability of the safety model checking problem follows (under mild hypotheses) by a well-known reduction to the reachability problem for Minsky machines (see Minsky, 1961). Under the same compatibility and local finiteness hypotheses, the model checking problem for quantifier-free safety properties is shown to be decidable. The

proof of this result suggests how the decision procedures for the constraint satisfiability problem for first-order theories and the algorithms for checking the satisfiability of propositional LTL formulae can be integrated. This facilitates the employment of efficient Satisfiability Modulo Theories solvers in the model checking of infinite-state systems. The decidability result for safety properties is finally generalized to the LTL properties.

Arrays with Dimension Since its introduction in McCarthy (1962), the theory of arrays has played a very important role in Computer Science. Unfortunately, as many previous works have already observed (see, e.g., Bradley, 2007; Bradley et al., 2006; Jaffar, 1981; Mateti, 1981; Suzuki and Jefferson, 1980), the theory of arrays alone or even extended with extensional equality between arrays (as in Armando et al., 2003; Stump et al., 2001) is not sufficient for many applications of verification. For example, the works in Jaffar (1981); Mateti (1981); Suzuki and Jefferson (1980) tried to extend the theory to reason about sorted arrays. More recently, the works in Bradley (2007); Bradley et al. (2006) have shown the decidability of the satisfiability problem for a restricted class of (possibly quantified) first-order formulae that allows one to express many important properties about the arrays.

As the last contribution of this thesis we consider the theory of arrays with extensionality whose indexes have the algebraic structure of Presburger Arithmetic, and extend it with additional (function or predicate) symbols expressing important features of the arrays (e.g., the dimension of an array or an array being sorted). We give a method to integrate two decision procedures for the constraint satisfiability problem, one for the theory of arrays and one for Presburger Arithmetic, with instantiation strategies that allow us to reduce the constraint satisfiability problem of (extensions of) the theory of arrays with dimension to the problem decided by the two available procedures. Our approach to show the correctness of a non-deterministic version of the decision procedure for the constraint satisfiability problem for the theory of arrays with dimension is inspired by model-theoretic methods for combinations of satisfiability problems (see Ghilardi, 2004).

While the non-deterministic procedures are useful for showing correctness, they are not suited for the implementation. We address the implementation issues in two ways. First, for certain extensions of the base theory, it is possible to significantly reduce the non-determinism by using rewriting-based methods to build decision procedures (see, e.g., Armando et al., 2003, 2007). Since the rewriting-based methods

are sensitive to the axiomatization of the theories and hence they are not applicable to all the extensions considered in this work, we adapt ideas developed in the Satisfiability Modulo Theories (SMT) community to design practical decision procedures for all the extensions of the theory of arrays with dimension. In particular, we exploit the insight in Bozzano et al. (2006) of using a Boolean solver to efficiently implement the guessing phase required by the non-deterministic procedures. This paves the way to re-use the optimizations for efficiency already available in SMT solvers and is the second (and main) way to solve the non-determinism.

Overview

Chapter 1, after introducing the formal preliminaries about first-order logic, disjoint combination, and model theory (Section 1.1), presents in Section 1.2 the key definitions that are used to develop the following two chapters. Section 1.3 reviews some results in the field of (non-disjoint) combination of decision procedures for the constraint satisfiability problem that are useful in order to have a better insight into the contents of this thesis. In this context, the new result in Subsection 1.3.3 shows that, under suitable compatibility requirements, the property of being an effectively Noetherian extension is modular. Subsection 1.3.4 is dedicated to the readers that are interested in having a deeper understanding of the mathematical issues related to the notions introduced in Section 1.2. Finally, Sections 1.4 and 1.5 present some examples of theories satisfying the compatibility requirements and an entirely new class of theories amenable for being used in the combination context.

Chapter 2 is devoted to the study of what happens if we "add a temporal dimension" (in the sense discussed above) to a decidable fragment of a first-order theory with identity. Section 2.1 presents the syntax and semantic of LTL($\Sigma^{\underline{a}}$)-sentences and introduces the key notion of data-flow theories. In Section 2.2, a reduction to the constraint satisfiability problem for unions of (signature disjoint) theories in a first-order framework proves the undecidability of the (ground) satisfiability problem for (totally flexible) data-flow theories even if the underlying first-order theory has decidable constraint satisfiability problem. Decidability is obtained by adding a compatibility requirement; the result is first shown relying on the local finiteness of the rigid (i.e., time-independent) signature (Section 2.3), and then also under weaker Noetherianity hypotheses (Section 2.4). Finally, Section 2.5 shows how these results can be extended to any modal/temporal logic whose propositional relativized satis-

fiability problem is decidable.

Chapter 3 presents the framework for the model checking of infinite-state systems. After introducing the notion of LTL-system specification used to encode the transition systems, the model checking problem is addressed (Section 3.1). Section 3.2 shows that, even if the compatibility requirement is fulfilled, the ground model checking problem for an LTL-system specification based on a totally rigid data-flow theory is undecidable; this result is obtained through a simple reduction to the (undecidable) reachability problem of Minsky machines. Under the local finiteness requirement of the rigid signature, it is first shown that the (ground) model checking problem for safety properties is decidable, and then it is proved the decidability of its generalization to the LTL properties (Section 3.3). Section 3.4 provides examples to which our algorithm can be successfully applied.

Finally, **Chapter 4** gives decidability results for the universal fragments of (extensions of) the theory of arrays with dimension. After a brief presentation of some motivations and the intuition behind arrays with dimensions (Section 4.1), the nondeterministic decision procedure is presented in Section 4.2 where the complexity of the problem is also analyzed. Section 4.3 shows the decidability of the constraint satisfiability problem for some interesting extensions of the theory of arrays with dimensions. Finally, Section 4.4 is devoted to address some of the problems arising in the implementation of the procedures presented, both by using the rewriting-approach to build satisfiability procedures and relying on Satisfiability Modulo Theories solvers.

Many results of this thesis have already been published. More in detail, the results stated in Chapter 2 regarding the locally finite case were published in Ghilardi et al. (2007b), whereas the ones in the same chapter involving Noetherianity (as well as the content of Section 1.5) can be found in Ghilardi et al. (2007c). Moreover, the decidability of the model checking problem for safety properties shown in Chapter 3 was published in Ghilardi et al. (2007b). Finally, the work about the theory of arrays with dimension presented in Chapter 4 was published in Ghilardi et al. (2006a, 2007a).

Chapter 1

Non-Disjoint Combination

Many areas of computer science (such as software and hardware verification, artificial intelligence, knowledge representation and even computational algebra) are interested in the study and in the development of combination and integration techniques for existing decision procedures: this is so because there is a need to reason in heterogeneous domains, so that modularity in combining and re-using algorithms and concrete implementations becomes crucial. The key ingredient in such cases is the Nelson-Oppen method (see Nelson and Oppen, 1979; Oppen, 1980; Tinelli and Harandi, 1996), which was originally designed in order to combine decision procedures for the universal fragment of first-order theories whose signature shares only the equality predicate.

Recently it has been shown how to apply the Nelson-Oppen method also in case the signature of the theories involved are non-disjoint (see Ghilardi, 2004; Ghilardi et al., 2006b; Nicolini, 2007). The aim of this chapter is manyfold. First of all, after fixing the standard background on first-order logic, disjoint combination, and model theory, we give all the fundamental definitions that will be used in the rest of the thesis. Secondly, we review the extensions to the non-disjoint case of the Nelson-Oppen method in order to have a better insight into the results of Chapters 2 and 3 (in this context, a new modularity property is also proved). Thirdly, some mathematical observations that help deeply understand the notion introduced in the first part of the chapter are presented. Finally, some examples of theories fulfilling the 'combinability' requirements are given together with an entirely new class of theories amenable for being used in the combination context.

1.1 Formal preliminaries

The following subsections present some formal preliminaries for first-order logic, a brief review of the combination schema for the constraint satisfiability problem for theories over disjoint signatures, and some basic facts in model theory that will be used in the rest of this thesis.

1.1.1 First-Order Logic

A *signature* Σ is a set of functions and predicate symbols (each endowed with the corresponding arity). We assume the binary equality predicate symbol '=' to be always present in any signature Σ (so, if $\Sigma = \emptyset$, then Σ does not contain other symbols than equality). To avoid confusion, we use the symbol $\equiv$ (instead of $=$) in the metalanguage to mean identity of syntactic expressions. The signature obtained from Σ by adding a set $\underline{a}$ of new constants (i.e., 0-ary function symbols) is denoted by $\Sigma^{\underline{a}}$. Σ-*terms*, Σ-*atoms*, Σ-*literals*, Σ-*clauses*, and (elementary) Σ-*formulae* are defined in the usual way (we will omit the prefix Σ when it is clear from the context). A *positive clause* is a disjunction of atoms. A *constraint* is a conjunctions of literals. Terms, literals, clauses and formulae are called *ground* whenever no variable appears in them. Formulae without free variables are *sentences*. A Σ-*theory* T is a set of sentences (called the axioms of T) in the signature Σ. A formula is *quantifier-free* (or open) iff it does not contain quantifiers. The universal (resp., existential) closure of a formula φ is the sentence obtained by adding to φ a prefix of universal (resp., existential) quantifiers binding all variables which happen to have a free occurrence in φ.

In the following, letters $\varphi, \psi, \ldots$ are used for formulae; the following notations will be used below: $\varphi(\underline{x})$ means that the set of free variables in φ is contained in the finite set $\underline{x}$ whereas $\varphi(\underline{a}/\underline{x})$ (or, simply, $\varphi(\underline{a})$ leaving the $\underline{x}$ implicit) means that $\varphi(\underline{a})$ is the formula obtained from $\varphi(\underline{x})$ by the componentwise replacement of the free variables in $\underline{x}$ with the constants in $\underline{a}$.

From the semantic side, we have the standard notion of a Σ-*structure* $\mathcal{M} = (M, \mathcal{I})$: this is nothing but a support set M endowed with an arity-matching interpretation $\mathcal{I}$ of the function and predicate symbols from Σ. We use $f^{\mathcal{M}}$ (resp. $P^{\mathcal{M}}$) to denote the interpretation of the function symbol f (resp. predicate symbol P) in the structure $\mathcal{M}$ (the equality predicate $=$ is always interpreted as the identity relation over M). *Truth* of a Σ-formula in $\mathcal{M}$ is defined in any one of the standard

ways (so that truth of a formula is equivalent to truth of its *universal* closure). We let $\bot$ denote an arbitrary formula which is true in no structure. A formula φ is *satisfiable* in $\mathcal{M}$ iff its *existential* closure is true in $\mathcal{M}$.

A Σ-structure $\mathcal{M}$ is a *model* of a Σ-theory T (in symbols $\mathcal{M} \models T$) iff all the sentences of T are true in $\mathcal{M}$. If φ is a formula, $T \models \varphi$ ('φ *is a logical consequence of* T') means that φ is true in all the models of T ($T \models \varphi$ is equivalent to $T \models \forall \underline{x}\, \varphi$, where $\forall \underline{x}\, \varphi$ is the universal closure of φ). A Σ-theory T is *complete* iff for every Σ-sentence φ, either φ or $\neg\varphi$ is a logical consequence of T; T is *consistent* iff it has a model, i.e., $T \not\models \bot$. A sentence φ is T-consistent iff $T \cup \{\varphi\}$ is consistent.

A theory is *universal* iff it has universal closures of open formulae as axioms. A Σ-theory T admits *quantifier elimination* iff for every formula $\varphi(\underline{x})$ there is a quantifier-free formula $\varphi'(\underline{x})$ such that $T \models \varphi(\underline{x}) \leftrightarrow \varphi'(\underline{x})$. There are many well-known theories eliminating quantifiers (see Chang and Keisler, 1990), e.g., Linear (Integer[1] or Rational) Arithmetics, Real Arithmetics, acyclic lists, and any theory axiomatizing enumerated datatypes.

The *constraint satisfiability problem* for the constraint theory T is the problem of deciding whether a Σ-constraint is satisfiable in a model of T (or, equivalently, T-satisfiable).[2] In the following, we use free constants instead of variables in constraint satisfiability problems, so that we (equivalently) redefine a constraint satisfiability problem for the theory T as the problem of *establishing the consistency of* $T \cup \Gamma$ *for a finite set* Γ *of ground* $\Sigma^{\underline{a}}$*-literals* (where $\underline{a}$ is a finite set of new constants). For the same reason, from now on, *by a '*Σ*-constraint' we mean a 'ground* $\Sigma^{\underline{a}}$*-constraint'*, where the finite set of free constants $\underline{a}$ should be clear from the context (if not explicitly mentioned).

If $\Sigma_0 \subseteq \Sigma$ is a subsignature of Σ and if $\mathcal{M}$ is a Σ-structure, the Σ_0*-reduct* of $\mathcal{M}$ is the Σ_0-structure $\mathcal{M}_{|\Sigma_0}$ obtained from $\mathcal{M}$ by forgetting the interpretation of function and predicate symbols from $\Sigma \setminus \Sigma_0$. A Σ*-embedding* (or, simply, an embedding) between two Σ-structures $\mathcal{M} = (M, \mathcal{I})$ and $\mathcal{N} = (N, \mathcal{J})$ is any mapping $\mu : M \longrightarrow N$ among the corresponding support sets satisfying the condition

$$\mathcal{M} \models \varphi \quad \text{iff} \quad \mathcal{N} \models \varphi \tag{1.1}$$

[1] For Integer Arithmetic, infinite predicates expressing equivalence modulo n must be included in the language in order for quantifiers to be eliminable.

[2] Notice that the complementary constraint unsatisfiability problem (i.e. the problem of deciding whether a finite set of Σ-literals is unsatisfiable in all the models of T) is easily reduced to the problem of deciding whether $T \models \varphi$ holds, for quantifier-free φ.

for all Σ^M-atoms φ (here $\mathcal{M}$ is regarded as a Σ^M-structure, by interpreting each additional constant $a \in M$ into itself and $\mathcal{N}$ is regarded as a Σ^M-structure by interpreting each additional constant $a \in M$ into $\mu(a)$). Notice the following facts: (a) as we have equality in the language, an embedding is an injective function; (b) an embedding $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ must be an algebraic homomorphism, that is for every n-ary function symbol f and for every $a_1, \ldots, a_n \in M$, we must have $f^{\mathcal{N}}(\mu(a_1), \ldots, \mu(a_n)) = \mu(f^{\mathcal{M}}(a_1, \ldots, a_n))$;[3] (c) for an n-ary predicate symbol P we must have $(a_1, \ldots, a_n) \in P^{\mathcal{M}}$ iff $(\mu(a_1), \ldots, \mu(a_n)) \in P^{\mathcal{N}}$. It is easily seen that an embedding $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ can be equivalently defined as a mapping $\mu : M \longrightarrow N$ satisfying (a)-(b)-(c) above.

If $M \subseteq N$ and if the embedding $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ is just the identity inclusion $M \subseteq N$, we say that $\mathcal{M}$ is a *substructure* of $\mathcal{N}$ or that $\mathcal{N}$ is an *extension* of $\mathcal{M}$. In case (1.1) holds for all first order formulae, the embedding μ is said to be an *elementary* embedding. Correspondingly, in case μ is also an inclusion, we say that $\mathcal{M}$ is an elementary substructure of $\mathcal{N}$ or that $\mathcal{N}$ is an elementary extension of $\mathcal{M}$.

1.1.2 Disjoint Combination

Suppose we are given two first-order theories T_1 and T_2 over the signatures Σ_1 and Σ_2 respectively (notice that it may happen that the signatures Σ_1 and Σ_2 are non-disjoint). If we are able to solve the constraint satisfiability problem in both T_1 and T_2, we wonder whether it is possible to solve the same problem in $T_1 \cup T_2$.

In order to be able to re-use any existing decision procedure, it is useful to adopt a so-called *black-box approach*. This means the following: we assume that a decision procedure DP_1 solves the constraint satisfiability problem for the theory T_1 and a decision procedure DP_2 solves the constraint satisfiability problem for the theory T_2. The provers DP_1 and DP_2 can exchange information only externally, according to a protocol to be specified: in any case, DP_1 and DP_2 cannot be internally modified.

One of the simplest methodologies for the combination of decision procedures following the black-box approach is represented by the *Nelson-Oppen procedure* (see Nelson and Oppen, 1979), which was originally designed only for the disjoint signatures case. The Nelson-Oppen procedure can be summarized essentially in two steps, namely the *purification* preprocessing and the *exchange loop*.

Purification. The preprocessing step consists in the transformation of the initial

[3] To see this, apply (1.1) to the Σ^M-atom $f(a_1, \ldots, a_n) = a$, where $a \in M$ is just $f^{\mathcal{M}}(a_1, \ldots, a_n)$.

finite set Γ of $(\Sigma_1 \cup \Sigma_2)^{\underline{c}}$-ground literals into a set

$$\Gamma_1 \cup \Gamma_2$$

where, for some $\underline{c}'$, Γ_1 is a set of $\Sigma_1^{\underline{c},\underline{c}'}$-ground literals and Γ_2 is a set of $\Sigma_2^{\underline{c},\underline{c}'}$-ground literals. This transformation preserves satisfiability; in standard implementations, purification is linear (equations $c = t$, for new constants c and alien subterms t, are successively added).

Exchange Loop. Whenever the decision procedure DP_i ($i \in \{1, 2\}$) finds a disjunction $A_1 \vee \cdots \vee A_n$ ($n \geq 1$) of ground $\Sigma_0^{\underline{c},\underline{c}'}$-atoms (here $\Sigma_0 = \Sigma_1 \cap \Sigma_2$) such that $\Gamma_i \cup \{\neg A_1, \ldots, \neg A_n\}$ is T_i-unsatisfiable but $\Gamma_j \cup \{\neg A_1, \ldots, \neg A_n\}$ is T_j-satisfiable ($i, j \in \{1, 2\}$, $i \neq j$), then chooses nondeterministically $k \in \{1, \ldots, n\}$ and updates the current constraints by $\Gamma_1 := \Gamma_1 \cup \{A_k\}$ and $\Gamma_2 := \Gamma_2 \cup \{A_k\}$.

The exchange loop is non-deterministic, thus obviously case splitting and backtracking mechanisms are required. Notice however that, if the theories T_i are Σ_0-convex, the exchange of atoms becomes deterministic. Following Tinelli (2003), a theory T on the signature Σ is said to be Σ_0-*convex* ($\Sigma_0 \subseteq \Sigma$) iff whenever $T \cup \Gamma \models A_1 \vee \cdots \vee A_n$ (for a finite set of $(\Sigma \cup A)$-literals Γ, for $n \geq 1$ and for ground $(\Sigma_0 \cup A)$-atoms $A_1, \ldots, A_n$), there is $k \in \{1, \ldots, n\}$ such that $T \cup \Gamma \models A_k$. The procedure returns *"unsatisfiable"* if, for all backtracks, Γ_1 (or Γ_2) eventually becomes unsatisfiable modulo T_1 (modulo T_2, respectively). Otherwise, if there is a backtrack such that the loop terminates without finding any inconsistency, it returns *"satisfiable"*.

The deterministic Nelson-Oppen procedure is guaranteed to be terminating and complete under the following assumptions:

(i) Σ_1 and Σ_2 are disjoint;

(ii) the theories T_1 and T_2 are Σ_0-convex;

(iii) they admit only non trivial models (i.e. only models having cardinality bigger than 1).

The latter is not a real limitation: indeed, it is easy to adjust the deterministic procedure in order to drop it. In the non deterministic case, we can eliminate assumption (iii) and weaken the convexity assumption (ii) to:

(ii') the theories T_1 and T_2 are stably infinite.

Here a theory T over the signature Σ is said to be *stably infinite* iff any quantifier-free Σ-formula φ which is satisfiable in a model of T is satisfiable in a model of T whose domain is infinite. It is interesting to notice that theories which are both convex and do not admit trivial models are also stably infinite (see Barrett et al., 2002).

1.1.3 Basic Facts in Model Theory

In this subsection standard background in model theory will be recalled; for further information, we refer the reader to classical textbooks such as Chang and Keisler (1990); Hodges (1993). Given a Σ-structure $\mathcal{M} = (M, \mathcal{I})$ and a subset $C \subseteq M$, the *substructure of $\mathcal{M}$ generated by C* is the substructure obtained from $\mathcal{M}$ by restricting $\mathcal{I}$ to the subset $\{t^{\mathcal{M}}(\underline{c}) \mid \underline{c} \subseteq C \text{ and } t(\underline{x}) \text{ is a } \Sigma\text{-term}\}$ (here $t^{\mathcal{M}}$ is the function interpreting the term t in $\mathcal{M}$). In case this substructure coincides with $\mathcal{M}$, we say that C is a set of *generators* for $\mathcal{M}$; moreover, if C is finite, we say that $\mathcal{M}$ is *finitely generated* (by the generators C).

If C is a set of generators for $\mathcal{M}$, the *diagram* $\Delta(\mathcal{M})$ of $\mathcal{M}$ (w.r.t. Σ, C) consists of all ground Σ^C-literals that hold in $\mathcal{M}$; analogously, the *elementary diagram* $\Delta^e(\mathcal{M})$ of $\mathcal{M}$ (w.r.t. Σ, C) consists of all ground Σ^C-sentences that hold in $\mathcal{M}$ (often C is not specified at all, in these cases it is assumed to coincide with the whole carrier set of $\mathcal{M}$).

Diagrams (in combination with the compactness of the logical consequence relation) will be repeatedly used in the proofs of the main results of this thesis. A typical standard use is the following: suppose that we want to embed $\mathcal{M}$ into a model of a theory T, then it is sufficient to check that $T \cup \Delta(\mathcal{M})$ is consistent. This argument is justified by Robinson's Diagram Lemma (see Chang and Keisler, 1990), which relates embeddings and diagrams as follows.

Lemma 1.1.1 (Robinson's Diagram Lemma)**.** *Let $\mathcal{M}$ be a Σ-structure generated by a set C, and let $\mathcal{N}$ be another Σ-structure; then $\mathcal{M}$ can be embedded (resp. elementarily embedded) into $\mathcal{N}$ iff $\mathcal{N}$ can be expanded to Σ^C-model of the diagram $\Delta(\mathcal{M})$ (resp. of the elementary diagram $\Delta^e(\mathcal{M})$) of $\mathcal{M}$ w.r.t. Σ, C.*

Since the technique used for proving Lemma 1.1.1 is simple, we sketch it. If we have an expansion of $\mathcal{N}$ to a Σ^C-structure (to be called $\mathcal{N}$ again for simplicity), then, since every element of the support of $\mathcal{M}$ is of the kind $t^{\mathcal{M}}(\underline{c})$ for some $\underline{c} \subseteq C$,

we can define the embedding μ by putting $\mu(t^{\mathcal{M}}(\underline{c})) := t^{\mathcal{N}}(\underline{c}^{\mathcal{N}})$: this is well-defined and it is an embedding precisely because $\mathcal{N} \models \Delta(\mathcal{M})$. Conversely, if we have the embedding μ, then we can get the desired expansion by taking $c^{\mathcal{N}} := \mu(c)$ for all $c \in C$.

Since a surjective embedding is just an isomorphism, the argument just sketched shows also the following fact:

Lemma 1.1.2. *If two Σ-structures $\mathcal{M}$, $\mathcal{N}$ are both generated by a set C and if one of them, say $\mathcal{N}$, satisfies the other's diagram (w.r.t. Σ, C), then the two structures are Σ^C-isomorphic.*

Ground formulae are invariant under embeddings in the following sense.

Lemma 1.1.3. *Let $\mathcal{M} = (M, \mathcal{I})$ be a Σ-structure that can be embedded into another Σ-structure $\mathcal{N}$. For all ground Σ^M-sentences φ, we have that*

$$\mathcal{M} \models \varphi \qquad \Leftrightarrow \qquad \mathcal{N} \models \varphi,$$

where $\mathcal{N}$ is extended to a Σ^M-structure by interpreting each $a \in M$ by its image under the embedding.

Next lemma states the well-known property (called submodel-completeness) of theories enjoying quantifier-elimination:

Lemma 1.1.4. *Suppose that $T^\star$ is a Σ_0-theory enjoying quantifier elimination and that $\Delta(\mathcal{R})$ is a diagram of a substructure $\mathcal{R} = (R, \mathcal{J})$ of a model $\mathcal{M}$ of $T^\star$; then the Σ^R-theory $T^\star \cup \Delta(\mathcal{R})$ is complete.*

Proof. By Robinson's Diagram Lemma 1.1.1, the models of $T^\star \cup \Delta(\mathcal{R})$ are the models of $T^\star$ endowed with a Σ_0-embedding from $\mathcal{R}$. One such model is $\mathcal{M}$; we show that any other model $\mathcal{M}'$ satisfies the same Σ^R-sentences as $\mathcal{M}$ (we assume without loss of generality the Σ_0-embedding from $\mathcal{R}$ into $\mathcal{M}'$ to be an inclusion). Pick an arbitrary Σ^R-sentence $\varphi(\underline{c})$ (where the $\underline{c}$ are parameters from the set of generators of $\mathcal{R}$ used in order to build $\Delta(\mathcal{R})$): this sentence is equivalent, modulo $T^\star$, to a ground Σ^R-sentence $\varphi^\star(\underline{c})$. Since truth of ground sentences is preserved by substructures (cf. Lemma 1.1.3), we have the following chain of equivalences

$$\mathcal{M}' \models \varphi(\underline{c}) \Leftrightarrow \mathcal{M}' \models \varphi^\star(\underline{c}) \Leftrightarrow \mathcal{R} \models \varphi^\star(\underline{c}) \Leftrightarrow \mathcal{M} \models \varphi^\star(\underline{c}) \Leftrightarrow \mathcal{M} \models \varphi(\underline{c}),$$

showing our claim. □

Next result is also part of basic classical model theory: a proof of it can be easily deduced from Craig's Interpolation Theorem (alternatively, a direct proof using a double chain argument is possible, see Chang and Keisler, 1990, pp. 141-142):

Theorem 1.1.5 (Robinson's Joint Consistency Theorem)**.** *Let H_1, H_2 be, respectively, consistent Θ_1, Θ_2-theories and let Θ_0 be the signature $\Theta_1 \cap \Theta_2$. Suppose that there is a complete Θ_0-theory H_0 such that $H_0 \subseteq H_1$ and $H_0 \subseteq H_2$; then $H_1 \cup H_2$ is a consistent $\Theta_1 \cup \Theta_2$-theory.*

1.2 Compatible Theories

We recall some notions used to develop results for the non-disjoint combination of theories (see, e.g., Baader and Ghilardi, 2006; Baader et al., 2006; Ghilardi, 2004; Ghilardi and Santocanale, 2003; Ghilardi et al., 2006b). We refer the reader to Ghilardi (2004) for more information and for the proofs of side claims we are making in this section (these side claims will never be used within this thesis, but might be useful for a better insight into the notions we are going to introduce).

Definition 1.2.1 (T_0-compatibility – Ghilardi, 2004)**.** Let T be a theory in the signature Σ and let T_0 be a universal theory in a subsignature $\Sigma_0 \subseteq \Sigma$. We say that T is T_0-*compatible* iff $T_0 \subseteq T$ and there is a Σ_0-theory T_0^* such that

(i) $T_0 \subseteq T_0^\star$;

(ii) $T_0^\star$ has quantifier elimination;

(iii) every model of T_0 can be embedded into a model of $T_0^\star$;

(iv) every model of T can be embedded into a model of $T \cup T_0^\star$.

The requirements (i) to (iii) make the theory $T_0^\star$ unique, provided it exists ($T_0^\star$ is nothing but the so-called *model completion* of T_0, see Chang and Keisler, 1990).[4]

In principle, we do not need to have a characterization of $T_0^\star$, the mere information of its existence is enough for our decision procedures to be sound and complete and to implement them. As for T_0 itself, it is usually sufficient to take as T_0 the set of universal Σ_0-sentences which are logical consequence of T (for instance, this

[4] The standard definition of model completion (adopted also in Ghilardi, 2004) is slightly different, but can be proved to be equivalent to the above one in the case of universal theories, see the Appendix B of Ghilardi (2003) for details.

will be always the case for the temporal logic decision problems analyzed in this thesis). No information will be needed on axiomatizations of T_0 to run our decision procedures too, we shall just need qualitative information on properties of T_0, such as local finiteness, Noetherianity, etc. (see below).

A lot of examples of theories fitting Definition 1.2.1 can be easily obtained as follows: suppose that T_0^* is a Σ_0-theory that eliminates quantifiers and take T be any theory *whatsoever* in a bigger signature such that $T \supseteq T_0^*$. Then T is T_0-compatible, if we take as T_0 the theory having as axioms all the universal Σ_0-sentences which are logical consequences of $T_0^\star$.

Of course, the key requirements in Definition 1.2.1 are requirements (iii) and (iv). Such requirements trivialize in the case considered in the last paragraph; to understand what they mean, notice that (by Robinson's Diagram Lemma 1.1.1 and by compactness) they are equivalent to the following statements:

(iii') *every Σ_0-constraint which is satisfiable in a model of T_0 is satisfiable also in a model of $T_0^\star$;* [5]

(iv') *every Σ-constraint which is satisfiable in a model of T is satisfiable also in a model of $T_0^\star \cup T$.*[6]

These requirements are nothing but a generalization of the stable infiniteness requirement of the Nelson-Oppen combination procedure (see Nelson and Oppen, 1979; Tinelli and Harandi, 1996): in fact, if T_0 is the empty theory in the empty signature, $T_0^\star$ is the theory axiomatizing an infinite domain, so that (iii') holds trivially and (iv') is precisely stable infiniteness.

Other examples of T_0-compatible theories are given in Ghilardi (2004): for instance, any extension (in a richer functional signature and by means of equational axioms) of the theory BA of Boolean algebras is BA-compatible.

1.2.1 Locally Finite Theories

T_0-compatibility is used in order to obtain the completeness of combination algorithms; for termination, local finiteness and Noetherianity are the relevant requirements.

[5] Equivalently, T_0 and $T_0^\star$ entail the same universal Σ_0-sentences.
[6] Equivalently, T and $T \cup T_0^\star$ entail the same universal Σ-sentences.

Definition 1.2.2 (Local Finiteness)**.** We say that Σ_0-theory T_0 is *locally finite* iff Σ_0 is finite and, for every finite set of free constants $\underline{a}$, there are finitely many ground $\Sigma_0^{\underline{a}}$-terms $t_1, \ldots, t_{k_{\underline{a}}}$ such that for every further ground $\Sigma_0^{\underline{a}}$-term u, we have that $T_0 \models u = t_i$ (for some $i \in \{1, \ldots, k_{\underline{a}}\}$). If such $t_1, \ldots, t_{k_{\underline{a}}}$ are effectively computable from $\underline{a}$ (and t_i is computable from u), then T_0 is said to be *effectively locally finite*.

If Σ_0 is finite and does not contain any function symbol, then any Σ_0-theory is effectively locally finite; among effectively locally finite theories we have Boolean algebras, Linear Integer Arithmetic modulo a fixed integer, Arrays, and theories axiomatizing enumerated datatypes.

The main way in which local finiteness is exploited lies in the computation of finite *representatives* sets of ground atoms, clauses and formulae[7] in finitely expanded signatures. This means the following (e.g. in the case of atoms): consider the signature $\Sigma_0^{\underline{a}}$, obtained from Σ_0 by expanding it with finitely many free constants $\underline{a}$. Thanks to effective local finiteness of T_0, it is possible to compute finitely many $\Sigma_0^{\underline{a}}$-atoms $\psi_1(\underline{a}), \ldots, \psi_m(\underline{a})$ such that for any further $\Sigma_0^{\underline{a}}$-atom $\psi(\underline{a})$ there is some i such that $T_0 \models \psi_i(\underline{a}) \leftrightarrow \psi(\underline{a})$. These atoms $\psi_1(\underline{a}), \ldots, \psi_m(\underline{a})$ are called representatives (modulo T_0-equivalence) because they can freely replace arbitrary $\Sigma_0^{\underline{a}}$-atoms in computational considerations.

1.2.2 Noetherian Theories

Local finiteness is a quite strong requirement: in many cases a much weaker requirement is sufficient. This requirement is called a 'Noetherianity' requirement, because it generalizes standard conditions from abstract algebra.

In abstract algebra, the adjective Noetherian is used to describe structures that satisfy an ascending chain condition on congruences (see, e.g., MacLane and Birkhoff, 1988): since congruences can have special representations, Noetherianity concerns, e.g., chains of ideals in the case of rings and chains of submodules in the case of modules. Although this is somewhat non-standard, we may take a more abstract view and say that a *variety* (i.e. an equational class of structures) is Noetherian iff finitely generated free algebras satisfy the ascending chain condition for congruences or, equivalently, iff finitely generated algebras are finitely presented. Now, congruences over finitely generated free algebras may be represented as sets of equations

[7] Recall that when we say that a formula is ground we mean that it does not contain variables, neither free nor bounded.

among terms. This allows us to equivalently re-state the Noetherianity of varieties as "there are no infinite ascending chains of sets of equations modulo logical consequence". This observation was the basis for the abstract notion of Noetherian Fragment introduced in Ghilardi et al. (2006b), here adapted for an arbitrary first-order theory.

Definition 1.2.3 (Noetherian Theory)**.** A Σ_0-theory T_0 is *Noetherian* if and only if for every *finite* set of free constants $\underline{a}$, every infinite ascending chain

$$\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_n \subseteq \cdots$$

of sets of ground $\Sigma_0^{\underline{a}}$-atoms is eventually constant modulo T_0, i.e. there is an n such that $T_0 \cup \Theta_n \models A$, for every natural number m and atom $A \in \Theta_m$.

Natural examples of Noetherian theories are the first-order axiomatization (in equational logic) of varieties such as K-algebras, K-vector spaces, and R-modules, where K is a field and R is a Noetherian ring (see MacLane and Birkhoff, 1988 for further details). *Abelian semigroups* are also Noetherian (see Chenadec, 1986, Theorem 3.11). Notice that, since any extension (in the same signature) of a Noetherian theory is also Noetherian, any theory extending the theory of a single Associative-Commutative symbol is Noetherian. This shows that the family of Noetherian theories is important for verification because theories axiomatizing *integer addition* or *multiset union* formalize crucial aspects of a system to be verified (e.g., multisets may be used to check that the result of some operations like sorting on a collection of objects yields a permutation of the initial collection).

Before being able to describe our new combination method, we need to introduce some preliminary notions. In the remaining of this section, *we fix two theories* $T_0 \subseteq T$ *in their respective signatures* $\Sigma_0 \subseteq \Sigma$.

Definition 1.2.4 (T_0-basis)**.** Given a finite set Θ of ground clauses (built out of symbols from Σ and possibly further free constants) and a finite set of free constants $\underline{a}$, a T_0*-basis* for Θ w.r.t. $\underline{a}$ is a set Δ of *positive* ground $\Sigma_0^{\underline{a}}$-clauses such that

(i) $T \cup \Theta \models C$, for all $C \in \Delta$ and

(ii) if $T \cup \Theta \models C$ then $T_0 \cup \Delta \models C$, for every positive ground $\Sigma_0^{\underline{a}}$-clause C.

Notice that only constants in $\underline{a}$ may occur in a T_0-basis for Θ w.r.t. $\underline{a}$, although Θ may contain constants not in $\underline{a}$.

Definition 1.2.5 (Residue Enumerator)**.** Given a finite set $\underline{a}$ of free constants, a *T-residue enumerator for T_0 w.r.t. $\underline{a}$* is a computable function $Res^{\underline{a}}_T(\Gamma)$ mapping a Σ-constraint Γ to a finite T_0-basis for Γ w.r.t. $\underline{a}$.

If Γ is T-unsatisfiable, then without loss of generality a residue enumerator can always return the singleton set containing the empty clause. The concept of (Noetherian) residue enumerator is inspired by the work on partial theory reasoning (see, e.g., Baumgartner et al., 1992) and generalizes the notion of deduction complete procedure of Kirchner et al. (2005). Given a residue enumerator for constraints (cf. Definition 1.2.5), it is always possible to build one for clauses (this will be useful for the combination method, see below).

Lemma 1.2.6. *Given a finite set $\underline{a}$ of free constants and a T-residue enumerator $Res^{\underline{a}}_T$ for T_0 w.r.t. $\underline{a}$, there exists a computable function $Res^{\underline{a}}_T(\Theta)$ mapping a finite set of ground clauses Θ to a finite T_0-basis of Θ w.r.t. $\underline{a}$.*

Proof. We proceed as follows. First of all, let us convert Θ into its disjunctive normal form $\bigvee_i \Gamma_i$. Let $\Delta_i := Res^{\underline{a}}_T(\Gamma_i)$; we claim that Δ, namely the conversion into conjunctive normal form of $\bigvee_i \Delta_i$, is a T_0-basis for Θ w.r.t. $\underline{a}$. Indeed, Definition 1.2.4(i) is verified since, for each i, $T \cup \Gamma_i \models \Delta_i$ (because Δ_i is a T_0-basis for Γ_i), so it follows $T \cup \bigvee_i \Gamma_i \models \bigvee_i \Delta_i$, hence $T \cup \Theta \models \Delta$ (recall that Δ is logically equivalent to $\bigvee_i \Delta_i$). Moreover, Definition 1.2.4(ii) is verified because $T \cup \Theta \models C$ iff $T \cup \bigvee_i \Gamma_i \models C$ if and only if, for each i, $T \cup \Gamma_i \models C$, hence, for each i, $T_0 \cup \Delta_i \models C$ (again because Δ_i is a T_0-basis for Γ_i), and finally $T_0 \cup \Delta \models C$. □

If T_0 is Noetherian, then it is possible to show that a finite T_0-basis for Γ w.r.t. $\underline{a}$ always exists, for every Σ-constraint Γ and for every set $\underline{a}$ of constants, by using the following

Lemma 1.2.7. *Every infinite ascending chain of sets of positive ground $\Sigma^{\underline{a}}_0$-clauses is eventually constant for logical consequence modulo a Noetherian Σ-theory T_0.*

Proof. By contradiction, suppose not; in this case it is immediate to see that there are infinitely many positive ground T_0-clauses $C_1, C_2, \ldots$ such that for all i the clause C_i is not a logical consequence of $T_0 \cup \{C_1, \ldots, C_{i-1}\}$.

Let us build a chain of trees $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \cdots$, whose nodes are labeled by ground $\Sigma^{\underline{a}}_0$-atoms as follows. $\mathcal{T}_0$ consists of the root only, which is labeled $\top$. Suppose $\mathcal{T}_{i-1}$ is already built and consider the clause $C_i \equiv B_1 \vee \cdots \vee B_m$. To build $\mathcal{T}_i$, do

the following for every leaf K of $\mathcal{T}_{i-1}$ (let the branch leading to K be labeled by $A_1, \ldots, A_k$): append new sons to K labeled $B_1, \ldots, B_m$, respectively, if C_i is such that $T_0 \cup \{A_1, \ldots, A_k\} \not\models C_i$ (if this is not the case, do nothing for the leaf K).

Consider now the union tree $\mathcal{T} = \bigcup \mathcal{T}_i$: since, whenever a node labeled A_{k+1} is added, A_{k+1} is not a logical consequence w.r.t. T_0 of the formulae labeling the predecessor nodes, by the Noetherianity of T_0 all branches are then finite and by König lemma the whole tree is itself finite. This means that for some index j, the examination of clauses C_i (for $i > j$) did not yield any modification of the already built tree. Now, C_{j+1} is not a logical consequence of $T_0 \cup \{C_1, \ldots, C_j\}$: this means that there is a $\Sigma_0^{\underline{a}}$-structure $\mathcal{M}$ which is a model of T_0 and in which all atoms of C_{j+1} are false and the $C_1, \ldots, C_j$ are all true. By induction on $i = 0, \ldots, j$, it is easily seen that there is a branch in $\mathcal{T}_i$ whose labeling atoms are true in $\mathcal{M}$: this contradicts the fact that the tree $\mathcal{T}_j$ has not been modified in step $j + 1$ □

Unfortunately, a basis for a Noetherian theory is not always computable; this motivates the following

Definition 1.2.8. A theory T is an *effectively Noetherian extension of* T_0 if and only if T_0 is Noetherian and there exists a T-residue enumerator for T_0 w.r.t. every finite set $\underline{a}$ of free constants.

For example, the theory of commutative K-algebras is an effectively Noetherian extension of the theory of K-vector spaces, where K is a field (see Ghilardi et al., 2006b; Nicolini, 2007 for details). Locally finite theories and Linear Real Arithmetic are further examples taken from the literature about automated theorem proving.

The class of locally finite theories is (strictly) contained in that of Noetherian theories: to see this, it is sufficient to notice that, once fixed a finite set of free constants $\underline{a}$, only finitely many representatives over $\underline{a}$ are equivalent (modulo the locally finite theory) to any arbitrary set of atoms over $\underline{a}$. From this, it is obvious that any infinite ascending chain of sets of such atoms must be eventually constant. Under the hypotheses that T_0 is effectively locally finite and its extension T has decidable constraint satisfiability problem, it is straightforward to build a T-residue enumerator for T_0.

The case of Linear Real Arithmetic can be treated as follows (see Nicolini, 2007 for further details). Let us consider the signature $\Sigma = \{0, +, -, \{f_r\}_{r \in \mathbb{R}}, \leq\}$ where 0 is a constant, $-$ and f_r are unary function symbols, $+$ is a binary function symbol, $\leq$ is a binary predicate symbol, and $\Sigma_0 = \Sigma \setminus \{\leq\}$. We consider the theory $T_{\mathbb{R}}^{\leq} =$

$Th_\Sigma(\mathbb{R})$, i.e. the set of all Σ-sentences true in $\mathbb{R}$, which is seen as an $\mathbb{R}$-vector space equipped with a linear ordering, where the f_r's represent the external product so that terms are all equivalent to homogeneous linear polynomials. Finally, let $T_\mathbb{R}$ be $Th_{\Sigma_0}(\mathbb{R})$, i.e. the set of all Σ_0-sentences true in $\mathbb{R}$, which is seen as an $\mathbb{R}$-vector space without the ordering (so $T_\mathbb{R}$ is the theory of the $\mathbb{R}$-vector spaces, not reduced to $\{0\}$). The Noetherianity of $T_\mathbb{R}$ follows from general algebraic properties (see, e.g., MacLane and Birkhoff, 1988). A $T^{\leq}_\mathbb{R}$-residue enumerator for $T_\mathbb{R}$ can be obtained as follows. Let $\Gamma = \{C_1, \ldots, C_m\}$ be a set of inequalities, i.e. Σ-atoms whose main predicate symbol is $\leq$. By Definition 1.2.4, the Σ_0-basis for Γ is the set of all the (disjunctions of) equalities implied by Γ. Actually, to compute a basis, it is sufficient to identify the set of *implicit* equalities in Γ, i.e. the equalities $C_i^=$ such that $T^{\leq}_\mathbb{R} \models \Gamma \rightarrow C_i^=$ (here $C_i^=$ is obtained from C_i by substituting $\leq$ with $=$). This is so because (i) $T^{\leq}_\mathbb{R}$ is Σ_0-*convex* (i.e. if $T^{\leq}_\mathbb{R} \models \Gamma \rightarrow (e_1 \vee \cdots \vee e_n)$, then there exists $i \in \{1, \ldots, n\}$ such that $T^{\leq}_\mathbb{R} \models \Gamma \rightarrow e_i$, for $n \geq 1$ and equalities $e_1, \ldots, e_n$) and (ii) given a system of inequalities Γ, if Δ is the collection of all the implicit equalities of Γ and e is an equality such that $T^{\leq}_\mathbb{R} \models \Gamma \rightarrow e$, then $T_\mathbb{R} \models \Delta \rightarrow e$ (see Lassez and McAloon, 1992 for full details, Nicolini, 2007 for the adaptation to our context). The interest of implicit equalities is that they can be easily identified by using the Fourier-Motzkin variable elimination method (see Lassez and Maher, 1992 for details on how to do this).

1.3 Combination Results for Non-Disjoint Theories

In first two subsections, we review the combination results of Ghilardi (2004), taking into consideration also further extensions from Ghilardi et al. (2006b): the main results presented here (Theorems 1.3.1 and 1.3.3) will not be used in the remaining part of this thesis. Nevertheless, they might be useful in order to understand the role played within combination problems by the notion introduced so far. Moreover, the reader should notice that the proofs of Theorems 1.3.1 and 1.3.3 are given here by introducing Lemmas 1.3.2 and 1.3.8 that slightly extend the corresponding lemmas in Ghilardi (2004); Ghilardi et al. (2006b). These two lemmas will be used in order to prove the results presented in Chapters 2 and 3.

In the third subsection, some properties regarding the notions presented are proved; more in detail, we first recall the proof of Ghilardi (2004, Theorem 5.2 and Proposition 4.4) stating a "ground interpolation" property for T_0-compatible

theories and the modularity of the T_0-compatibility notion. Furthermore, we present a new result showing the modularity of the property of being effectively Noetherian extension.

Finally, the last subsection collects some remarks of mathematical interest that can help the reader have a better insight in the notion of T_0-basis, local finiteness, and noetherianity.

1.3.1 The Locally Finite Case

Suppose we are given theories T_1, T_2 in signatures Σ_1, Σ_2 and suppose that constraint satisfiability problem is decidable for both T_1 and T_2; *what can we say about constraint satisfiability problem for the* $(\Sigma_1 \cup \Sigma_2)$*-theory* $T_1 \cup T_2$? In general, not so much: constraint satisfiability problem in $T_1 \cup T_2$ can be undecidable, even if the shared signature $\Sigma_1 \cap \Sigma_2$ is empty (see Bonacina et al., 2006). We look for sufficient conditions making this 'decidability transfer result' available. We first state the following basic combination result:

Theorem 1.3.1 (Ghilardi, 2004)**.** *Suppose that the theories* T_1, T_2 *(in signatures* Σ_1, Σ_2*) both have decidable constraint satisfiability problem; then the* $(\Sigma_1 \cup \Sigma_2)$*-theory* $T_1 \cup T_2$ *also has decidable constraint satisfiability problem in case* T_1, T_2 *are both* T_0*-compatible for some universal and effectively locally finite* $(\Sigma_1 \cap \Sigma_2)$*-theory* T_0 *contained in* T_1, T_2*.*

As pointed out in Section 1.2, to get concrete applications of Theorem 1.3.1 it is sufficient to take any theories T_1, T_2 extending a locally finite quantifier eliminating theory $T_0^\star$ in the shared signature $\Sigma_1 \cap \Sigma_2$ (the T_0 fitting the hypotheses of Theorem 1.3.1 is then the theory whose axioms are all the universal consequences of $T_0^\star$): examples of such a $T_0^\star$ include Boolean algebras, Linear Integer Arithmetic modulo n, the theory of dense total orders without endpoints, and any theory axiomatizing enumerated datatypes. Another family of applications (covering the fusion decidability transfer result for global consequence relation in modal logic, see Wolter, 1998) arises by taking as T_1, T_2 equational extensions of the theory BA of Boolean algebras (in this case, the hidden $T_0^\star$ is the theory of atomless Boolean algebras, see Ghilardi, 2004 for details). Finally, it should be clear that Theorem 1.3.1 extends Nelson-Oppen combination result for disjoint signatures (take T_0 to be the empty theory and $T_0^\star$ to be the theory of an infinite domain).

The algorithm LFCOMB suggested by the plain proof of Theorem 1.3.1 consists in the following three steps:

Step 1. The input $(\Sigma_1 \cup \Sigma_2)$-constraint Γ is *purified*, in the sense that, by repeatedly adding to it equations like $c = t$ (here t is a term occurring in Γ and c is a fresh constant), an equisatisfiable constraint $\Gamma_1 \cup \Gamma_2$ is produced, where Γ_i is a Σ_i-constraint for $i = 1, 2$;

Step 2. A maximal $\Sigma_0^{\underline{c}}$-constraint Δ is guessed (here Σ_0 is the shared signature $\Sigma_1 \cap \Sigma_2$, whereas the $\underline{c}$'s are the free constants occurring in both Γ_1 and Γ_2). A $\Sigma_0^{\underline{c}}$-constraint Δ is maximal iff for every $\Sigma_0^{\underline{c}}$-atom ψ, Δ contains a literal which is T_0-equivalent either to ψ or to $\neg\psi$ (notice that maximal constraints are computable, and finitely many modulo T_0, thanks to effective local finiteness of T_0).

Step 3. Return *"satisfiable"* iff $\Gamma_1 \cup \Delta$ is T_1-satisfiable and $\Gamma_2 \cup \Delta$ is T_2-satisfiable; return *"unsatisfiable"* iff all guessing Δ fail.

If there is a $(\Sigma_1 \cup \Sigma_2)^{\underline{c}}$-structure $\mathcal{M}$ which is a model for $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2$ then clearly the algorithm LFCOMB returns *"satisfiable"*. Conversely, if the algorithm returns *"satisfiable"*, then there exist $\Sigma_i^{\underline{c}}$-structures $\mathcal{N}_i$ such that $\mathcal{N}_i \models T_i \cup \Gamma_i$ and $\mathcal{N}_i$'s share the same $\Sigma_0^{\underline{c}}$-atoms ($i \in \{1, 2\}$). The existence of the $(\Sigma_1 \cup \Sigma_2)^{\underline{c}}$-structure model for $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2$ is stated by the following lemma, which slightly extends Lemma 9.4 in Ghilardi (2004), in the case $I = \{1, 2\}$ and $\underline{a}_1 = \underline{a}_2 = \emptyset$.

Lemma 1.3.2. *Let $\Sigma_i^{\underline{c},\underline{a}_i}$ (for $i \in I$) be signatures (expanded with free constants $\underline{c}, \underline{a}_i$), whose pairwise intersections are all equal to a certain signature $\Sigma_0^{\underline{c}}$ (that is, we have $\Sigma_i^{\underline{c},\underline{a}_i} \cap \Sigma_j^{\underline{c},\underline{a}_j} = \Sigma_0^{\underline{c}}$ for all distinct $i, j \in I$). Suppose we are also given Σ_i-theories T_i which are all T_0-compatible, where $T_0 \subseteq \bigcap_i T_i$ is a universal Σ_0-theory; let finally $\{\mathcal{N}_i = (N_i, \mathcal{I}_i)\}_{i \in I}$ be a sequence of $\Sigma_i^{\underline{c},\underline{a}_i}$-structures which are models of T_i and satisfy the same $\Sigma_0^{\underline{c}}$-atoms. Under these hypotheses, there exist a $\bigcup_i(\Sigma_i^{\underline{c},\underline{a}_i})$-structure $\mathcal{M} \models \bigcup_i T_i$ such that for each i, $\mathcal{N}_i$ has a $\Sigma_i^{\underline{c},\underline{a}_i}$-embedding into $\mathcal{M}$.*

Proof. By Robinson's Diagram Lemma 1.1.1 and Lemma 1.1.2 (and up to a partial renaming of the support sets), the fact that the $\mathcal{N}_i$ satisfy the same $\Sigma_0^{\underline{c}}$-atoms is another way of saying that they share the same $\Sigma_0^{\underline{c}}$-substructure generated by the $\underline{c}$ (let us call $\mathcal{R} = (R, \mathcal{J})$ this substructure); by T_0-compatibility, we may also freely assume that $\mathcal{N}_i \models T_i \cup T_0^\star$. Notice also that, by Lemma 1.1.4 above, the theory $T_0^\star \cup \Delta(\mathcal{R})$ is complete, where $\Delta(\mathcal{R})$ is the diagram of $\mathcal{R}$ as a Σ_0-structure.

Again by Robinson's Diagram Lemma 1.1.1, we only need to show that the union of the elementary diagrams $\Delta^e(\mathcal{N}_i)$ is consistent (here $\Delta^e(\mathcal{N}_i)$ is the elementary diagram of $\mathcal{N}_i$ as a $\Sigma_i^{\underline{c},\underline{a}_i}$-structure).[8]

By compactness, we can freely assume that the index set I is finite, let it be $\{1,\ldots,k\}$ and let us argue by induction on k. The case $k=1$ is trivial. For $k>1$, we use Robinson's Joint Consistency Theorem 1.1.5 as follows.

By renaming some elements in the supports if needed, we can freely suppose that the sets $N_1 \setminus R$ and $(N_2 \cup \cdots \cup N_k) \setminus R$ are disjoint. Given the hypotheses of the lemma on the signatures $\Sigma_i^{\underline{c},\underline{a}_i}$, we can apply the Joint Consistency Theorem 1.1.5 to the theories $\Delta^e(\mathcal{N}_1)$ and $\Delta^e(\mathcal{N}_2) \cup \cdots \cup \Delta^e(\mathcal{N}_k)$: in fact, they are both consistent (the latter by induction) and they both contain the complete subtheory $T_0^\star \cup \Delta(\mathcal{R})$ in the shared subsignature. This proves that $\Delta^e(\mathcal{N}_1) \cup \cdots \cup \Delta^e(\mathcal{N}_k)$ is consistent, as desired. □

1.3.2 The Noetherian Case

As remarked in Ghilardi et al. (2006b), a backtracking version of the combined decision algorithm can be sound and complete (although not terminating) even in case T_0 lacks the local finiteness requirement. In order to re-gain termination, a Noetherianity requirement can be used, witness the following result:

Theorem 1.3.3 (Ghilardi et al., 2006b)**.** *Suppose that the theories T_1, T_2 (in signatures Σ_1, Σ_2) both have decidable constraint satisfiability problem; then the $(\Sigma_1 \cup \Sigma_2)$-theory $T_1 \cup T_2$ also has decidable constraint satisfiability problem in case there is some universal and Noetherian $(\Sigma_1 \cap \Sigma_2)$-theory T_0 such that T_1, T_2 are both T_0-compatible effectively Noetherian extensions of T_0.*

Since the theory T_0 is not locally finite, we can no more use guessings. However, a slightly different proof of Theorem 1.3.1 suggests an alternative algorithm, based on propagation instead of guessing. Even better, instead of propagating entailed positive clauses (such as in Ghilardi, 2004), a splitting mechanism with backtracking can be used, as suggested in Ghilardi et al. (2006b). To this aim, the following procedure NComb is introduced:

Step 1. The input $(\Sigma_1 \cup \Sigma_2)$-constraint Γ is *purified*, in the sense that, by repeatedly adding to it equations like $c = t$ (here t is a term occurring in Γ and c is a

[8] We need the elementary diagrams here, and not just diagrams, because we want the model being defined to be a model of $\bigcup_i T_i$.

fresh constant), an equisatisfiable constraint $\Gamma_1 \cup \Gamma_2$ is produced, where Γ_i is a Σ_i-constraint for $i = 1, 2$;

Loop. If $Res^{\underline{c}}_{T_i}(\Gamma_i)$ contains the empty clause (here the $\underline{c}$'s are the free constants occurring in both Γ_1 and Γ_2), then update the current constraints by $\Gamma_1 := \Gamma_1 \cup \{\bot\}$ and $\Gamma_2 := \Gamma_2 \cup \{\bot\}$. Otherwise, pick a $\Sigma^{\underline{c}}_0$-clause C in $Res^{\underline{c}}_{T_i}(\Gamma_i)$ (let C be $A_1 \vee \cdots \vee A_n$, for $n \geq 1$) such that $\Gamma_j \cup \{\neg A_1, \ldots, \neg A_n\}$ is T_j-satisfiable (for $i, j \in \{1, 2\}$, $j \neq i$); choose nondeterministically $k \in \{1, \ldots, n\}$ and update the current constraints by $\Gamma_1 := \Gamma_1 \cup \{A_k\}$ and $\Gamma_2 := \Gamma_2 \cup \{A_k\}$.

Step 3. If Γ_1 and Γ_2 do not contain $\bot$ then return *"satisfiable"*; if all backtracks fail, return *"unsatisfiable"*.

Notice that backtracking is not needed if T_1 and T_2 are both Σ_0-convex theories, because in this case we can limit ourselves to positive unit clauses in the Loop.

The procedure NComb generate a tree labeled by sets of ground $\Sigma^{\underline{c}}_0$-atoms (for the sake of simplicity, in this context we *include the inconsistent proposition* $\bot$ *among atoms*). The root is labeled with the empty set and leaves are the unique nodes whose label set can contain $\bot$. The successors of an internal node labeled by Λ are of the following kind:

(i) a single leaf labeled by $\Lambda \cup \{\bot\}$ if $Res^{\underline{c}}_{T_i}(\Gamma_i \cup \Lambda)$ contains the empty clause for some $i \in \{1, 2\}$;

(ii) otherwise, nodes labeled by $\Lambda \cup \{A_1\}, \ldots, \Lambda \cup \{A_k\}$ if the $\Sigma^{\underline{c}}_0$-clause C chosen by the algorithm is $A_1 \vee \cdots \vee A_k$.

Proposition 1.3.4 (Termination)**.** *The procedure* NComb *always terminates.*

Proof. It is easy to see that the algorithm terminates if T_0 is a Noetherian theory. Indeed, suppose by contradiction that the algorithm does not stop. In this way the tree generated by the execution of the procedure (see above), which is a finitely branching tree by construction, is not finite and thus it has an infinite branch by König lemma. This means that there is an infinite chain of sets of ground $\Sigma^{\underline{c}}_0$-atoms $\Lambda_1 \subseteq \Lambda_2 \subseteq \cdots \subseteq \Lambda_n \subseteq \ldots$ where Λ_i is the label of a node that belongs to that infinite path, $\Lambda_{i+1} = \Lambda_i \cup \{A_i\}$ and $T_0 \cup \Lambda_i \not\models A_i$ (recall that T_0 is contained both in T_1 and T_2 and that the clause C containing A_i is such that $T_j \cup \Lambda_i \not\models C$ for some $j \in \{1, 2\}$). Contradiction, since T_0 is Noetherian. □

Proposition 1.3.5 (Soundness)**.** *If the procedure* NCOMB *returns "unsatisfiable", then the purified constraint* $\Gamma_1 \cup \Gamma_2$ *is* $(T_1 \cup T_2)$*-unsatisfiable.*

Proof. We consider the tree generated by the execution of the procedure. The thesis consists of proving that, if all the leaves contain $\bot$, then the purified constraint $\Gamma_1 \cup \Gamma_2$ is $(T_1 \cup T_2)$-unsatisfiable. The proof applies an inductive argument on the tree.

Consider a node labeled with Λ which is the parent of a leaf whose label contains $\bot$: by construction the empty clause belongs to $Res^c_{T_i}(\Gamma_i \cup \Lambda)$ for some $i \in \{1, 2\}$, thus $\Gamma_i \cup \Lambda$ is T_i-unsatisfiable by Definitions 1.2.4 and 1.2.5, and so $\Gamma_1 \cup \Gamma_2 \cup \Lambda$ is $(T_1 \cup T_2)$-unsatisfiable.

Consider now a tree whose leaves are labeled with sets containing $\bot$ and whose root is labeled by Λ. Suppose now, by inductive hypothesis, that each child of the root (labeled by $\Lambda \cup \{A_j\}$) is such that $\Gamma_1 \cup \Gamma_2 \cup \Lambda \cup \{A_j\}$ is $(T_1 \cup T_2)$-unsatisfiable ($j \in \{1, \ldots, k\}$). $\Gamma_1 \cup \Gamma_2 \cup \Lambda \cup \{A_j\}$ is $(T_1 \cup T_2)$-unsatisfiable for each j iff $\Gamma_1 \cup \Gamma_2 \cup \Lambda \cup \{A_1 \vee \cdots \vee A_k\}$ is $(T_1 \cup T_2)$-unsatisfiable: this means that our inductive hypothesis entails the $(T_1 \cup T_2)$-unsatisfiability of $\Gamma_1 \cup \Gamma_2 \cup \Lambda \cup \{A_1 \vee \cdots \vee A_k\}$. By construction, our internal nodes are labeled by $\Lambda \cup \{A_1\}, \ldots, \Lambda \cup \{A_k\}$ iff the Σ^c_0-clause chosen by the algorithm is $A_1 \vee \cdots \vee A_k$; hence, by construction and by Definitions 1.2.4 and 1.2.5, there exists an $i \in \{1, 2\}$ such that $T_i \cup \Gamma_i \cup \Lambda \models A_1 \vee \cdots \vee A_k$, thus being $A_1 \vee \cdots \vee A_k$ a logical consequence of $\Gamma_1 \cup \Gamma_2 \cup \Lambda$ w.r.t. $T_1 \cup T_2$. From the two considerations above it follows that $\Gamma_1 \cup \Gamma_2 \cup \Lambda$ is $(T_1 \cup T_2)$-unsatisfiable itself.

The thesis follows from the consideration that, when we run the procedure for the pure constraint $\Gamma_1 \cup \Gamma_2$, by construction the root of the tree is labeled by the empty set. □

To prove the completeness of the procedure NCOMB we need to introduce the following

Definition 1.3.6. Given a set of index I, a set $\mathcal{B}^\star$ of positive ground Σ^c_0-clauses is said to be saturated iff for every $i \in I$ and for every positive ground Σ^c_0-clause C it happens that:

$$T_i \cup \Gamma_i \cup \mathcal{B}^\star \models C \quad \Rightarrow \quad C \in \mathcal{B}^\star.$$

We are now in the position of proving the following

Lemma 1.3.7 (Completeness)**.** *If the procedure* NCOMB *returns "satisfiable", then the purified constraint* $\Gamma_1 \cup \Gamma_2$ *is* $(T_1 \cup T_2)$*-satisfiable.*

Proof. If NCOMB returns *"satisfiable"*, then the tree generated by the execution of the procedure (see above) contains a (finite, by Proposition 1.3.4) branch labeled by sets of $\Sigma_0^{\underline{c}}$-atoms $\Lambda_1 \subseteq \Lambda_2 \subseteq \cdots \subseteq \Lambda_n$ such that Λ_i does not contain $\bot$ $(1 \leq i \leq n)$. We define $\mathcal{B}^\star := \{C \mid C$ is a positive ground $\Sigma_0^{\underline{c}}$-clause such that $T_1 \cup \Gamma_1 \cup \Lambda_n \models C\}$.

We claim that $\mathcal{B}^\star$ is saturated; this can be proved by showing that, for each positive and ground $\Sigma_0^{\underline{c}}$-clause, $T_1 \cup \Gamma_1 \cup \Lambda_n \models C$ iff $T_2 \cup \Gamma_2 \cup \Lambda_n \models C$. If $T_1 \cup \Gamma_1 \cup \Lambda_n \models C$ then $T_2 \cup \Gamma_2 \cup \Lambda_n \models C$ because (i) by Definitions 1.2.4 and 1.2.5, $T_1 \cup \Gamma_1 \cup \Lambda_n \models C$ implies that $T_0 \cup Res_{T_1}^{\underline{c}}(\Gamma_1 \cup \Lambda_n) \models C$ and (ii) since Λ_n is the label of a leaf (thus, by construction, the loop of the procedure NCOMB terminates) $T_2 \cup \Gamma_2 \cup \Lambda_n \models Res_{T_1}^{\underline{c}}(\Gamma_1 \cup \Lambda_n)$. The converse holds for the same reasons.

Moreover, $\mathcal{B}^\star$ does not contain the empty clause because, since Λ_n does not contain $\bot$ and Λ_n is the label of a leaf, then $Res_{T_i}^{\underline{c}}(\Gamma_i \cup \Lambda_n)$ does not contain the empty clause, hence $\Gamma_i \cup \Lambda_n$ is T_i-satisfiable.

Since $\mathcal{B}^\star$ is saturated and does not contain the empty clause, then Lemma 1.3.9 applies with $I = \{1, 2\}$ and $\underline{a}_1 = \underline{a}_2 = \emptyset$, thus the $(T_1 \cup T_2)$-satisfiability of $\Gamma_1 \cup \Gamma_2$ obtains. □

The statement of next lemma extends the statement of Lemma 9.3 in Ghilardi (2004) and is proved in the same way.

Lemma 1.3.8. *Let T_i be Σ_i-theories (for $i \in I$) and let Σ_0 be a subsignature of all the Σ_i's. Let*

$$\Gamma_1, \ldots, \Gamma_i, \ldots \quad (i \in I)$$

be sets of ground $\Sigma_i^{\underline{a}_i, \underline{c}}$-clauses (here $\underline{a}_i, \underline{c}$ are free constants); suppose that $\mathcal{B}^\star$ is a saturated set of positive ground $\Sigma_0^{\underline{c}}$-clauses that does not contain the empty clause. Then there are $\Sigma_i^{\underline{a}_i, \underline{c}}$-structures $\mathcal{M}_i$ such that $\mathcal{M}_i \models T_i \cup \Gamma_i \cup \mathcal{B}^\star$; moreover, the $\Sigma_0^{\underline{c}}$-substructures generated by the elements (denoted by) $\underline{c}$ coincide for all the $\mathcal{M}_i$'s.

Proof. A set of ground $\Sigma_0^{\underline{c}}$-literals is said to be exhaustive iff it contains, for every ground $\Sigma_0^{\underline{c}}$-literal A, either A itself or its negation. The statement of the lemma is proved if we are able to find an exhaustive set Δ of ground $\Sigma_0^{\underline{c}}$-literals which is consistent with $T_i \cup \Gamma_i \cup \mathcal{B}^\star$ for each $i \in I$. In this case, in fact, given models $\mathcal{M}_i \models T_i \cup \Gamma_i \cup \mathcal{B}^\star \cup \Delta$, we have that the $\Sigma_0^{\underline{c}}$-substructures generated by $\underline{c}$ in all the $\mathcal{M}_i$'s all have diagram Δ, consequently they are $\Sigma_0^{\underline{c}}$-isomorphic (and can be made coincident by suitable renaming).

We shall adapt the notion of productive clause used in nowadays refutational completeness proofs for resolution or paramodulation based calculi. Consider any

strict total terminating order on ground Σ_0^c-atoms and extend it to a strict total terminating order $>$ for positive ground Σ_0^c-clauses by taking standard multiset extension. We shall define increasing sets Δ_C^+ (varying $C \in \mathcal{B}^\star$) of ground Σ_0^c-atoms as follows. Recall that, as the empty clause is not in $\mathcal{B}^\star$, all positive clauses in $\mathcal{B}^\star$ are of the kind $A \vee A_1 \vee \cdots \vee A_n$ $(n \geq 0)$.

The definition is by transfinite induction on $>$. Say that the clause $C \equiv A \vee A_1 \vee \cdots \vee A_n$ from $\mathcal{B}^\star$ is *productive* iff (i) $\{A\} > \{A_1, \ldots, A_n\}$ and (ii) $A_1, \ldots, A_n \notin \Delta_{<C}^+$ (where $\Delta_{<C}^+$ is $\bigcup_{D<C} \Delta_D^+$). Now, if C is productive, we let Δ_C^+ to be $\Delta_{<C}^+ \cup \{A\}$, otherwise Δ_C^+ is simply $\Delta_{<C}^+$.

Let Δ^+ be $\bigcup_{C \in \mathcal{B}^\star} \Delta_C^+$ and Δ be $\Delta^+ \cup \{\neg A \mid A$ is a ground Σ_0^c-atom not belonging to $\Delta^+\}$. By construction, $\Delta \models \mathcal{B}^\star$, so we simply need to show that $T_i \cup \Gamma_i \cup \Delta$ is consistent for each $i \in I$. We need a preliminary claim.

Claim. If the clause $A \vee A_1 \vee \cdots \vee A_n$ is productive and A is the maximum atom in it, then $A_1, \ldots, A_n \notin \Delta^+$; this is evident, as the A_i's could only be produced by clauses smaller than $A \vee A_1 \vee \cdots \vee A_n$.

Suppose now that $T_i \cup \Gamma_i \cup \Delta$ is not consistent. Then there are ground atoms $B_1, \ldots, B_m \notin \Delta^+$ and productive clauses

$$\begin{aligned} C_1 &\equiv A_1 \vee A_{11} \vee \cdots \vee A_{1k_1} \\ &\cdots \\ C_n &\equiv A_n \vee A_{n1} \vee \cdots \vee A_{nk_n} \end{aligned}$$

(with maximum atoms $A_1, \ldots, A_n$, respectively), such that

$$T_i \cup \Gamma_i \cup \{A_1, \ldots, A_n\} \models B_1 \vee \cdots \vee B_m.$$

By trivial logical manipulations, it follows that

$$T_i \cup \Gamma_i \cup \{C_1, \ldots, C_n\} \models \bigvee_{i,j} A_{ij} \vee B_1 \vee \cdots \vee B_m.$$

As $C_1, \ldots, C_n$ are clauses in $\mathcal{B}^\star$ and as $\mathcal{B}^\star$ is saturated, the clause

$$D \equiv \bigvee_{i,j} A_{ij} \vee B_1 \vee \cdots \vee B_m$$

is also in $\mathcal{B}^\star$. By construction (anyway, either D is productive or not) some of the

atoms $\{A_{11}, \ldots, A_{nk_n}, B_1, \ldots, B_m\}$ are in Δ^+. By the claim, $A_{11}, \ldots, A_{nk_n}$ cannot be there, so one of the B_j's is in Δ^+, contradiction. □

If we put together Lemmas 1.3.2 and 1.3.8, we get the following

Lemma 1.3.9. *Suppose we are given the following data:*

(i) I *is a (possibly infinite) set of indexes;*

(ii) $\Sigma_i^{\underline{c},\underline{a}_i}$ *(for* $i \in I$*) are signatures (expanded with free constants* $\underline{c}, \underline{a}_i$*), whose pairwise intersections are all equal to a certain signature* $\Sigma_0^{\underline{c}}$ *(that is, we have* $\Sigma_i^{\underline{c},\underline{a}_i} \cap \Sigma_j^{\underline{c},\underline{a}_j} = \Sigma_0^{\underline{c}}$ *for all distinct* $i, j \in I$*);*

(iii) T_i *are* Σ_i*-theories (for* $i \in I$*) which are all* T_0*-compatible, where* $T_0 \subseteq \bigcap_i T_i$ *is a universal* Σ_0*-theory;*

(iv) $\{\Gamma_i\}_{i \in I}$ *are sets of ground* $\Sigma_i^{\underline{c},\underline{a}_i}$*-clauses;*

(v) $\mathcal{B}^\star$ *is a saturated set of positive ground* $\Sigma_0^{\underline{c}}$*-clauses not containing the empty clause.*

If the above data are given, then there exists a $\bigcup_i(\Sigma_i^{\underline{c},\underline{a}_i})$*-structure* $\mathcal{M} \models \bigcup_i(T_i \cup \Gamma_i)$*. Equivalently: there exist* $\Sigma_i^{\underline{c},\underline{a}_i}$*-structures* $\mathcal{M}_i$ *(*$i \in I$*) satisfying* $T_i \cup \Gamma_i$*, whose* $\Sigma_0^{\underline{c}}$*-reducts coincide.*

Proof. By Lemmas 1.3.2 and 1.3.8. □

1.3.3 Interesting Properties

In this subsection three interesting properties will be investigated. We firstly recall two properties from Ghilardi (2004) used for the development of the results in Chapter 3, and then we present a new modularity result.

Ground Interpolation The following lemma (which is a variant of Ghilardi, 2004, Theorem 5.2) intuitively states that a property of ground interpolation holds for T_0-compatible theories.

Lemma 1.3.10. *Suppose that* T_0, T_1, T_2 *are* Σ_0*-,* Σ_1*-, and* Σ_2*-theories (respectively) such that* $\Sigma_0 = \Sigma_1 \cap \Sigma_2$*,* T_1 *is* T_0*-compatible, and* T_2 *is* T_0*-compatible; if the ground* $\Sigma_1^{\underline{a}_1,\underline{c}}$*-sentence* $\psi_1(\underline{a}_1, \underline{c})$ *and the ground* $\Sigma_2^{\underline{a}_2,\underline{c}}$*-sentence* $\psi_2(\underline{a}_2, \underline{c})$ *(here the tuples of free constants* $\underline{a}_1, \underline{a}_2, \underline{c}$ *are pairwise disjoint) are such that* $\psi_1(\underline{a}_1, \underline{c}) \wedge \psi_2(\underline{a}_2, \underline{c})$ *is*

$(T_1 \cup T_2)$-inconsistent, then there is a ground Σ_0^c-sentence $\psi_0(\underline{c})$ such that $T_1 \models \psi_1(\underline{a}_1, \underline{c}) \rightarrow \psi_0(\underline{c})$ and $T_2 \models \psi_0(\underline{c}) \rightarrow \neg\psi_2(\underline{a}_2, \underline{c})$.

Proof. By compactness, it is sufficient to show that the set Ψ of ground Σ_0^c-sentences $\psi_0(\underline{c})$ such that $T_1 \models \psi_1(\underline{a}_1, \underline{c}) \rightarrow \psi_0(\underline{c})$ is not T_2-consistent with $\psi_2(\underline{a}_2, \underline{c})$. Suppose it is, hence there is a T_2-model $\mathcal{M}_2$ of $\Psi \cup \{\psi_2(\underline{a}_2, \underline{c})\}$. Let $\mathcal{R}$ be the Σ_0-substructure of $\mathcal{M}_2$ generated by the $\underline{c}$'s and let $\Delta(\mathcal{R})$ be its diagram. We claim that $\Delta(\mathcal{R})$ is T_1-consistent with $\psi_1(\underline{a}_1, \underline{c})$: this is because, if $\psi_0(\underline{c})$ is a ground Σ_0^c-sentence true in $\mathcal{R}$ and not consistent with $\psi_1(\underline{a}_1, \underline{c})$, then $\neg\psi_0(\underline{c})$ would be in Ψ and hence would be true in $\mathcal{R}$, contradiction. Since $\Delta(\mathcal{R})$ is T_1-consistent with $\psi_1(\underline{a}_1, \underline{c})$, there is a model $\mathcal{M}_1$ of T_1 (having $\mathcal{R}$ as a substructure) in which $\psi_1(\underline{a}_1, \underline{c})$ is true. By Lemma 1.3.2 (take $I = \{1, 2\}$), the models $\mathcal{M}_1, \mathcal{M}_2$ embed, over $\mathcal{R}$, into a model $\mathcal{M}$ of $T_1 \cup T_2$; but then $\mathcal{M}$ is also a model of $\psi_1(\underline{a}_1, \underline{c}) \wedge \psi_2(\underline{a}_2, \underline{c})$ (because $\psi_1(\underline{a}_1, \underline{c})$ and $\psi_2(\underline{a}_2, \underline{c})$ are ground, cf. Lemma 1.1.3), a contradiction. □

Modularity of T_0-compatibility The following lemma proves that T_0-compatibility is a modular property. This result can be found in Ghilardi (2004, Proposition 4.4): we report the proof here.

Lemma 1.3.11. *If T_0, T_1, T_2 are Σ_0-, Σ_1-, and Σ_2-theories (respectively) such that $\Sigma_0 = \Sigma_1 \cap \Sigma_2$, T_1 is T_0-compatible, and T_2 is T_0-compatible, then $T_1 \cup T_2$ is T_0-compatible too.*

Proof. Take a model $\mathcal{M} = (M, \mathcal{I})$ of $T_1 \cup T_2$ and embeds its Σ_i-reducts into models $\mathcal{M}_i = (M_i, \mathcal{I}_i)$ of $T_i \cup T_0^\star$ $(i = 1, 2)$. We can freely suppose that the embeddings are inclusions and that we have $M = M_1 \cap M_2$ for supports. Now $T_0^\star \cup \Delta(\mathcal{M})$ is a complete theory by Lemma 1.1.4 (here $\Delta(\mathcal{M})$ is the diagram of $\mathcal{M}$ as a Σ_0-structure), hence by Robinson Joint Consistency Theorem 1.1.5 there is a model $\mathcal{N} = (N, \mathcal{J})$ of $\Delta^e(\mathcal{M}_1) \cup \Delta^e(\mathcal{M}_2)$. It follows that $\mathcal{N}$ is a $(\Sigma_1 \cup \Sigma_2)^{M_1 \cup M_2}$-model of $T_1 \cup T_2 \cup T_0^\star$ and that there are Σ_i^M-embeddings $\mu_i : \mathcal{M}_i \longrightarrow \mathcal{N}$. In particular, for $b \in M$, we have $\mu_1(b) = b^{\mathcal{N}} = \mu_2(b)$; let us call μ the common restriction of μ_1 and μ_2 to M. We show that μ is a $(\Sigma_1 \cup \Sigma_2)$-embedding of $\mathcal{M}$ into $\mathcal{N}$. Observe in fact that for every n-ary Σ_i-function symbol f and for every n-tuple $\underline{b}$ of elements from the support of $\mathcal{M}$, we have[9]

$$\mu(f^{\mathcal{M}}(\underline{b})) = \mu_i(f^{\mathcal{M}_i}(\underline{b})) = f^{\mathcal{N}}(\mu_i(\underline{b})) = f^{\mathcal{N}}(\mu(\underline{b}));$$

[9] Here, if $\underline{b} = (b_1, \ldots, b_n)$, we write $\mu(\underline{b})$ for the tuple $(\mu(b_1), \ldots, \mu(b_n))$.

analogously, for every n-ary Σ_i-predicate symbol P, we have

$$\mathcal{M} \models P(\underline{b}) \text{ iff } \mathcal{M}_i \models P(\underline{b}) \text{ iff } \mathcal{N} \models P(\mu_i(\underline{b})) \text{ iff } \mathcal{N} \models P(\mu(\underline{b})).$$

This proves that $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ is a $(\Sigma_1 \cup \Sigma_2)$-embedding. □

Modularity of Noetherian Extensions The fact that T_0-compatibility is a modular property is proved in Ghilardi (2004, Proposition 4.4) and it is also shown in this thesis (cf. Lemma 1.3.11), hence one can ask whether the property of being effectively Noetherian extension is modular as well. It turns out that the question has a positive answer in case the involved theories are T_0-compatible, as stated by the following

Theorem 1.3.12. *Let T_1 and T_2 be theories (in signatures Σ_1 and Σ_2 respectively) that are effectively Noetherian and T_0-compatible extensions of the same Σ_0-theory T_0 ($\Sigma_0 = \Sigma_1 \cap \Sigma_2$). Then the $(\Sigma_1 \cup \Sigma_2)$-theory $T_1 \cup T_2$ is an effectively Noetherian (and T_0-compatible) extension of T_0.*

The above theorem is proved by showing that Algorithm 1 gives a $(T_1 \cup T_2)$-residue enumerator for T_0. First of all, let us show that the Algorithm 1 terminates. This is stated by the following

Lemma 1.3.13. *Algorithm 1 always terminates.*

Proof. We have to prove that the test at line 10 eventually succeeds. To this aim we recall the fact (proved in Lemma 1.2.7) that every infinite ascending chain of sets of positive ground $\Sigma_0^{a,c}$-clauses is eventually constant for logical consequence w.r.t. a Noetherian theory T_0. The test at line 10 eventually have to succeed by the following reason: if we let $\mathcal{B}^0, \mathcal{B}^1, \mathcal{B}^2, \ldots$ be the values of the local variable $\mathcal{B}$ after each execution of the loop, we have that $T_0 \cup \mathcal{B}^{i+1} \models \mathcal{B}^i$, for each i, by Definition 1.2.4(ii). Thus, if we let $\mathcal{D}_i := \bigcup_{j \leq i} \mathcal{B}_j$, then the sequence

$$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \ldots$$

is increasing and hence eventually constant modulo T_0, which means that also the above mentioned test eventually succeeds. □

Consider the (finite, by the lemma above) sequence

$$\mathcal{B}^0, \mathcal{B}^1, \ldots, \mathcal{B}^h$$

Algorithm 1 $(T_1 \cup T_2)$-residue enumerator for T_0

Require: Γ $(\Sigma_1 \cup \Sigma_2)$-constraint

procedure $Res^{\underline{a}}_{T_1 \cup T_2}(\Gamma)$
  $\Gamma_1 \cup \Gamma_2 \leftarrow$ PURIFY(Γ) ▷ $\underline{c}$ constants occurring in both Γ_i's
  $\mathcal{B}' \leftarrow \emptyset$
  repeat
    $\mathcal{B} \leftarrow \mathcal{B}'$
    for all $i \in \{1, 2\}$ **do**
      $\mathcal{B}_i \leftarrow Res^{\underline{a},\underline{c}}_{T_i}(\Gamma_i \cup \mathcal{B})$
    end for
    $\mathcal{B}' \leftarrow \bigcup_i \mathcal{B}_i$
  until $\bigwedge_i$ DP-T$_i(\mathcal{B} \wedge \neg\mathcal{B}') =$ *"unsatisfiable"*
  return $Res^{\underline{a}}_{T_1}(\mathcal{B}')$
end procedure

of values of the local variable $\mathcal{B}$ after each execution of the loop. We need the following preliminary lemma

Lemma 1.3.14. *$\mathcal{B}^h$ is a T_0-basis for $\Gamma_1 \cup \Gamma_2$ w.r.t. $\underline{a} \cup \underline{c}$ (where $\underline{c}$'s are the free constants occurring in both Γ_1 and Γ_2).*

Proof. We have to prove that Definition 1.2.4(i) and (ii) hold on the set of ground positive $\Sigma_0^{\underline{a},\underline{c}}$-clause $\mathcal{B}^h$, i.e. that the following conditions hold:

(i) $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$, for all $C \in \mathcal{B}^h$ and

(ii) if $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$ then $T_0 \cup \mathcal{B}^h \models C$, for every positive ground $\Sigma_0^{\underline{a},\underline{c}}$-clause C.

Let us prove (i) by induction on $\mathcal{B}^i$. $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$ for all $C \in \mathcal{B}^0$ since $\mathcal{B}^0 = \emptyset$. Let us assume for induction hypothesis that $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$ for all $C \in \mathcal{B}^j$ $(0 \leq j < h)$. Since by construction for each $C \in \mathcal{B}^{j+1}$ there is an $i \in \{1, 2\}$ such that $T_i \cup \Gamma_i \cup \mathcal{B}^j \models C$, from the induction hypothesis it follows that $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$.

We now prove (ii). We assume that $\mathcal{B}^h$ does not contain the empty clause (otherwise (ii) clearly holds). By contradiction, suppose that there exists a positive ground $\Sigma_0^{\underline{a},\underline{c}}$-clause $\hat{C}$ such that $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models \hat{C}$ and $T_0 \cup \mathcal{B}^h \not\models \hat{C}$. We will show that there exists a $(\Sigma_1 \cup \Sigma_2)^{\underline{a},\underline{c}}$-structure $\mathcal{M}$ such that $\mathcal{M} \models T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2$ and such that $\mathcal{M} \models \neg\hat{C}$.

To this aim, consider the set $\mathcal{B}^\star := \{C \mid C$ is a positive ground $\Sigma_0^{\underline{a},\underline{c}}$-clause such

that $T_0 \cup \mathcal{B}^h \models C\}$. Let us prove that $\mathcal{B}^\star$ is saturated, i.e.

$$\begin{array}{lcll} T_1 \cup \Gamma_1 \cup \mathcal{B}^\star \models C & \Rightarrow & C \in \mathcal{B}^\star & \text{and} \\ T_2 \cup \Gamma_2 \cup \mathcal{B}^\star \models C & \Rightarrow & C \in \mathcal{B}^\star & \end{array}$$

for each positive ground $\Sigma_0^{\underline{a},\underline{c}}$-clause.

To see that $\mathcal{B}^\star$ is saturated, we proceed as follows. Consider a positive ground $\Sigma_0^{\underline{a},\underline{c}}$-clause C such that $T_1 \cup \Gamma_1 \cup \mathcal{B}^\star \models C$. By construction of $\mathcal{B}^\star$ this implies that $T_1 \cup \Gamma_1 \cup \mathcal{B}^h \models C$; since $T_1 \cup \mathcal{B}^{h-1} \models \mathcal{B}^h$ because the condition of line 10 holds, it follows that $T_1 \cup \Gamma_1 \cup \mathcal{B}^{h-1} \models C$, hence $T_0 \cup Res_{T_1}^{\underline{a},\underline{c}}(\Gamma_1 \cup \mathcal{B}^{h-1}) \models C$ (by Definitions 1.2.4 and 1.2.5) thus, a fortiori, $T_0 \cup Res_{T_1}^{\underline{a},\underline{c}}(\Gamma_1 \cup \mathcal{B}^{h-1}) \cup Res_{T_2}^{\underline{a},\underline{c}}(\Gamma_2 \cup \mathcal{B}^{h-1}) \models C$, so $T_0 \cup \mathcal{B}^h \models C$ (by definition of $\mathcal{B}^h$); this means that $C \in \mathcal{B}^\star$. The other condition follows analogously.

Now we can use the construction of Lemma 1.3.8 to obtain an exhaustive set of ground $\Sigma_0^{\underline{a},\underline{c}}$-literals Δ out of $\mathcal{B}^\star$ such that $\Delta \models \mathcal{B}^\star$, Δ is consistent both with $T_1 \cup \Gamma_1$ and with $T_2 \cup \Gamma_2$, and finally $\Delta \models \neg\hat{C}$. Let $\hat{C} \equiv A_1 \vee \cdots \vee A_k$ $(k \geq 1)$; to produce the required Δ out of $\mathcal{B}^\star$, it is sufficient to consider any strict total terminating order on ground $\Sigma_0^{\underline{a},\underline{c}}$-atoms such that $A_1 < A_2 < \cdots < A_k$ and every other $\Sigma_0^{\underline{a},\underline{c}}$-atom is greater than A_k (this order exists by the well-ordering principle and by the fact that any well-order can always be extended to a well-order by adding a new minimum element). It is easy to see that the construction of Lemma 1.3.8 produces an exhaustive set Δ out of $\mathcal{B}^\star$ such that $\Delta \models \mathcal{B}^\star$, Δ is consistent both with $T_1 \cup \Gamma_1$ and with $T_2 \cup \Gamma_2$, and finally $\Delta \models \neg\hat{C}$. The last property can be shown by observing that none of the A_i's belong to Δ^+. By contradiction, if $A_n \in \Delta^+$ for some $n \in \{1, \ldots, k\}$, it follows that A_n is the maximum atom in a productive clause C' belonging to $\mathcal{B}^\star$; our requirement on the order on the ground $\Sigma_0^{\underline{a},\underline{c}}$-atoms implies that the atoms of C' are among the A_i's, thus $C' \models \hat{C}$ hence $\mathcal{B}^\star \models \hat{C}$, contradicting the fact that $T_0 \cup \mathcal{B}^h \not\models \hat{C}$ (recall that $\mathcal{B}^\star := \{C \mid C$ is a positive ground $\Sigma_0^{\underline{a},\underline{c}}$-clause such that $T_0 \cup \mathcal{B}^h \models C\}$).

So we have obtained $\Sigma_i^{\underline{a},\underline{c}}$-structures $\mathcal{N}_i$ which are model of the T_0-compatible theories T_i and that satisfy the same $\Sigma_0^{\underline{a},\underline{c}}$-atoms ($i \in \{1,2\}$ and $\Sigma_0 = \Sigma_1 \cap \Sigma_2$). The contradiction will follow from Lemma 1.3.2 that states that there exists a $(\Sigma_1 \cup \Sigma_2)^{\underline{a},\underline{c}}$-structure $\mathcal{M}$ such that $\mathcal{M} \models T_1 \cup T_2$ and $\mathcal{N}_i$ has a $\Sigma_i^{\underline{a},\underline{c}}$-embedding into $\mathcal{M}$. Because of the $\Sigma_i^{\underline{a},\underline{c}}$-embeddings, from $\mathcal{N}_i \models \neg\hat{C}$ it follows that $\mathcal{M} \models \neg\hat{C}$, but since $\mathcal{M} \models T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2$, $\mathcal{M} \models \hat{C}$ (we recall that $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models \hat{C}$ by our hypothesis). □

We are now in the position of proving that Algorithm 1 gives a $(T_1 \cup T_2)$-residue enumerator for T_0. This is stated by the following

Lemma 1.3.15. *$Res^{\underline{a}}_{T_1}(\mathcal{B}^h)$ is a T_0-basis for Γ w.r.t. $\underline{a}$.*

Proof. We start showing that, for each purified $(\Sigma_1 \cup \Sigma_2)$-constraint $\Gamma_1 \cup \Gamma_2$, the following conditions hold on the set of ground positive $\Sigma^{\underline{a}}_0$-clause $Res^{\underline{a}}_{T_1}(\mathcal{B}^h)$:

(i) $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$, for all $C \in Res^{\underline{a}}_{T_1}(\mathcal{B}^h)$ and

(ii) if $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$ then $T_0 \cup Res^{\underline{a}}_{T_1}(\mathcal{B}^h) \models C$, for every positive ground $\Sigma^{\underline{a}}_0$-clause C.

Since, by Lemma 1.3.14, $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$ for each clause $C \in B^h$ and $T_1 \cup \mathcal{B}^h \models Res^{\underline{a}}_{T_1}(\mathcal{B}^h)$, it follows that $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$, for all $C \in Res^{\underline{a}}_{T_1}(\mathcal{B}^h)$. As far as (ii) is concerned, if $T_1 \cup T_2 \cup \Gamma_1 \cup \Gamma_2 \models C$ then $T_0 \cup \mathcal{B}^h \models C$ for every positive ground $\Sigma^{\underline{a}}_0$-clause (Lemma 1.3.14 proves that the above condition holds for all the positive ground $\Sigma^{\underline{a},\underline{c}}_0$-clauses), hence a fortiori $T_1 \cup \mathcal{B}^h \models C$, thus $T_0 \cup Res^{\underline{a}}_{T_1}(\mathcal{B}^h) \models C$ by Definitions 1.2.4 and 1.2.5.

To conclude the proof, it is sufficient to notice that, since Γ_1 and Γ_2 are produced out of Γ by purification, they are in the form $\Gamma_1(\underline{b})$ and $\Gamma_2(\underline{b})$ for some (finite) set of free constants $\underline{b}$ occurring neither in Γ nor in $\underline{a}$; indeed, $\Gamma_1(\underline{b})$ and $\Gamma_2(\underline{b})$ are such that $\exists \underline{x}\,(\Gamma_1(\underline{x}) \wedge \Gamma_2(\underline{x}))$ is logically equivalent to Γ. Hence, for every positive ground $\Sigma^{\underline{a}}_0$-clause C, $T_1 \cup T_2 \cup \Gamma_1(\underline{b}) \cup \Gamma_2(\underline{b}) \models C$ iff $T_1 \cup T_2 \cup \exists \underline{x}\,(\Gamma_1(\underline{x}) \wedge \Gamma_2(\underline{x})) \models C$ (because $\underline{b} \cap \underline{a} = \emptyset$) iff $T_1 \cup T_2 \cup \Gamma \models C$. □

1.3.4 Mathematical Remarks

The following subsection presents some results of mathematical interest that are useful to have a better insight in some notion presented so far. However, since the content of this subsection is not used to develop the results of this thesis, the reader who is not interested can freely skip it. In this subsection will be investigated the conditions under which a nice property of T_0-basis holds, and the semantic counterpart of the notion of local finiteness and Noetherianity.

BC Property A desirable property of T_0-basis is the following:

Property (BC)**.** *Let Θ be a set of ground $\Sigma^{\underline{c}}_0$-clauses and Δ is a T_0-basis for Θ w.r.t. $\underline{a}$. Δ is a T_0-basis for Θ also w.r.t. $\underline{a} \cup \underline{b}$ in case $\underline{b}$ is disjoint from $\underline{a}$ and $\underline{c}$.*

Unfortunately, the property BC (loosely related to the well-known Beck-Chevalley condition – see, e.g., Makkai and Reyes, 1977 for further details) is not directly implied by Definition 1.2.4, but it holds if we impose some further requirements (below, positive formulae mean formulae in which negation does not occur).

Fact. If T is T_0-compatible and quantifier elimination for $T_0^\star$ is such that every positive universal formula $\forall \underline{x}\, \varphi(\underline{a}, \underline{x})$ is equivalent modulo $T_0^\star$ to a positive quantifier-free formula $\psi(\underline{a})$, then Property BC holds.

Proof. Let Δ be a T_0-basis for Θ w.r.t. $\underline{a}$; in order to show that Δ is a T_0-basis for Θ w.r.t. $\underline{a} \cup \underline{b}$ we have simply to show that Definition 1.2.4(ii) holds (Definition 1.2.4(i) is obvious), hence we want to show that if $C(\underline{a}, \underline{b})$ is a positive ground $\Sigma_0^{\underline{a},\underline{b}}$-clause such that $T \cup \Theta \models C(\underline{a}, \underline{b})$, then $T_0 \cup \Delta \models C(\underline{a}, \underline{b})$. This can be seen by the following sequence of implications: $T \cup \Theta \models C(\underline{a}, \underline{b})$ implies that $T \cup \Theta \models \psi(\underline{a})$ (being $\psi(\underline{a})$ the positive quantifier-free formula such that $T_0^\star \models \forall \underline{x}\, C(\underline{a}, \underline{x}) \leftrightarrow \psi(\underline{a})$), hence $T \cup \Theta \models \bigwedge_i C_i(\underline{a})$ (being $\bigwedge_i C_i(\underline{a})$ the conjunctive normal form of $\psi(\underline{a})$) and so $T_0 \cup \Delta \models \bigwedge_i C_i(\underline{a})$ (recall that Δ is a T_0-basis for Θ w.r.t. $\underline{a}$), thus $T_0 \cup \Delta \models \psi(\underline{a})$, which finally implies that $T_0 \cup \Delta \models C(\underline{a}, \underline{b})$. The first implication holds because $T \cup \Theta \models C(\underline{a}, \underline{b})$ implies that $T \cup T_0^\star \cup \Theta \models C(\underline{a}, \underline{b})$, hence $T \cup T_0^\star \cup \Theta \models \forall \underline{x}\, C(\underline{a}, \underline{x})$ (since $\underline{b}$ is disjoint from both $\underline{a}$ and $\underline{c}$) and so $T \cup T_0^\star \cup \Theta \models \psi(\underline{a})$, which finally implies $T \cup \Theta \models \psi(\underline{a})$ by Definition 1.2.1(iv). A similar argument can be used for the last implication: $T_0 \cup \Delta \models \psi(\underline{a})$ implies that $T_0^\star \cup \Delta \models \psi(\underline{a})$, hence $T_0^\star \models C(\underline{a}, \underline{b})$, which finally implies $T_0 \cup \Delta \models C(\underline{a}, \underline{b})$ by Definition 1.2.1(iii). □

We remark that all the examples considered in this thesis of effectively Noetherian extensions of a theory T_0, when T_0 is not locally finite, admit T_0-bases satisfying Property BC.

Locally Finite Theories In the following, for the sake of readability, we often do not distinguish between elements of a structure and their names in an expanded signature. Now we want to give a semantic characterization of the universal theories that are locally finite. In particular, the following holds (cf. Subsection 1.1.3 for the definition of finitely generated structure):

Theorem 1.3.16. *Let T_0 be a universal first-order theory. Then T_0 is locally finite if and only if every finitely generated model of T_0 is finite.*

In order to prove the theorem above, we need to introduce the following

Definition 1.3.17. A Σ_0-theory T_0 is almost locally finite iff Σ_0 is finite and, for every finite set of free constants $\underline{a}$, there is a finite set of ground $\Sigma_0^{\underline{a}}$-terms $\mathcal{T}_{\underline{a}}$ such that for every further ground $\Sigma_0^{\underline{a}}$-term t, we have that $T \models t = t_1 \vee \cdots \vee t = t_n$ where $t_i \in \mathcal{T}_{\underline{a}}$.

Proposition 1.3.18. *Let T_0 be a universal Σ_0-theory. Then T_0 is almost locally finite iff every finitely generated model of T_0 is finite.*

Proof. The 'only if' case is obvious. For the converse, let $\mathcal{M} := \langle M, \mathcal{I} \rangle$ be a finite $\Sigma_0^{\underline{a}}$-structure generated by the parameters $\underline{a}$ (for the sake of simplicity, we assume that Σ_0 contains only function symbols); since $\mathcal{M}$ is finite and generated by $\underline{a}$, then $M := \{t_1(\underline{a}), \ldots, t_n(\underline{a})\}$. We define the *multiplication table* $\delta_{\mathcal{M}}$ of $\mathcal{M}$ as follows:

$$\delta_{\mathcal{M}} := \bigwedge_{t_i \in M, f \in \Sigma} \big\{ f(t_1, \ldots, t_k) = t_{k+1} \mid \mathcal{M} \models f(t_1, \ldots, t_k) = t_{k+1} \big\}.$$

Notice that, since $\mathcal{M}$ is a finite structure generated by $\underline{a}$, its multiplication table is a (finite) conjunction of $\Sigma_0^{\underline{a}}$-literals;[10] moreover, every model of $\delta_{\mathcal{M}}$ generated by $\underline{a}$ is a quotient of $\mathcal{M}$.

Let us now prove that there exist finitely many models of T_0 generated by $\underline{a}$. By contradiction, suppose that

$$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n, \ldots$$

are all the infinitely many models of T_0 generated by $\underline{a}$. Since T_0 is a universal theory (hence is preserved by substructures), $T_0 \cup \{\neg\delta_{\mathcal{M}_1}, \neg\delta_{\mathcal{M}_2}, \ldots\}$ is inconsistent, hence by compactness

$$T_0 \models \delta_{\mathcal{M}_{i_1}} \vee \cdots \vee \delta_{\mathcal{M}_{i_m}},$$

thus the models of T_0 generated by $\underline{a}$ are finitely many (at most as many as the quotients of $\mathcal{M}_{i_1}, \ldots, \mathcal{M}_{i_m}$).

Let $\mathcal{M}_1 := \langle M_1, \mathcal{I}_1 \rangle, \ldots, \mathcal{M}_s := \langle M_s, \mathcal{I}_s \rangle$ be the set of all the models of T_0 generated by $\underline{a}$ up to isomorphism. Let us build a finite set $\mathcal{T}_{\underline{a}}$ of $\Sigma_0^{\underline{a}}$-terms such that, for each $i \in \{1, \ldots, s\}$, $M_i \subseteq \{t^{\mathcal{M}_i} \mid t \in \mathcal{T}_{\underline{a}}\}$. It is easy to see that such a $\mathcal{T}_{\underline{a}}$ exists; moreover, for every further ground $\Sigma_0^{\underline{a}}$-term t, we have that $T_0 \models t = t_1 \vee \cdots \vee t = t_n$ where $t_i \in \mathcal{T}_{\underline{a}}$. Indeed, suppose by contradiction that there exists a $\Sigma_0^{\underline{a}}$-term t and a

[10] For example, if (i) $\Sigma_0 := \{f, g\}$ (where f and g are unary function symbols), (ii) $\underline{a} := \{a\}$, (iii) $M = \{f(a), a\}$, and (iv) $f^{\mathcal{M}}(f(a)) = a, f^{\mathcal{M}}(a) = f(a), g^{\mathcal{M}}(f(a)) = g^{\mathcal{M}}(a) = a$, then $\delta_{\mathcal{M}}$ is $f(f(a)) = a \wedge f(a) = f(a) \wedge g(f(a)) = a \wedge g(a) = a$.

model $\mathcal{N}$ of T_0 such that $\mathcal{N} \models t \neq u$ for every $u \in \mathcal{T}_{\underline{a}}$; since T_0 is a universal theory, the substructure of $\mathcal{N}$ generated by $\underline{a}$ is among $\mathcal{M}_1, \ldots, \mathcal{M}_s$, thus, by construction of $\mathcal{T}_{\underline{a}}$, $\mathcal{N} \models t = u$ for some $u \in \mathcal{T}_{\underline{a}}$. □

Proposition 1.3.19. *Let T be a universal Σ_0-theory. Then T_0 is almost locally finite iff T_0 is locally finite.*

Proof. The 'if' case is obvious. For the converse, let $\mathcal{T}_{\underline{a}}$ be the set of ground $\Sigma_0^{\underline{a}}$-terms such that for every further ground $\Sigma_0^{\underline{a}}$-term t, we have that $T_0 \models t = t_1 \vee \cdots \vee t = t_n$ where $t_i \in \mathcal{T}_{\underline{a}}$ (see Definition 1.3.17). We know from the proof of Proposition 1.3.18 that there exist finitely many models of T_0 generated by $\underline{a}$, namely $\mathcal{M}_1, \ldots, \mathcal{M}_k$. For each k-tuple $\underline{u} := \langle u_1, \ldots, u_k \rangle$ of $\Sigma_0^{\underline{a}}$-terms $u_i \in \mathcal{T}_{\underline{a}}$, we define the set of ground $\Sigma_0^{\underline{a}}$-terms $\mathcal{T}(\underline{u}) := \{t \mid \mathcal{M}_1 \models u_1 = t, \ldots, \mathcal{M}_k \models u_k = t\}$. Let $\mathcal{T}'_{\underline{a}}$ be the set of ground $\Sigma_0^{\underline{a}}$-terms obtained by choosing one element (if exists) in every $\mathcal{T}(\underline{u})$ varying $\underline{u}$ in $\underbrace{\mathcal{T}_{\underline{a}} \times \cdots \times \mathcal{T}_{\underline{a}}}_{k \text{ times}}$ (it is easy to see that $\mathcal{T}'_{\underline{a}}$ contains at most $|\mathcal{T}_{\underline{a}}|^k$ elements).

It follows that, for every ground $\Sigma_0^{\underline{a}}$-term t, there exists a $\Sigma_0^{\underline{a}}$-term $u \in \mathcal{T}'_{\underline{a}}$ such that $\mathcal{M}_i \models t = u$ for every $i \in \{1, \ldots, k\}$. Indeed, given a $\Sigma_0^{\underline{a}}$-term t, from the almost locally finiteness hypothesis we have that $T_0 \models t = t_1 \vee \cdots \vee t = t_n$ where $t_i \in \mathcal{T}_{\underline{a}}$, hence we have that $\mathcal{M}_1 \models t = t_{j_1}, \ldots, \mathcal{M}_k \models t = t_{j_k}$ ($t_{j_i} \in \{t_1, \ldots, t_n\}$), thus $\mathcal{T}(\langle t_{j_1}, \ldots, t_{j_k} \rangle) \neq \emptyset$. Consider now the element u added in $\mathcal{T}'_{\underline{a}}$ from $\mathcal{T}(\langle t_{j_1}, \ldots, t_{j_k} \rangle)$; it follows that, for each $i \in \{1, \ldots, k\}$, $\mathcal{M}_i \models t = u$ since $\mathcal{M}_i \models t = t_{j_i}$ and $\mathcal{M}_i \models u = t_{j_i}$. Hence it follows that $T_0 \models t = u$; indeed, suppose by contradiction that there exists a model $\mathcal{N}$ of T_0 such that $\mathcal{N} \models t \neq u$. Since T_0 is universal, the substructure of $\mathcal{N}$ generated by $\underline{a}$ is one of the $\mathcal{M}_i$'s; contradiction. □

Theorem 1.3.16 now easily follows from Propositions 1.3.18 and 1.3.19.

Noetherian Universal Horn Theories Now we want to give a semantic characterization of the universal Horn theories that are Noetherian. By universal Horn theory we mean a set of sentences that are universal closures of formulae of the kind

$$\psi_1 \wedge \cdots \wedge \psi_n \rightarrow \varphi$$

where the ψ_i's are atoms and φ is an atom. Once again, in the following we often do not distinguish between elements of a structure and their names in an expanded signature.

Let Σ_0 be a signature (for the sake of simplicity, we assume that Σ_0 contains only function symbols), T_0 be a universal Horn Σ_0-theory, $\underline{a}$ be a (possibly infinite) set of free constants, and finally let P be a set of ground $\Sigma_0^{\underline{a}}$-atoms. We define the following congruence $\sim$ over the set of ground $\Sigma_0^{\underline{a}}$-terms $\mathcal{T}$:

$$t \sim u \quad \text{iff} \quad T_0 \cup P \models t = u$$

It is easy to see that, since T_0 is a universal Horn theory, $\mathcal{T}/_\sim$ is a model of $T_0 \cup P$. Given a structure $\mathcal{M}$ isomorphic to $\mathcal{T}/_\sim$, the pair $\langle \underline{a}, P \rangle$ is called a *presentation* of $\mathcal{M}$, and the $\underline{a}$'s are called the *generators* of $\mathcal{M}$. If $\underline{a}$ and P are finite, $\mathcal{M}$ is said to be *finitely presented*. Notice that every structure $\mathcal{M} = \langle M, \mathcal{I} \rangle$ admits a presentation; indeed, consider the set $\Delta^+(\mathcal{M})$ of all Σ^M-atoms that hold in $\mathcal{M}$: clearly, $\langle M, \Delta^+(\mathcal{M}) \rangle$ is a presentation of $\mathcal{M}$.

Theorem 1.3.20. *Let T_0 be a universal Horn Σ_0-theory. T_0 is Noetherian if and only if every finitely generated model of T_0 is finitely presented.*

Proof. If T_0 is Noetherian, clearly every presentation $\langle \underline{a}, P \rangle$ of a finitely generated model of T_0 is equivalent to a finite one. For the converse, let

$$\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_n \subseteq \cdots$$

be a chain of sets of ground $\Sigma_0^{\underline{a}}$-atoms, and

$$P := \bigcup_i \Theta_i.$$

The pair $\langle \underline{a}, P \rangle$ is a presentation of the model $\mathcal{T}/_\sim$ built as above. Since $\mathcal{T}/_\sim$ is finitely generated, it admits a finite presentation, say $\langle \underline{a}, P' \rangle$.[11] By construction of $\mathcal{T}/_\sim$, it follows that $P = \bigcup_i \Theta_i$ is equivalent to the finite P' modulo T_0, thus the chain of Θ_i's is eventually constant modulo T_0. □

1.4 Examples

This section simply collects examples of theories satisfying the requirements of the notions presented in Section 1.2 (further details can be found in this thesis and in Ghilardi, 2003, 2004; Nicolini, 2007).

[11] For the sake of simplicity, we assume that the generators $\underline{a}$ of the two presentations $\langle \underline{a}, P \rangle$ and $\langle \underline{a}, P' \rangle$ are the same. If not, it is possible to adapt the proof using a compactness argument.

T_0-compatible Theories

1. T_0^* is a Σ_0-theory that eliminates quantifiers and T is any theory *whatsoever* in a bigger signature such that $T \supseteq T_0^*$. Then T is T_0-compatible (taking as T_0 the theory having as axioms all the universal Σ_0-sentences which are logical consequences of $T_0^\star$). Theories that admits quantifier elimination are, for example:
   - The theories of Integer, Rational, and Real Arithmetic;
   - The theory of acyclic binary lists (see Ghilardi, 2003, Appendix A);
   - Any theory axiomatizing enumerated datatypes.
2. T_0 is any theory that satisfies the requirements of Definition 1.2.1, and T is an extension of T_0 with free function symbols only. Then T is T_0-compatible. For example, T_0 can be:
   - The pure theory of equality ($T_0^\star$ is the theory of infinite sets);
   - The theory of integral domains ($T_0^\star$ is the theory of algebraically closed fields), the theory of torsion free abelian groups ($T_0^\star$ is the theory of divisible torsion free abelian groups);
   - The theory of Boolean algebras ($T_0^\star$ is the theory of atomless Boolean algebras);
   - Any universal and locally finite Horn theory T_0 (in a finite signature) such that the amalgamation property holds for models of T_0 (see Wheeler, 1976).[12]
3. Other examples:
   - T is the theory of Rational or Real Arithmetic, T_0 is the theory of linear orders;
   - T is the theory of Real Arithmetic, T_0 is the theory of Real Arithmetic without the ordering;
   - T is any stably infinite theory over a signature including a free unary function symbol f, T_0 is the empty theory over the signature $\{f\}$ (cf. Section 1.5);

[12] The amalgamation property state that, if $\mathcal{M}_1, \mathcal{N}_1, \mathcal{N}_2$ are models of T_0 and $\mu_1 : \mathcal{M}_1 \longrightarrow \mathcal{N}_1$, $\mu_2 : \mathcal{M}_1 \longrightarrow \mathcal{N}_2$ are embeddings, then there are a model $\mathcal{M}_2$ of T_0 and embeddings $\nu_1 : \mathcal{N}_1 \longrightarrow \mathcal{M}_2$ and $\nu_2 : \mathcal{N}_2 \longrightarrow \mathcal{M}_2$ such that $\nu_1\mu_1 = \nu_2\mu_2$.

- T is the theory of modal algebras, T_0 is the theory of Boolean algebras;
- T is the theory of K-algebras, T_0 is the theory of K-vector spaces (where K is a field).

Locally Finite Theories

- Any theory over a finite signature not containing function symbols (e.g., orders);
- Any theory axiomatizing enumerated datatypes;
- The theories of Arrays and Records (see, e.g., Armando et al., 2007) over pairwise distinct sorts;
- The theory of Linear Integer Arithmetic modulo a fixed integer;
- The theory of Boolean algebras.

Noetherian Theories

- The theory of Integer Offsets (see, e.g., Armando et al., 2007);
- The theories of K-algebras, K-vector spaces, and R-modules (where K is a field and R is a Noetherian ring);
- The theory of a single Associative-Commutative symbol (see Chenadec, 1986);
- The empty theory of a free unary function symbol (cf. Section 1.5);
- Any extension (in the same signature) of a Noetherian theory.

Effectively Noetherian Extensions

- T is the theory of Real Arithmetic, T_0 is the theory of Real Arithmetic without the ordering;
- T is the theory of K-algebras, T_0 is the theory of K-vector spaces (where K is a field);
- T is any stably infinite theory over a signature including a free unary function symbol f whose constraint satisfiability is decidable, T_0 is the empty theory over the signature $\{f\}$ (cf. Section 1.5);

– Any theory T with decidable constraint satisfiability problem which is compatible with an effectively locally finite theory T_0.

1.5 The Theory of a Free Unary Function Symbol

By collecting the observations above, it is easy to identify pairs of theories (T, T_0) such that $T_0 \subseteq T$ and T is a T_0-compatible effectively Noetherian extension of T_0. Here, we consider an entirely new (and somewhat remarkable) class of theories to obtain such pairs (T, T_0) of theories.

Let f be a unary function symbol. If T is a theory, then T_f is the theory obtained from T by adding f to its signature (as a new free function symbol). So, e.g., if E the empty theory over the empty signature, E_f denotes the empty theory over the signature $\{f\}$.

Proposition 1.5.1. *E_f is Noetherian.*

Proof. By contradiction, suppose that there is a chain $\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_n \subseteq \cdots$ of sets of ground $\Sigma^{\underline{a}}$-atoms which is not eventually constant for logical consequence w.r.t. T. Without loss of generality, we can assume that $\Theta_1 \subseteq \Theta_2 \subseteq \cdots \subseteq \Theta_n \subseteq \cdots$ is such that for each i there exists a $\Sigma^{\underline{a}}$-atom $\ell_i \in \Theta_i$ such that $T \cup \Theta_{i-1} \not\models \ell_i$.

Notice that, since f is a unary function symbol, each element of the infinite sequence $\{\ell_i\}_{i\in\mathbb{N}}$ of $\Sigma^{\underline{a}}$-atoms is a $\Sigma^{\{a_i,a_j\}}$-atom (for some $a_i, a_j \in \underline{a}$). Thus, since $\underline{a}$ is finite, we can extract an infinite subsequence of ground $\Sigma^{\{a,b\}}$-atoms (for some fixed elements $a, b \in \underline{a}$) inducing an infinite ascending chain $\Theta_{1_{|\Sigma^{\{a,b\}}}} \subseteq \Theta_{2_{|\Sigma^{\{a,b\}}}} \subseteq \cdots \subseteq \Theta_{n_{|\Sigma^{\{a,b\}}}} \subseteq \cdots$ which is not eventually constant for logical consequence w.r.t. T (here $\Theta_{i_{|\Sigma^{\{a,b\}}}}$ is the collection of all the ground $\Sigma^{\{a,b\}}$-atoms occurring in Θ_i).

Suppose that a $\Sigma^{\{a,b\}}$-atom of the kind $\ell := f^m(a) = f^n(a)$ occurs in such an infinite subsequence (here $m \neq n$ otherwise $T \models \ell$, contrary to our choice of these atoms). Notice that $T \cup \ell$ is such that there are only finitely many $\Sigma^{\{a\}}$-terms that are not logically equivalent w.r.t. $T \cup \ell$, which implies that every infinite ascending chain of sets of ground $\Sigma^{\{a\}}$-atoms is eventually constant for logical consequence w.r.t. $T \cup \ell$ (the same argument apply to atoms of the kind $\ell := f^m(b) = f^n(b)$).

Suppose now that a $\Sigma^{\{a,b\}}$-atom of the kind $\ell := f^m(a) = f^n(b)$ belongs to such an infinite chain of $\Sigma^{\{a,b\}}$-atoms. The only $\Sigma^{\{a,b\}}$-atoms of the form $f^{m'}(a) = f^{n'}(b)$ not implied by $T \cup \ell$ are such that either (i) $m - n \neq m' - n'$ or (ii) $m' < m$ and $n' < n$. It is clear that there are only finitely many atoms of the kind (ii); for (i), notice that

$f^m(a) = f^n(b) \wedge f^{m'}(a) = f^{n'}(b)$ implies that $f^{m+n'}(a) = f^{n+n'}(b) = f^{m'+n}(a)$ and that $f^{n+m'}(b) = f^{m+m'}(a) = f^{n'+m}(b)$ (where $m + n' \neq m' + n$ by (i)), so we are reduced to the first case.

The arguments above imply that the chain

$$\Theta_{1|\Sigma\{a,b\}} \subseteq \Theta_{2|\Sigma\{a,b\}} \subseteq \cdots \subseteq \Theta_{n|\Sigma\{a,b\}} \subseteq \cdots$$

is eventually constant for logical consequence w.r.t. T. Contradiction. □

In the following we assume the reader is familiar with the fundamentals of Superposition Calculus $\mathcal{SP}$,) as explained for instance in Nieuwenhuis and Rubio (2001) (see also the Appendix for a very brief overview on this topic). We shall be especially interested in the saturation (modulo redundancy) $\mathcal{SP}(\Gamma)$ of a finite set of ground literals Γ: we recall that this can be achieved by $\mathcal{SP}$ in finitely many steps with respect to any reduction ordering. In fact, on this kind of inputs, $\mathcal{SP}$ behaves like standard Knuth-Bendix completion (with simplification). We just fix the relevant facts for future reference:

Lemma 1.5.2 (Nieuwenhuis and Rubio, 2001)**.** *Let Γ be a consistent*[13] *ground constraint; given any reduction ordering total on ground terms, the saturation $\mathcal{SP}(\Gamma)$ of a Γ consists of a finite set R of equations and a finite set I of inequations such that:*

(i) Γ is logically equivalent to $I \cup R$;

(ii) the equation in R (once oriented from left to right) form a convergent ground rewriting system;

(iii) every equation $l = r \in R$ is in normal form with respect to $R \setminus \{l \to r\}$;

(iv) the inequations in I are in R-normal form;

(v) every positive clause C is a logical consequence of Γ iff there is a disjunct $s = t$ in C such that s and t have the same R-normal form.

For the last claim, notice that free theories are convex,[14] hence we have that $\Gamma \models C$ holds iff there is an equation $s = t$ in C such that $\Gamma \models s = t$ and the latter holds iff s and t have the same R-normal form.

[13] If Γ is not consistent, $\mathcal{SP}(\Gamma)$ just consists of the empty clause.

[14] A theory T is said to be convex iff whenever for a constraint Γ we have $T \cup \Gamma \models A_1 \vee \cdots \vee A_n$ (here the A_i are atoms and $n \geq 1$), then there is i such that $T \cup \Gamma \models A_i$. Among examples of convex theories, we have all Horn theories.

We recall that a theory T is stably infinite (see, e.g., Nelson and Oppen, 1979; Tinelli and Harandi, 1996) iff it is E-compatible, or, equivalently, iff any T-satisfiable constraint is satisfiable in a model of T whose domain is infinite.

Proposition 1.5.3. *If T is stably infinite and has decidable constraint satisfiability problem, then T_f is an effectively Noetherian extension of E_f.*

Proof. Let Γ be a T_f-constraint (we write $\Gamma(\underline{a},\underline{b})$ to emphasize that the free constants occurring in Γ are in the tuple $\underline{a},\underline{b}$): we want to compute an E_f-basis of Γ w.r.t. $\underline{a}$. Notice that $T_f = T \cup E_f$: since both theories are stably infinite and their intersection is E, Nelson-Oppen results apply. In particular, the following is a decision procedure for T_f-consistency of Γ (see, e.g., Ghilardi, 2004; Nelson and Oppen, 1979; Tinelli and Harandi, 1996):

(a) produce a T-constraint $H(\underline{a},\underline{b},\underline{c})$ and an E_f-constraint $L(\underline{a},\underline{b},\underline{c})$ such that $\Gamma(\underline{a},\underline{b})$ is logically equivalent to $\exists \underline{x}(H(\underline{a},\underline{b},\underline{x}) \wedge L(\underline{a},\underline{b},\underline{x}))$ (this is a standard purification step);

(b) guess an $(\underline{a},\underline{b},\underline{c})$-arrangement $G(\underline{a},\underline{b},\underline{c})$ (an $(\underline{a},\underline{b},\underline{c})$-arrangement is a set of literals containing for each $c_1, c_2 \in \underline{a} \cup \underline{b} \cup \underline{c}$ either $c_1 = c_2$ or $c_1 \neq c_2$);

(c) check $H(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$ for T-satisfiability and $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$ for E_f-satisfiability;

(d) output *satisfiable* iff both tests are successful and *unsatisfiable* iff they fail for all arrangements.

The correctness of the procedure is obvious, its completeness is due to the fact that, given a T-model $\mathcal{M}$ for $H(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$ and an E_f-model $\mathcal{N}$ for $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$, one can produce out of them a T_f-model $\mathcal{G}$ whose reducts to the signatures of T and of E_f are such that $\mathcal{M}$ and $\mathcal{N}$ respectively embed into them. The argument is the following: one can suppose that $\mathcal{M}$, $\mathcal{N}$ to be both infinite and of the same cardinality (by stable infiniteness and Löwenheim-Skolem theorem). Then, one can simply glue them because (up to renaming) they agree on the interpretation of the shared constants $\underline{a},\underline{b},\underline{c}$. Notice that stable infiniteness of a theory T can be formulated either by saying that every constraint is satisfiable in an infinite model of T or by saying that every model of T embeds into an infinite model of T (the equivalence of the two statements follows from the diagram theorem and compactness).

Notice that the E_f-satisfiability test for $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$ can be obtained through Superposition: when doing that, we use a lexicographic path ordering (see Baader and Nipkow, 1998) induced by a precedence giving the $\underline{b},\underline{c}$'s higher precedence with respect to both f and the $\underline{a}$'s. As a consequence, Lemma 1.5.2(v) immediately implies the following:

Claim. Let $B_G(\underline{a})$ be the set of equations from $\mathcal{SP}(L(\underline{a},\underline{b},\underline{c})\wedge G(\underline{a},\underline{b},\underline{c}))$ not involving the $\underline{b},\underline{c}$. We have that a positive clause $C(\underline{a})$ is a logical consequence of $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$ iff $B_G(\underline{a}) \models C(\underline{a})$.

We now show that $\bigvee_G B_G(\underline{a})$ is an E_f-basis for $\Gamma(\underline{a},\underline{b})$ with respect to $\underline{a}$ (the index G ranges over all arrangements for which the consistency tests in (c) are both positive).[15]

That $T_f \cup \{\Gamma(\underline{a},\underline{b})\} \models \bigvee_G B_G(\underline{a})$ is clear: by (a), $\Gamma(\underline{a},\underline{b})$ is logically equivalent to $\exists \underline{x}(H(\underline{a},\underline{b},\underline{x}) \wedge L(\underline{a},\underline{b},\underline{x}))$, the latter is equivalent to $\exists \underline{x}(H(\underline{a},\underline{b},\underline{x}) \wedge L(\underline{a},\underline{b},\underline{x}) \wedge \bigvee_G G(\underline{a},\underline{b},\underline{x}))$ and finally $L(\underline{a},\underline{b},\underline{c}) \wedge \bigvee_G G(\underline{a},\underline{b},\underline{c})$ entails $\bigvee_G B_G(\underline{a})$.

Conversely, suppose that $C(\underline{a})$ is a positive E_f-clause such that $T_f \cup \{\Gamma(\underline{a},\underline{b})\} \models C(\underline{a})$; we need to show that $B_G(\underline{a}) \models C(\underline{a})$ for any given arrangement $G(\underline{a},\underline{b},\underline{c})$ (such that both consistency tests in (c) are positive). We first show that $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c}) \models C(\underline{a})$: to see this, let $\mathcal{N}$ be an arbitrary model of $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$. Since the first consistency test in (c) is positive, there is a T-model $\mathcal{M}$ of $H(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c})$: by the above Nelson-Oppen combination argument, there is a model $\mathcal{G}$ of T_f whose reducts to the signatures of T and of E_f are such that $\mathcal{M}$ and $\mathcal{N}$ respectively embed into them. Since $\mathcal{G}$ is a model of T_f and $\Gamma(\underline{a},\underline{b})$, $\mathcal{G} \models C(\underline{a})$, hence also $\mathcal{N} \models C(\underline{a})$ (because $\mathcal{N}$ embeds into the E_f-reduct of $\mathcal{G}$); being $\mathcal{N}$ arbitrary, this means that $L(\underline{a},\underline{b},\underline{c}) \wedge G(\underline{a},\underline{b},\underline{c}) \models C(\underline{a})$. But now the above Claim shows that $B_G(\underline{a}) \models C(\underline{a})$. □

To the aim of proving Theorem 1.5.6, we need to introduce the theory $E_f^\star$ and to prove that it admits quantifier elimination. The theory $E_f^\star$ in the signature consisting of a unary function symbol f says the following:

(i) for each positive integer n there exist infinite elements x such that $f^n(x) = x$ and $f^m(x) \neq x$ (for all $0 < m < n$);

(ii) every element x is of the form $f(y)$ for infinitely many y.

[15] Of course, if there are none of them, the index set is empty and $\bigvee_G B_G(\underline{a})$ is the empty disjunction, namely $\bot$. Formally, the notion of an E_f-basis requires a set of clauses, hence $\bigvee_G B_G(\underline{a})$ should be brought in conjunctive normal form.

$E_f^\star$ is a consistent theory: this is shown by producing a chain of E_f-models whose union is a model of $E_f^\star$ (the first model of the chain consists of infinitely many loops of any finite size, the $(i+1)^{\text{th}}$ model is obtained by adding an f-predecessor to any element of the i^{th} model).

Lemma 1.5.4. *The theory $E_f^\star$ admits quantifier elimination; moreover, every model of E_f embeds into a model of $E_f^\star$.*[16]

Proof. We first show how to reduce the whole statement of the lemma to the following

Claim. Suppose that $\Gamma(a, b_1, \ldots, b_k)$ is a constraint satisfying the following conditions: (i) the free constant a occurs in all literals from Γ; (ii) Γ is saturated (i.e. $\mathcal{SP}(\Gamma) = \Gamma$) with respect to the lexicographic path ordering induced by the precedence

$$a > b_1 > \cdots > b_k > f.$$

Then $E_f^\star \models \forall y_1 \cdots \forall y_k \exists x\, \Gamma(x, y_1, \ldots, y_k)$.

If the Claim holds, we can eliminate quantifiers from any simply primitive formula $\exists x G(x, y_1, \ldots, y_k)$ as follows: first, saturate $G(a, b_1, \ldots, b_k)$ and then, keep only the literals not involving a (or output $\bot$ if the saturation produces the empty clause). The Claim shows also that every model $\mathcal{M}$ of E_f embeds into a model of $E_f^\star$: in fact, $E_f^\star$ is consistent and hence (by the above argument) consistent with the diagram of $\mathcal{M}$.

Thus, it only remains to prove the Claim: let $\Gamma(a, b_1, \ldots, b_k)$ be a constraint satisfying the two conditions of the Claim. By our choice of the reduction ordering, it is straightforward to see that (a) $f^n(a) > f^m(b_i)$ for each b_i and $n, m \geq 0$ and (b) $f^n(c) > f^m(c)$ iff $n > m$ for each constant c. Now, since Γ is saturated and all literals from Γ contains an occurrence of a, we see that Γ is either of the kind

$$\{f^m(a) = u, f^{m-k_1}(a) \neq u_1, \ldots, f^{m-k_n}(a) \neq u_n\}$$

or of the kind

$$\{f^{m_1}(a) \neq u_1, \ldots, f^{m_n}(a) \neq u_n\}$$

(here $n, m, m_i \geq 0$ and $0 < k_i \leq m$). Indeed, by contradiction, suppose that two equalities involving a occur in Γ or that the equality $f^m(a) = u$ and an inequality of

[16] The reader interested in a purely model-theoretic proof of the model-completability of the 'loop-free extension' of E_f can consult Hodges (1993).

the kind $f^{m+k}(a) \neq t$ occur in Γ; in both cases, by our hypothesis of the ordering, a occurs in the maximum term of the equations, thus a reduction rewriting rule would apply, contradicting the fact that Γ is saturated. To simplify the matter further, notice that we can get rid of the case in which the equation $f^m(a) = u$ does not appear, because we can add it freely, taking as u the constant b_{k+1} which is not among the original $b_1, \ldots, b_k$ (proving the claim for this case would in fact be stronger).

We now distinguish two cases, depending on the form of the term u occurring in the only equation $f^m(a) = u$ of Γ:

(i) *a does not occur in u* (that is, u is of the form $f^l(b_j)$): the constraint Γ is

$$\{f^m(a) = u, f^{m-k_1}(a) \neq u_1, \ldots, f^{m-k_n}(a) \neq u_n\}.$$

Pick a model $\mathcal{M}$ of $E_f^\star$ and for simplicity let us indicate directly with $b_1, \ldots, b_k$ a given k-tuple of elements of the support of $\mathcal{M}$: we must show that we can find a so that $\mathcal{M} \models \Gamma(a, b_1, \ldots, b_k)$. Notice that any term t not involving a is of the kind $f^j(b_i)$ and hence gets interpreted as a specific element of $\mathcal{M}$ (that we still call t), because $b_1, \ldots, b_k$ have been assigned an interpretation. We let X be the set of such terms among the $u, u_1, \ldots, u_n$ (notice that the complement set $\{u, u_1, \ldots, u_n\} \setminus X$ is formed by terms of the kind $f^j(a)$, where $j < m$).[17]

By induction, we define elements $a_m, a_{m-1}, \ldots, a_1, a_0$ in the following way: we let a_m to be u and, when defining a_{i-1} we choose it in such a way that $f^{\mathcal{M}}(a_{i-1}) = a_i$ and a_{i-1} is different from all interpretations of elements from X and also from $a_m, \ldots, a_i$: this is possible by the second group of axioms for $E_f^\star$. If we let a to be a_0, it is clear that $\mathcal{M} \models \Gamma(a, b_1, \ldots, b_k)$ holds (saturation prevents the constraint from containing inconsistent inequations like $t \neq t$).

(ii) *a occurs in u* (that is, u is of the form $f^{m-l}(a)$, for $0 < l \leq m$): the constraint Γ is

$$\{f^m(a) = f^{m-l}(a), f^{m-k_1}(a) \neq u_1, \ldots, f^{m-k_n}(a) \neq u_n\}.$$

Again we pick a model $\mathcal{M}$ of $E_f^\star$, a k-tuple $b_1, \ldots, b_k$ of elements from the support of $\mathcal{M}$, and we still follow the convention of indicating with t the resulting interpretation of terms t of the kind $f^j(b_i)$ (we also collect in a set called X

[17] We cannot have $j \geq m$, otherwise the constraint would not be saturated (a rewriting demodulation applies).

these terms). We have to find a in such a way that $\mathcal{M} \models \Gamma(a, b_1, \ldots, b_k)$ holds.

By the first group of axioms for $E_f^\star$, it is possible to pick a loop of length l formed by elements $a_{m-1}, \ldots, a_{m-l}$ which are pairwise distinct from each other and also distinct from the interpretations of the terms in X. We then define, by induction, elements $a_{m-l}, a_{m-l-1}, \ldots, a_1, a_0$ as in the previous case, starting from the already defined element a_{m-l}. If we finally take a to be a_0, we can ensure the condition $\mathcal{M} \models \Gamma(a, b_1, \ldots, b_k)$.

□

Proposition 1.5.5. *If T is stably infinite, then T_f is E_f-compatible.*

Proof. We need to show that:

(i) $E_f \subseteq E_f^\star$;

(ii) $E_f^\star$ has quantifier elimination;

(iii) every model of E_f can be embedded into a model of $E_f^\star$;

(iv) every model of T_f can be embedded into a model of $T_f \cup E_f^\star$.

We already know that (i) to (iii) hold from Lemma 1.5.4.

To show (iv), let $\mathcal{M}_0 = (M_0, \mathcal{I}_0)$ be a model of $T_f = T \cup E_f$. Take models $\mathcal{M}_1, \mathcal{M}_2$ such that: (1) $\mathcal{M}_1$ is an infinite model of T such that the reduct of $\mathcal{M}_0$ to the signature Σ of T embeds into $\mathcal{M}_1$ (it exists because T is stably infinite); (2) $\mathcal{M}_2$ is a model of $E_f^\star$ such that the reduct of $\mathcal{M}_0$ to the signature $\{f\}$ of E_f embeds into $\mathcal{M}_2$ (it exists by (iii) above).

We are now in the position of applying Lemma 1.3.2: we take $I := \{1, 2\}$, $\underline{c} := M_0$, $\Sigma_1 := \Sigma$, $\Sigma_2 := \{f\}$, $\Sigma_0 := \emptyset$, $\underline{a}_1 := \underline{a}_2 := \emptyset$, $T_1 := T$, $T_2 := E_f^\star$, $T_0 := E$. The hypotheses of Lemma 1.3.2 are satisfied because T_1, T_2 are both stably infinite (alias E-compatible), hence there exists $\mathcal{M} \models T \cup E_f^\star$ such that $\mathcal{M}_0$ has a $\Sigma \cup \{f\}$-embedding into $\mathcal{M}$: in fact, for $i = 1, 2$, $\mathcal{M}_0$ has a Σ_i-embedding into $\mathcal{M}_i$ and the latter $\Sigma_i^{M_0}$-embeds into $\mathcal{M}$. □

We are now ready to characterize our new class of theories.

Theorem 1.5.6. *Let T be a theory with decidable constraint satisfiability problem. If T is stably infinite, then T_f is an effectively Noetherian extension of E_f, which is also E_f-compatible.*

Proof. By Propositions 1.5.3 and 1.5.5. □

This result is a first step towards the integration in our framework of some theories that are useful for verification. For example, the theory of integer offsets can be seen as an extension of the theory of a loop-free unary function symbol (see, e.g., Armando et al., 2007). Properties of hardware systems can be expressed in a mixture of temporal logic – e.g., LTL or CTL – and the theory of integer offsets (see Bryant et al., 2002). Our decidability results on "temporalized" first-order theories below (cf. Theorems 2.4.5 and 2.5.4) can then be used to augment the degree of automation of tools attempting to solve this kind of verification problems.

1.6 Conclusions

In this chapter we have presented the basic definitions that will be used in the following and we have reviewed the combination results for the non-disjoint case that inspire the rest of the thesis. For the ease of the reader, we have collected a series of examples of theories fitting our 'combinability' hypotheses.

Two original contributions are also included: first, we have shown that, under suitable hypothesis of T_0-compatibility, the property of being an effective Noetherian extension of a theory is modular. Secondly, an important class of stably infinite theories extending the empty theory over a single unary function symbol has been shown to satisfy the hypotheses for the decidability of both the non-disjoint combination schema and, as we will show in the next chapter, the satisfiability of "temporalized" first-order theories.

Chapter 2

Satisfiability in Temporal and Modal Logics

One of the aim of this thesis is to study reactive systems by combining temporal operators and a first-order language. Since full first-order temporal logics are known to be highly undecidable (even Π^1_1-complete), researchers concentrated on finding fragments having good computational properties, such as the decidable monodic fragments investigated in, e.g., Degtyarev et al. (2006); Gabbay et al. (2003); Hodkinson et al. (2000). Although such fragments may also be used in verification, widely adopted formalisms for the specification of reactive or distributed systems (e.g., the one proposed in Manna and Pnueli, 1995 or the Temporal Logic of Actions in Lamport, 1994) are such that the temporal part, used to describe the dynamic behavior of the systems, is parametric with respect to the underlying language of first-order logic, and theories in first-order languages with equality are often needed to formalize the data structures manipulated by the systems. While the expressiveness of these formalisms helps in writing concise and abstract specifications, it is not clear how these can be amenable to automated analysis, since we will show that the fragments so obtained are in general recursively enumerable.

The work presented in this chapter contributes towards the solution of this problem, by analyzing what happens when we "add a temporal dimension" (in a sense similar to that investigated in Finger and Gabbay, 1992) to a decidable fragment of a first-order theory with identity. In the following, relying on the techniques developed in the previous chapter, we derive an undecidability result and then we identify suitable condition that allows to transfer the decidability of first-order fragments to

their temporalized version.

2.1 Temporalizing a First-Order Theory

As argued in Manna and Pnueli (1995, page 48), for most applications it is sufficient to fix a first-order signature Σ and to deal with formulae obtained by applying temporal and Boolean operators (but no quantifiers) to first-order Σ-formulae: the resulting formulae are called *state-quantified formulae* in Manna and Pnueli (1995) and are formally introduced as follows.

Definition 2.1.1 (LTL($\Sigma^{\underline{a}}$)-Sentences)**.** Given a signature Σ and a (finite or infinite) set of free constants $\underline{a}$, the set of LTL($\Sigma^{\underline{a}}$)-sentences is inductively defined as follows: (a) if φ is a first-order $\Sigma^{\underline{a}}$-sentence, then φ is an LTL($\Sigma^{\underline{a}}$)-sentence and (b) if ψ_1, ψ_2 are LTL($\Sigma^{\underline{a}}$)-sentence, so are $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\neg\psi_1$, $X\psi_1$, $\Box\psi_1$, $\Diamond\psi_1$, $\psi_1 U \psi_2$.

Notice that free constants are allowed in the definition of an LTL($\Sigma^{\underline{a}}$)-sentence. This is quite conventional: since we prefer not to use free variables, free constants handle variables and parameters of the system to be modeled.

Let us now discuss *semantic* issues. It is clear that an LTL($\Sigma^{\underline{a}}$)-structure must be a family of $\Sigma^{\underline{a}}$-structures $\mathcal{M} = \{\mathcal{M}_n = (M_n, \mathcal{I}_n)\}_{n\in\mathbb{N}}$ indexed by the natural numbers; when we fix also a background Σ-theory T, these structures will be taken to be models of T. The main question is the following: what should the various $\mathcal{M}_n$ share? A first requirement is that they should share their domains, that is we assume the M_n to be *constant*, i.e. all equal to each other. Although different semantics, with increasing and even distinct domains, have been proposed in the literature (see, e.g., Bräuner and Ghilardi, 2007), the constant domain assumption is rather common in computer science applications.

Definition 2.1.2. Given a signature Σ and a set $\underline{a}$ of free constants, an *LTL($\Sigma^{\underline{a}}$)-structure* (or simply a *structure*) is a sequence $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n\in\mathbb{N}}$ of $\Sigma^{\underline{a}}$-structures. The set M is called the *domain* (or the *universe*) and $\mathcal{I}_n$ is called the n^{th} *level interpretation function* of the LTL($\Sigma^{\underline{a}}$)-structure.[1]

When considering a background Σ-theory T, the structures $\mathcal{M}_n = (M_n, \mathcal{I}_n)$ will be taken to be models of T (further requirements will be analyzed later on).

[1] In more detail, $\mathcal{I}_n$ is such that $\mathcal{I}_n(P) \subseteq M^k$ for every predicate symbols $P \in \Sigma$ of arity k, and $\mathcal{I}_n(f) : M^k \longrightarrow M$ for each function symbol $f \in \Sigma$ of arity k.

Definition 2.1.3. Given an LTL($\Sigma^{\underline{a}}$)-sentence φ and $t \in \mathbb{N}$, the notion of "φ being true in the LTL($\Sigma^{\underline{a}}$)-structure $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n\in\mathbb{N}}$ at the instant t" (in symbols $\mathcal{M} \models_t \varphi$) is inductively defined as follows:

- if φ is an first-order sentence, $\mathcal{M} \models_t \varphi$ iff $\mathcal{M}_t \models \varphi$;
- $\mathcal{M} \models_t \neg\varphi$ iff $\mathcal{M} \not\models_t \varphi$;
- $\mathcal{M} \models_t \varphi \wedge \psi$ iff $\mathcal{M} \models_t \varphi$ and $\mathcal{M} \models_t \psi$;
- $\mathcal{M} \models_t \varphi \vee \psi$ iff $\mathcal{M} \models_t \varphi$ or $\mathcal{M} \models_t \psi$;
- $\mathcal{M} \models_t X\varphi$ iff $\mathcal{M} \models_{t+1} \varphi$;
- $\mathcal{M} \models_t \Box\varphi$ iff for each $t' \geq t$, $\mathcal{M} \models_{t'} \varphi$;
- $\mathcal{M} \models_t \Diamond\varphi$ iff for some $t' \geq t$, $\mathcal{M} \models_{t'} \varphi$;
- $\mathcal{M} \models_t \varphi U \psi$ iff there exists $t' \geq t$ such that $\mathcal{M} \models_{t'} \psi$ and for each t'', $t \leq t'' < t'$ $\Rightarrow \mathcal{M} \models_{t''} \varphi$.

The definition above is well given because, if the main connective of the formula is a Boolean operator, the definition of truth of an LTL($\Sigma^{\underline{a}}$)-sentence coincides with truth clause of Tarski semantics for first order languages. Let φ be an LTL($\Sigma^{\underline{a}}$)-sentence; we say that φ *is true in* $\mathcal{M}$ or, equivalently, that $\mathcal{M}$ *satisfies* φ (in symbols $\mathcal{M} \models \varphi$) iff $\mathcal{M} \models_0 \varphi$.

Let us now better examine the problem of the relationship between the interpretations $\mathcal{I}_n$ in an LTL($\Sigma^{\underline{a}}$)-structure: there are two radically opposite alternatives to cope with this problem. The customary Kripkean semantics for modal logics mostly deals with purely relational signatures and leave the interpretation of the predicate symbols *flexible*, i.e. time-dependant: no relationship among $\mathcal{I}_m(P)$ and $\mathcal{I}_n(P)$ is assumed for $n \neq m$. By contrast, constants are usually interpreted *rigidly* according to the orthodox Kripkean viewpoint, that is we have $\mathcal{I}_m(c) = \mathcal{I}_n(c)$ for all m, n and for all constants c.

On the other hand, the verification literature tends to consider the opposite solution: free constants are flexible (because the system variables are subject to change during runs) and symbols from Σ are rigidly interpreted, because they are supposed to model datatypes endowed with the corresponding time-independent operations (such as sum and successor for integers, read/write for arrays, etc.).

While keeping the same motivations of the verification literature, we adopt here a more elaborated point of view, according to which *certain symbols are declared rigid and the remaining ones are declared flexible* (i.e. time-dependent). We believe that there are various reasons supporting this choice. First of all, flexible interpretations are already used within the verification literature, where not only variables, but also propositions expressing program locations are in fact interpreted in a time-dependent way (to this aim, the Booleans sort is introduced in order to assimilate program locations to flexible variables). Moreover, reactive systems are supposed to interact with the environment and the environment action is somewhat unpredictable, to the point that it is better to model it through flexible function symbols – these function symbols obeying only to the constraints expressed by the background theory T or by the nondeterministic transition relation of the system (to see an example of what we mean, cf. the functions *in* and *out* within the water level controller example discussed in Section 3.4 below). Even predicates or function symbols expressing the internal evolution of the system may be subject to time change. Consider for instance a mutual exclusion protocol, like the 'bakery' protocol: here the set of processors wanting to enter into the critical section is variable and the ticket-assigning function is time-dependent too, for example because it need complete reset once the resource have been obtained (cf. again Section 3.4 for details). In these examples, the *constrained flexibility approach* we propose identifies the good *abstraction* level for the specification of the system behavior. Finally, there are also technical reasons supporting our proposal: big decidability problems in model checking arise when even minimal infinite states descriptors enter into the picture (cf. the proof of Theorem 3.2.1 below) and our setting allows to model the system by grouping problematic descriptors into two categories, the rigid and the flexible ones. As we shall see, if we succeed in keeping the rigid part of the specification relatively simple (i.e. 'locally finite'), then we do not loose the nice properties of the reasoning about finite-state specifications.

Definition 2.1.4. A *data-flow theory* is a 5-tuple $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ where Σ is a signature, T is a Σ-theory (called the underlying theory of $\mathcal{T}$), Σ_r is the *rigid subsignature* of Σ, $\underline{a}$ is a set of free constants (called *system variables*), and $\underline{c}$ is a set of free constants (called *system parameters*).

Σ_r is said to be the *rigid subsignature* of the data-flow theory; the constants $\underline{c}$ will be rigidly interpreted, whereas the constants $\underline{a}$ will be interpreted in a time-dependant way. A data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is *totally flexible* iff Σ_r is

empty and is *totally rigid* iff $\Sigma_r = \Sigma$.

Definition 2.1.5. An LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n\in\mathbb{N}}$ is *appropriate* for a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c}\rangle$ iff for all $m, n \in \mathbb{N}$, for all function symbol $f \in \Sigma_r$, for all relational symbol $P \in \Sigma_r$, and for all constant $c \in \underline{c}$, we have

$$\mathcal{M}_n \models T, \quad \mathcal{I}_n(f) = \mathcal{I}_m(f), \quad \mathcal{I}_n(P) = \mathcal{I}_m(P), \quad \mathcal{I}_n(c) = \mathcal{I}_m(c).$$

The *satisfiability problem for* $\mathcal{T}$ is the following: given an LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ, decide whether there is an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate for $\mathcal{T}$ such that $\mathcal{M} \models \varphi$. The *ground satisfiability problem for* $\mathcal{T}$ is similarly introduced, but φ is assumed to be ground.

Notice that appropriate structures are such that the equality symbol is always interpreted as the identity relation, since the equality is included in every signature (hence also in the rigid signature Σ_r).

In order to clarify the notions we have introduced so far with the help of an example, we foresee Example 3.4.1 from Chapter 3:

Example (Sofronie-Stokkermans, 2006). Consider a water level controller modeled as follows:

- changes in the water level by inflow/outflow are represented as functions *in* and *out* depending on the water level l and on the time instant; alarm and overflow levels $l_{\text{alarm}} < l_{\text{overflow}}$ are known;
- if the water level l is such that $l \geq l_{\text{alarm}}$ at a given state, then a valve is opened and the water level changes at the next observable time by $l' = in(out(l))$;
- if $l < l_{\text{alarm}}$ then the valve is closed; the water level changes at the next observable time by $l' = in(l)$.

The dependency of the functions *in* and *out* on the time instant means precisely that they can be modeled as *flexible* function symbols depending only on the water level. However, functions *in* and *out* cannot be completely uninterpreted, we impose the following restrictions on them:

$$\forall x\,(x < l_{\text{alarm}} \rightarrow in(x) < l_{\text{overflow}}) \tag{2.1}$$

$$\forall x\,(x < l_{\text{overflow}} \rightarrow out(x) < l_{\text{alarm}}) \tag{2.2}$$

In order to formalize the problem in our framework, we consider the data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ such that

- $\Sigma = \{in, out, l_{\text{alarm}}, l_{\text{overflow}}, <\}$ where in, out are two unary function symbols, $l_{\text{alarm}}, l_{\text{overflow}}$ are two constant symbols, $<$ is a binary predicate symbol;
- $\Sigma_r = \{l_{\text{alarm}}, l_{\text{overflow}}, <\}$;
- $T = T' \cup \{(2.1), (2.2)\}$ where T' is the theory of dense linear orders without endpoints endowed with the further axiom $l_{\text{alarm}} < l_{\text{overflow}}$ (see Example 3.4.1 for further details about the choice of T');
- l is the only system variable and there are no system parameters (that is, $\underline{a} := \{l\}$ and $\underline{c} := \emptyset$).

2.1.1 Recursive Enumerability of the Validity Problem

In general, the validity problem for first-order LTL with constant domains is Π^1_1-complete (see Bräuner and Ghilardi, 2007; Gabbay et al., 2003 for the upper and lower bound respectively). Since in our context we forbid any interplay between first-order quantifiers and temporal operators, the question whether the validity problem for LTL($\Sigma^{\underline{a}}$)-sentences belongs is Π^1_1-complete as well naturally arises. Given the data-flow theory $\mathcal{T} = \langle \Sigma, \emptyset, \Sigma_r, \underline{a}, \underline{c} \rangle$ and the LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ, we are interested in deciding whether each LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate for $\mathcal{T}$ is such that $\mathcal{M} \models \varphi$. Indeed, we briefly sketch the proof of the fact that this problem is Σ^0_1-complete (i.e., recursively enumerable); to this aim, we rely on the notion of PLTL-formulae that will be introduced below in Subsection 2.3.1.

From a bijective correspondence between propositional letters and first-order $\Sigma^{\underline{a},\underline{c}}$-sentence that are atoms or whose main connective is a quantifier, we define by recursion in the natural way the bijection $[\![\cdot]\!]$ (called propositional abstraction in the following) between LTL($\Sigma^{\underline{a},\underline{c}}$)-sentences and PLTL-formulae.[2] Let $\hat{G}$ be the set of all guessings over the propositional letters occurring in the propositional abstraction $[\![\varphi]\!]$ of φ. We have that φ is satisfiable in an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate

[2] More precisely, let μ be the correspondence between propositional letters and first-order $\Sigma^{\underline{a},\underline{c}}$-sentence that are atoms or whose main connective is a quantifier; we define the propositional abstraction $[\![\varphi]\!]$ of φ in the following way: (i) if φ is a ground atom or its main connective is a quantifier, then $[\![\varphi]\!] = \mu(\varphi)$; (ii) if φ is of the kind $\circ\psi$ where $\circ \in \{\neg, X, \Box, \Diamond\}$, then $[\![\varphi]\!] = \circ[\![\psi]\!]$; (iii) if φ is of the kind $\psi_1 \circ \psi_2$ where $\circ \in \{\wedge, \vee, U\}$, then $[\![\varphi]\!] = [\![\psi_1]\!] \circ [\![\psi_2]\!]$.

for $\mathcal{T}$ iff there exist $G_1, \ldots, G_k$ ($G_i \in \hat{G}$) such that (i) $[\![\varphi]\!] \wedge \Box(\bigvee_{i=1}^{k} G_i)$ is PLTL-satisfiable and (ii) $\bigwedge [\![G_i]\!]_i^{-1}$ is satisfiable (where $[\![\cdot]\!]_i^{-1}$ is the inverse of the function $[\![\cdot]\!]$ in which every flexible symbol $s \in \Sigma^{\underline{a}} \setminus \Sigma_r$ is renamed to s^i).

The 'only if' case is trivial, whereas for the converse we can argue as follows. From (i) there exists an infinite sequence of Boolean assignments $V := V_0 \rightarrow V_1 \rightarrow \cdots \rightarrow V_n \rightarrow \ldots$ that is a PLTL model for $[\![\varphi]\!] \wedge \Box(\bigvee_{i=1}^{k} G_i)$; by (ii), there exists a structure $\mathcal{M}$ such that $\mathcal{M} \models \bigwedge_{i=1}^{k} [\![G_i]\!]_i^{-1}$. Let $\mathcal{M}_i$ be the $\Sigma_i^{\underline{a},\underline{c}}$-reduct of $\mathcal{M}$ (where $\Sigma_i^{\underline{a},\underline{c}}$ is obtained from $\Sigma^{\underline{a},\underline{c}}$ by renaming to s^i each flexible symbol $s \in \Sigma^{\underline{a}} \setminus \Sigma_r$). Let us build the sequence $\mathcal{N} := \mathcal{N}_0 \rightarrow \mathcal{N}_1 \rightarrow \cdots \rightarrow \mathcal{N}_n \rightarrow \ldots$ such that $\mathcal{N}_i = \mathcal{M}_j$ iff $V_i \models G_j$ ($j \in \{1, \ldots, k\}$). By construction, $\mathcal{N}$ is appropriate for $\mathcal{T}$, and by adapting Lemma 2.3.3 below, it can be easily seen that $\mathcal{N} \models \varphi$.

Since the satisfiability problem for propositional LTL is decidable (more precisely it is PSPACE-complete, see Sistla and Clarke, 1985) and the satisfiability problem for first-order logic is Π_1^0-complete (i.e., co-recursively enumerable), it follows from the argument above that the satisfiability problem for LTL($\Sigma^{\underline{a}}$)-sentences is Π_1^0-complete, hence the validity problem for LTL($\Sigma^{\underline{a}}$)-sentences is Σ_1^0-complete. This argument applies also to the validity problem for LTL($\Sigma^{\underline{a}}$)-sentence modulo a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ where T is recursively axiomatized. In such a case, condition (ii) above becomes (ii') $\bigwedge [\![G_i]\!]_i^{-1}$ is $\bigcup_{i=1}^{k} T^i$-satisfiable, where T^i is obtained from T by renaming every flexible symbol $s \in \Sigma \setminus \Sigma_r$ occurring in the axioms of T in s^i.

In the following, we focus on classes of data-flow theories whose satisfiability problem for LTL($\Sigma^{\underline{a}}$)-sentences is decidable.

2.1.2 Some Classes of Data-Flow Theories and Further Assumptions

To study the ground satisfiability problem for data-flow theories, it is useful to distinguish three different classes of data-flow theories of increasing expressiveness and to lift to the temporal level the properties of (first-order) theories ensuring modularity (with respect to unions of theories) of decidability of constraint satisfiability problem (cf. Subsections 1.2.1 and 1.2.2 in Chapter 1).

Let Σ be a finite signature; an *enumerated datatype theory* in the signature Σ is the theory consisting of the set of sentences which are true in a finite given fixed Σ-structure $\mathcal{M} = (M, \mathcal{I})$ (we require $\mathcal{M}$ to have the additional property that for every $m \in M$ there is $c \in \Sigma$ such that $c^{\mathcal{M}} = m$). It is easy to see that an enumerated

datatype theory has a finite set of universal axioms and enjoys quantifier elimination.

Definition 2.1.6. A data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is said to be *finite state* iff it is totally rigid and T is an enumerated datatype theory.

Notice that enumerated datatype theories are locally finite, but not conversely;[3] thus, in order to generalize finite-state systems, one can require the underlying theory to be locally finite. We also want to drop the total rigidity requirement and weaken the quantifier elimination property of enumerated datatype theories to a compatibility requirement (recall Definition 1.2.1):

Definition 2.1.7. A data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is said to be *locally finite compatible* iff there is a universal and effectively locally finite Σ_r-theory T_r such that T is T_r-compatible.

Notice that, from our discussion in Section 1.2, it follows that a totally flexible data-flow theory is locally finite compatible in case its underlying theory is stably infinite.

We can get a further generalization by weakening local finiteness to Noetherianity (in the sense of Definition 1.2.8):

Definition 2.1.8. A data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is said to be *Noetherian compatible* iff there is a Σ_r-universal theory T_r such that T is an effectively Noetherian and T_r-compatible extension of T_r.

Definitions 2.1.7 and 2.1.8 refer to a Σ_r-theory T_r such that T is T_r-compatible. Although this is not relevant for the proofs of the results in this thesis, we notice that if such a theory T_r exists, then one can always take T_r to be the theory axiomatized by the universal Σ_r-sentences which are logical consequences of T.

We completed our conceptual setting: we need however to restrict it considerably, in order to be able to provide positive results. This is partially done by means of the following further assumption, to be kept in mind for the whole chapter.

Assumption 2.1.9. We shall concentrate on *ground* satisfiability problems. For this reason, we assume *the underlying theory T of a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ to have decidable constraint satisfiability problem.*

[3] For instance, the theory of dense linear orders is locally finite but cannot be the theory of a single finite structure, because finite linear orders are not dense.

We will see that this assumption alone is not sufficient to guarantee the decidability of the ground satisfiability problem for data-flow theories (cf. Section 2.2). Fortunately, the problem becomes decidable (cf. Sections 2.3 and 2.4) when the underlying theory T of a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ satisfies the same requirements for the correctness of the combination schema of Section 1.3.

2.2 Undecidability of the Satisfiability Problem

We show that the decidability of the (ground) satisfiability problem for (totally flexible) data-flow theories implies the decidability of the constraint satisfiability problem for unions of (signature disjoint) theories in a first-order framework. This reduction proves undecidability, as shown in Bonacina et al. (2006) (in fact, both recent and long standing literature – see, e.g., Nelson and Oppen, 1979; Tinelli and Harandi, 1996 – impose further requirements, such as stable infiniteness, on the component theories to obtain positive decidability transfer results of the constraint satisfiability problem).

Theorem 2.2.1. *There exists a totally flexible data-flow theory $\mathcal{T}$ whose ground satisfiability problem is undecidable.*

Proof. We must define a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ such that $\Sigma_r = \emptyset$, i.e. $\mathcal{T}$ is totally flexible, and the constraint satisfiability problem of T is decidable, according to Assumption 2.1.9.

To define a suitable T, the following two facts about combinations of theories are crucial:

(i) there exist theories T_1, T_2 whose constraint satisfiability problem is decidable, whose signatures Σ_1, Σ_2 are disjoint and such that the constraint satisfiability problem of $T_1 \cup T_2$ is undecidable (this is shown in Bonacina et al., 2006);

(ii) let T be a Σ-theory whose constraint satisfiability problem is decidable and Σ' be a signature such that $\Sigma' \supseteq \Sigma$. If we consider T as a Σ'-theory, then the constraint satisfiability problems of T is still decidable (this is proved in, e.g., Ganzinger, 2002; Tinelli and Zarba, 2005).

Consider now theories T_1, T_2 as in (i) above and let us define a new Σ-theory T as follows:

$$\Sigma := \Sigma_1 \cup \Sigma_2 \cup \{P\} \quad \text{and} \quad T := \{P \to \psi \mid \psi \in T_1\} \cup \{\neg P \to \psi \mid \psi \in T_2\},$$

where P is a fresh 0-ary predicate symbol (or, otherwise said, a fresh propositional letter). We claim that the constraint satisfiability problem for the Σ-theory T is decidable. In fact, given a $\Sigma_1 \cup \Sigma_2 \cup \{P\}$ constraint Γ, we first guess the truth value of P and add either P or $\neg P$ to Γ, accordingly. At this point, we are left with the problem of solving a constraint satisfiability problem of the $(\Sigma_1 \cup \Sigma_2 \cup \{P\})$-theory T_i for either $i = 1$ or $i = 2$. This is decidable by fact (ii) above: the constraint satisfiability problem of the Σ_i-theory T_i is decidable by assumption and the symbols from $\Sigma_j \cup \{P\}$ $(j \neq i)$ are free for T_i.

We now show that the ground satisfiability problem for $\mathcal{T}$ is undecidable by identifying a particular class of ground LTL$(\Sigma^{\underline{a},\underline{c}})$-sentences whose satisfiability cannot be decided. We assume that there are infinitely many system parameters (whereas the cardinality of the set of system variables is irrelevant). We claim that it is not possible to decide the $\mathcal{T}$-satisfiability of the following type of ground LTL$(\Sigma^{\underline{c}})$-sentences:

$$P \wedge \Gamma_1 \wedge X(\neg P \wedge \Gamma_2), \tag{2.3}$$

where Γ_i is a finite conjunction of $\Sigma_i^{\underline{c}}$-literals (for $i = 1, 2$) and the $\underline{c}$ are the free constants of the data-flow theory $\mathcal{T}$ (i.e. the rigid system parameters). In fact, if (2.3) is satisfiable (in the sense of Definition 2.1.5) then it is easy to build a model (in first-order semantics) for $T_1 \cup T_2$ satisfying $\Gamma_1 \cup \Gamma_2$, and also the converse holds. Thus the satisfiability of the sentences of the kind described in (2.3) is reduced to the satisfiability w.r.t. $T_1 \cup T_2$ of the arbitrary constraint $\Gamma_1 \cup \Gamma_2$: this is undecidable by fact (i) above (notice that the satisfiability of pure constraints, like $\Gamma_1 \cup \Gamma_2$, is equivalent to satisfiability of arbitrary $(\Sigma_1 \cup \Sigma_2)$-constraints, because every constraint is equisatisfiable with an effectively built pure constraint, see, e.g., Baader and Tinelli, 2002; Ghilardi, 2004). □

2.3 Decidability and Locally Finite Data-Flow Theories

Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a given data-flow theory. The arguments underlying the proof of Theorem 2.2.1 suggest that the undecidability of the ground satisfiability problem for $\mathcal{T}$ arises precisely for the same reasons leading to the undecidability of combined constraint satisfiability problems in the first-order framework. The hope is that the same (or similar) requirements yielding the decidability of the constraint satisfiability problem in unions of theories will also give the decidability of the ground satisfiability problem for $\mathcal{T}$. It turns out that this is indeed the case for both locally

finite and Noetherian theories (cf. Subsections 1.2.1 and 1.2.2 in Chapter 1).

Theorem 2.3.1. *The ground satisfiability problem for a locally finite compatible data-flow theory is decidable.*

Below, we give two constructive proofs of this theorem. The former is based on an eager reduction to the satisfiability problem for propositional LTL. The latter consists in a lazy integration between a standard tableau algorithm for the satisfiability problem of propositional LTL and a decision procedure for the constraint satisfiability problem in the background (first-order) theory T.

2.3.1 Propositional Linear Time Temporal Logic

Propositional $\mathcal{L}$-formulae (or PLTL-formulae or simply propositional formulae) are built up from a set of propositional letters $\mathcal{L}$ by using Boolean connectives and the temporal operators $X, \Box, \Diamond, U$. We use letters $\alpha, \beta, \ldots$ for propositional formulae. The semantics for PLTL is the standard one: we recall it for the sake of completeness. A *PLTL-Kripke model $V = \{V_n\}_n$ for $\mathcal{L}$* is a sequence of Boolean assignments

$$V_n : \mathcal{L} \longrightarrow \{0, 1\} \qquad (n \in \mathbb{N}).$$

Given such a Kripke model and a propositional formula α, the notion of α being true at instant $t \in \mathbb{N}$ in V is recursively defined as follows (this is parallel to Definition 2.1.2):

- if $p \in \mathcal{L}$, $V \models_t p$ iff $V_t(p) = 1$;
- $V \models_t \neg\alpha$ iff $V \not\models_t \alpha$;
- $V \models_t \alpha \wedge \beta$ iff $V \models_t \alpha$ and $V \models_t \beta$;
- $V \models_t \alpha \vee \beta$ iff $V \models_t \alpha$ or $V \models_t \beta$;
- $V \models_t X\alpha$ iff $V \models_{t+1} \alpha$;
- $V \models_t \Box\alpha$ iff for each $t' \geq t$, $V \models_{t'} \alpha$;
- $V \models_t \Diamond\alpha$ iff for some $t' \geq t$, $V \models_{t'} \alpha$;
- $V \models_t \alpha U \beta$ iff there exists $t' \geq t$ such that $V \models_{t'} \beta$ and for each t'', $t \leq t'' < t' \Rightarrow V \models_{t''} \alpha$.

We say that α is satisfied in V iff $V \models_0 \alpha$ (in general, if the subscript of $\models$ is omitted, it is intended to be equal to 0).

2.3.2 Eager Reduction to Propositional LTL-Satisfiability

In the rest of this section, let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a locally finite compatible data-flow theory. We prove Theorem 2.3.1 by a reduction to satisfiability in propositional linear temporal logic (PLTL, from now on). The syntactic relationship between first-order and propositional LTL-formulae is given by the notion of abstraction.

Definition 2.3.2 (PLTL-Abstraction)**.** Given a signature $\Sigma^{\underline{a}}$ and a set of propositional letters $\mathcal{L}$ (of the same cardinality as the set of ground $\Sigma^{\underline{a}}$–atoms), let $[\![\cdot]\!]$ be a bijection from the set of ground $\Sigma^{\underline{a}}$–atoms into $\mathcal{L}$. By translating identically Boolean and temporal connectives, the map is inductively extended to a bijective map (also called $[\![\cdot]\!]$) from the set of ground LTL($\Sigma^{\underline{a}}$)-sentences onto the set of propositional $\mathcal{L}$-formulae.

Given a ground LTL($\Sigma^{\underline{a}}$)-sentence φ, we call $[\![\varphi]\!]$ the *PLTL-abstraction* of φ. Given a set Θ of ground LTL($\Sigma^{\underline{a}}$)-sentences, $[\![\Theta]\!]$ denotes the set $\{[\![\varphi]\!] \mid \varphi \in \Theta\}$. The following straightforward lemma explains why PLTL-abstractions are relevant for satisfiability checking of LTL($\Sigma^{\underline{a}}$)-sentences.

Lemma 2.3.3. *Let $\mathcal{L}$ be a set of propositional letters, Σ be a signature, $\underline{a}$ be a set of free constants, and $[\![\cdot]\!]$ be a PLTL-abstraction function mapping ground LTL($\Sigma^{\underline{a}}$)-sentences into propositional $\mathcal{L}$-formulae. Suppose we are given a ground LTL($\Sigma^{\underline{a}}$)-sentence φ, a Kripke model V for $\mathcal{L}$ (based on $\mathbb{N}$ as a temporal flow) and an LTL($\Sigma^{\underline{a}}$)-structure $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$ such that for every $t \in \mathbb{N}$ and for every $\Sigma^{\underline{a}}$-ground atom ℓ occurring in φ we have*

$$\mathcal{M}_t \models \ell \qquad \textit{iff} \qquad V_t([\![\ell]\!]) = 1.$$

Then we have also

$$\mathcal{M} \models_t \varphi \qquad \textit{iff} \qquad V \models_t [\![\varphi]\!],$$

for every $t \in \mathbb{N}$.

Proof. The proof is by an easy induction on the complexity of the subformulae ψ

occurring in φ. The condition

$$\mathcal{M} \models_t \psi \qquad \text{iff} \qquad V \models_t [\![\psi]\!]$$

is obvious if ψ is an atom, and follows directly from the induction hypothesis if it is of the kind $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\neg\psi$.

We concentrate now on subformulae ψ of the kind $X\psi_1$, $\Box\psi_1$, and $\psi_1 U \psi_2$. Our induction hypothesis is that, for every $\Sigma^{\underline{a}}$–ground atom ℓ occurring in φ we have $\mathcal{M}_t \models \ell$ iff $V_t([\![\ell]\!]) = 1$ and, for each proper subformula ψ' of ψ, we have that $\mathcal{M} \models_t \psi'$ iff $V \models_t [\![\psi']\!]$. Suppose ψ is of the kind $X\psi_1$, we want to show that $\mathcal{M} \models_t X\psi_1$ iff $V \models_t [\![X\psi_1]\!]$. We first show that if $\mathcal{M} \models_t X\psi_1$ then $V \models_t [\![X\psi_1]\!]$. If $\mathcal{M} \models_t X\psi_1$ then $\mathcal{M} \models_{t+1} \psi_1$ thus, by induction hypothesis, $V \models_{t+1} [\![\psi_1]\!]$ and thus $V \models_t [\![X\psi_1]\!]$. The converse holds because, if $V \models_t [\![X\psi_1]\!]$, then $V \models_{t+1} [\![\psi_1]\!]$ and, again by induction hypothesis, $\mathcal{M} \models_{t+1} \psi_1$ and so $\mathcal{M} \models_t X\psi_1$.

Let us show that $\mathcal{M} \models_t \Box\psi_1$ iff $V \models_t [\![\Box\psi_1]\!]$. If $\mathcal{M} \models_t \Box\psi_1$ then $\mathcal{M} \models_i \psi_1$ for each $i \geq t$, thus by induction hypothesis, $V \models_i [\![\psi_1]\!]$ for each $i \geq t$, thus $V \models_t [\![\Box\psi_1]\!]$. Analogously, the converse holds.

Finally, let us show that $\mathcal{M} \models_t \psi_1 U \psi_2$ iff $V \models_t [\![\psi_1 U \psi_2]\!]$. If $\mathcal{M} \models_t \psi_1 U \psi_2$ there exists $k \geq t$ such that $\mathcal{M} \models_k \psi_2$ and such that, for every $t \leq i < k$, $\mathcal{M} \models_i \psi_1$. Applying the inductive hypothesis we obtain that $V \models_k [\![\psi_2]\!]$ and, for every $t \leq i < k$, $V \models_i [\![\psi_1]\!]$. Thus $V \models_t [\![\psi_1 U \psi_2]\!]$ obtains. The converse holds by an analogous argument. □

The key to define a reduction to the satisfiability problem in PLTL is guessing.

Definition 2.3.4 (S-Guessing)**.** Given a signature Σ and a finite set of Σ-atoms S, an *S-guessing* $\mathcal{G}$ is a Boolean assignment to members of S (we view $\mathcal{G}$ as the set $\{\varphi \mid \varphi \in S \text{ and } \mathcal{G}(\varphi) \text{ is assigned to true}\} \cup \{\neg\varphi \mid \varphi \in S \text{ and } \mathcal{G}(\varphi) \text{ is assigned to false}\}$).

Indeed, guessing must take into account rigid constants: each guessing of atoms over flexible symbols must be "compatible" with the guessing of atoms over rigid symbols. Formally, this is ensured as follows.

By definition of locally finite compatible data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$, there must exist a theory T_r such that $T_r \subseteq T$ is effectively locally finite. So, given a finite subset $\underline{c}_0$ of $\underline{c}$, it is possible to compute a finite set S of ground $\Sigma_r^{\underline{c}_0}$-atoms which are representative modulo T-equivalence of all ground $\Sigma_r^{\underline{c}_0}$-atoms: for this

choice of S, an S-guessing is called a *rigid $\underline{c}_0$-guessing*. Now, let $\hat{S}$ be any finite set of $\Sigma^{\underline{a},\underline{c}}$-atoms and $\mathcal{G}$ be a rigid $\underline{c}_0$-guessing: an $\hat{S}$-guessing $\hat{\mathcal{G}}$ is *$\mathcal{G}$-compatible* iff $\mathcal{G} \cup \hat{\mathcal{G}}$ is T-satisfiable. The set of $\mathcal{G}$-compatible $\hat{S}$-guessings is denoted by $C(\hat{S}, \mathcal{G})$.

Theorem 2.3.1 is an immediate consequence of the well-known fact that PLTL-satisfiability is decidable and the following proposition:

Proposition 2.3.5. *Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a locally finite compatible data-flow theory. Let $\mathcal{L}$ be a set of propositional letters and $[\![\cdot]\!]$ be a PLTL-abstraction function mapping ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentences into propositional $\mathcal{L}$-formulae. A ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ is satisfiable in an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate for $\mathcal{T}$ iff there exists a rigid $\underline{c}_0$-guessing $\mathcal{G}$ such that the propositional formula*

$$[\![\varphi]\!] \wedge \Box \bigwedge_{\psi \in \mathcal{G}} [\![\psi]\!] \wedge \Box \Big(\bigvee_{\hat{\mathcal{G}} \in C(At(\varphi), \mathcal{G})} \bigwedge_{\psi \in \hat{\mathcal{G}}} [\![\psi]\!] \Big) \tag{2.4}$$

is satisfiable in a PLTL-Kripke model (here $\underline{c}_0$ is the subset of the set $\underline{c}$ of system parameters occurring in φ and $At(\varphi)$ is the set of $\Sigma^{\underline{a},\underline{c}}$-atoms occurring in φ).

Proof. The 'only if' is immediate from Lemma 2.3.3. The converse can be derived from Lemma 1.3.2. Suppose that the PLTL-formula (2.4) is satisfiable in a Kripke model $V = \{V_n\}_{n \in \mathbb{N}}$ for a certain rigid $\underline{c}_0$-guessing $\mathcal{G}$. This means that for every n there is $\hat{\mathcal{G}}_n \in C(At(\varphi), \mathcal{G})$ such that $V \models_n \bigwedge_{\psi \in \mathcal{G}} [\![\psi]\!] \wedge \bigwedge_{\psi \in \hat{\mathcal{G}}_n} [\![\psi]\!]$. Since $\hat{\mathcal{G}}_n$ is $\mathcal{G}$-compatible, there is a $\Sigma^{\underline{a},\underline{c}_0}$-structure $\mathcal{M}_n$ which is a model of $T \cup \hat{\mathcal{G}}_n \cup \mathcal{G}$; by Lemma 1.3.2, the $\mathcal{M}_n$ can be $\Sigma^{\underline{a},\underline{c}_0}$-embedded into $\Sigma^{\underline{a},\underline{c}}$-structures $\mathcal{M}'_n$ such that $\mathcal{M}' := \{\mathcal{M}'_n\}_{n \in \mathbb{N}}$ is appropriate for $\mathcal{T}$.[4] The $\mathcal{M}_n$ can be seen as $\Sigma^{\underline{a},\underline{c}}$-structures by interpreting rigid parameters $\underline{c} \setminus \underline{c}_0$ arbitrarily (but in the same way in all $\mathcal{M}_n$). Since truth of ground literals is preserved through embeddings, $\mathcal{M}'_n$ is again a model of $\hat{\mathcal{G}}_n$ for every n. But then Lemma 2.3.3 ensures that $\mathcal{M}' \models \varphi$, given that $V \models [\![\varphi]\!]$. □

Example 2.3.6 (Plaisted, 1986)**.** Let $\mathcal{T} = \langle \{>\}, T_{lo}, \{>\}, \underline{a}, \underline{c} \rangle$ be a data-flow theory, where T_{lo} is the theory of strict linear orders and $>$ is a binary predicate symbol. Since T_{lo} (i) is universal, (ii) admits as a model completion the theory of dense linear order without endpoints and (iii) is effectively locally finite, then $\mathcal{T}$ is a locally finite compatible data-flow theory; moreover, it is easy to check that the constraint

[4] Lemma 1.3.2 is used with $I := \mathbb{N}$, and $T_i := T$, but symbols from $\Sigma \setminus \Sigma_r$ are disjointly renamed when building the signature Σ_i for the i^{th} copy of T (the same observation applies also to the flexible constants $\underline{a}$). In this way, a model of $\bigcup_i T_i$ is the same thing as a sequence of models $\{\mathcal{M}'_n\}_{n \in \mathbb{N}}$ of T whose Σ_r-reducts coincide.

satisfiability problem for T_{lo} is decidable. We are interested to check the satisfiability of the following LTL($\Sigma^{\underline{a}}$)-sentence:[5]

$$\varphi \quad :\equiv \quad a > b \wedge b > c \wedge (\Diamond a = c \vee \Diamond c > a)$$

Indeed, the solution to this satisfiability problem depends on how we classify the symbols a, b, and c. Notice that the set $At(\varphi)$ of atoms in φ is $\{a > b, b > c, a = c, c > a\}$. Now, let us consider two cases according to how a, b, c are considered as flexible or rigid.

1. $\underline{a} = \{b\}$ and $\underline{c} = \{a, c\}$. The set of representative $\Sigma^{\underline{c}}$-atoms is $\{a > c, a = c, c > a\}$. The rigid $\underline{c}$-guessings which are consistent w.r.t. T_{lo} are therefore the following:

$$\begin{aligned}\mathcal{G}_1 :=& \{a > c, a \neq c, c \not> a\},\\ \mathcal{G}_2 :=& \{a \not> c, a = c, c \not> a\},\\ \mathcal{G}_3 :=& \{a \not> c, a \neq c, c > a\}.\end{aligned}$$

   We omitted to consider the rigid $\underline{c}$-guessings which are not T_{lo}-satisfiable because every T_{lo}-unsatisfiable $\underline{c}$-guessing $\mathcal{G}$ leads to the inconsistency of the formula (2.4) since there is no $\mathcal{G}$-compatible $At(\varphi)$-guessing. Consider now the first two conjuncts of (2.4) for each $\mathcal{G}_i$:

   $\mathcal{G}_1$**:** from

$$\begin{aligned}&[\![a > b]\!] \wedge [\![b > c]\!] \wedge (\Diamond [\![a = c]\!] \vee \Diamond [\![c > a]\!]) \wedge\\ &\wedge \Box([\![a > c]\!] \wedge \neg [\![a = c]\!] \wedge \neg [\![c > a]\!])\end{aligned}$$

   we obtain

$$\begin{aligned}&([\![a > b]\!] \wedge [\![b > c]\!] \wedge \underline{\Diamond [\![a = c]\!]} \wedge \Box [\![a > c]\!] \wedge \underline{\Box \neg [\![a = c]\!]} \wedge \Box \neg [\![c > a]\!]) \vee\\ &\vee([\![a > b]\!] \wedge [\![b > c]\!] \wedge \underline{\Diamond [\![c > a]\!]} \wedge \Box [\![a > c]\!] \wedge \Box \neg [\![a = c]\!] \wedge \underline{\Box \neg [\![c > a]\!]})\end{aligned}$$

   Each disjunct is easily found PLTL-unsatisfiable because of the inconsistency between the underlined part of the formula.

[5] The formula is obtained by negating $a > b \wedge b > c \rightarrow \Box(a > c)$

$\mathcal{G}_2$: from

$$[\![a > b]\!] \wedge [\![b > c]\!] \wedge (\Diamond[\![a = c]\!] \vee \Diamond[\![c > a]\!]) \wedge \\ \wedge \Box(\neg[\![a > c]\!] \wedge [\![a = c]\!] \wedge \neg[\![c > a]\!])$$

we obtain

$$([\![a > b]\!] \wedge [\![b > c]\!] \wedge \Diamond[\![a = c]\!] \wedge \Box\neg[\![a > c]\!] \wedge \Box[\![a = c]\!] \wedge \Box\neg[\![c > a]\!]) \vee \\ \vee([\![a > b]\!] \wedge [\![b > c]\!] \wedge \underline{\Diamond[\![c > a]\!]} \wedge \Box\neg[\![a > c]\!] \wedge \Box[\![a = c]\!] \wedge \underline{\Box\neg[\![c > a]\!]})$$

The second disjunct is easily found PLTL-unsatisfiable because of the inconsistency between the underlined part of the formula. We are left to check the PLTL-unsatisfiable of the following formula obtained by considering all $\mathcal{G}_2$-compatible guessings:

$$\underline{[\![a > b]\!]} \wedge \underline{\underline{[\![b > c]\!]}} \wedge \Diamond[\![a = c]\!] \wedge \Box\neg[\![a > c]\!] \wedge \Box[\![a = c]\!] \wedge \Box\neg[\![c > a]\!] \wedge \\ \wedge \Box \left(\begin{array}{lll} & (\underline{\neg[\![a > b]\!]} \wedge [\![b > c]\!] \wedge [\![a = c]\!] \wedge \neg[\![c > a]\!]) & \vee \\ \vee & ([\![a > b]\!] \wedge \underline{\underline{\neg[\![b > c]\!]}} \wedge [\![a = c]\!] \wedge \neg[\![c > a]\!]) & \vee \\ \vee & (\underline{\neg[\![a > b]\!]} \wedge \underline{\underline{\neg[\![b > c]\!]}} \wedge [\![a = c]\!] \wedge \neg[\![c > a]\!]) & \end{array} \right)$$

which is easily found PLTL-inconsistent by observing the underlined literals.

$\mathcal{G}_3$: from

$$[\![a > b]\!] \wedge [\![b > c]\!] \wedge (\Diamond[\![a = c]\!] \vee \Diamond[\![c > a]\!]) \wedge \\ \wedge \Box(\neg[\![a > c]\!] \wedge \neg[\![a = c]\!] \wedge [\![c > a]\!])$$

we obtain

$$([\![a > b]\!] \wedge [\![b > c]\!] \wedge \underline{\Diamond[\![a = c]\!]} \wedge \Box\neg[\![a > c]\!] \wedge \underline{\Box\neg[\![a = c]\!]} \wedge \Box[\![c > a]\!]) \vee \\ \vee([\![a > b]\!] \wedge [\![b > c]\!] \wedge \Diamond[\![c > a]\!] \wedge \Box\neg[\![a > c]\!] \wedge \Box\neg[\![a = c]\!] \wedge \Box[\![c > a]\!])$$

The first disjunct is easily found PLTL-unsatisfiable because of the inconsistency between the underlined part of the formula. We are left to check the PLTL-unsatisfiable of the following formula obtained by considering

all the $\mathcal{G}_3$-compatible guessings:

$$\underline{[\![a > b]\!]} \wedge \underline{\underline{[\![b > c]\!]}} \wedge \Diamond [\![c > a]\!] \wedge \Box\neg [\![a > c]\!] \wedge \Box\neg [\![a = c]\!] \wedge \Box [\![c > a]\!] \wedge$$
$$\wedge\Box \left(\begin{array}{lll} & (\underline{\neg [\![a > b]\!]} \wedge [\![b > c]\!] \wedge \neg [\![a = c]\!] \wedge [\![c > a]\!]) & \vee \\ \vee & ([\![a > b]\!] \wedge \underline{\underline{\neg [\![b > c]\!]}} \wedge \neg [\![a = c]\!] \wedge [\![c > a]\!]) & \vee \\ \vee & (\underline{\neg [\![a > b]\!]} \wedge \underline{\underline{\neg [\![b > c]\!]}} \wedge \neg [\![a = c]\!] \wedge [\![c > a]\!]) & \end{array} \right)$$

which is easily found PLTL-inconsistent by observing the underlined literals.

Since there is no rigid guessing such that the formula (2.4) is PLTL-satisfiable, we are entitled to conclude that φ is unsatisfiable in any LTL($\Sigma^{\{b\},\{a,c\}}$)-structure appropriate for $\mathcal{T}$.

2. $\underline{a} = \{a, b, c\}$ and $\underline{c} = \emptyset$. Since there are no system parameters, all the $At(\varphi)$-guessings which are T_{lo}-satisfiable are trivially compatible with every rigid $\underline{c}$-guessing. It easy to check that the corresponding instance of (2.4) is PLTL-satisfiable. Hence, by Theorem 2.3.1, we conclude that φ is satisfiable in an LTL($\Sigma^{\{a,b,c\},\emptyset}$)-structure appropriate for $\mathcal{T}$.

Proposition 2.3.5 gives an algorithm to solve the ground satisfiability problem for $\mathcal{T}$, when $\mathcal{T}$ is a locally finite compatible data-flow theory. For the PLTL-satisfiability test, one may use any decision procedure for PLTL-satisfiability based on tableaux, automata, or temporal extensions of resolution. Such an algorithm can be regarded as an *eager reduction algorithm*, in the sense that it produces an instance of a PLTL-satisfiability problem. The drawback is that the resulting PLTL-satisfiability problem may be quite large. The main advantage is that both decision procedures for the constraint satisfiability problem of the underlying locally finite theory and decision procedures for PLTL can be used 'off-the-shelf'. In the following, we consider a procedure which is likely to scale up more smoothly at the price of a finer grain integration between the constraint reasoner in the background theory and the PLTL satisfiability solver.

2.3.3 A Lazy Tableau Procedure

We describe a *lazy* approach to solve the ground satisfiability problem for data-flow theories by extending the classic approach to temporal propositional satisfiability

adopted in the Tableaux community. The key idea is to lift the definition of Hintikka sets to ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentences of the form (2.4) of Proposition 2.3.5. The soundness and completeness proof of the resulting algorithm (cf. Corollary 2.3.12 below) is immediate from Proposition 2.3.5 and basic properties of tableaux for PLTL (see, e.g., Manna and Pnueli, 1995, Section 5.5).

As before, let us fix a locally finite compatible data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$. A ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence is in *Negation Normal Form* (NNF) iff it is built up from LTL($\Sigma^{\underline{a},\underline{c}}$)-literals by using $\vee, \wedge, X, \Box, U$. It can be shown that every ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence is logically equivalent to a formula in NNF.[6]

Definition 2.3.7. Given a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ in NNF, the *closure* of φ is the set $cl(\varphi)$ containing:

(i) all subformulae of φ and all negations of atoms occurring in φ;

(ii) a representative set (modulo T-equivalence) of $\Sigma_r^{\underline{c}_0}$-literals, where $\underline{c}_0$ is the finite set of system parameters occurring in φ;

(iii) formulae of the form $X(\psi U \chi)$, where $\psi U \chi$ is a subformula of φ;

(iv) formulae of the form $X\Box\psi$, where $\Box\psi$ is a subformula of φ.

Notice that $cl(\varphi)$ is finite and has cardinality $O(\max(n, k(\underline{c}_0)))$, if n is the length of φ and $k(\underline{c}_0)$ is the cardinality of a representative set of $\Sigma_r^{\underline{c}_0}$-literals.

Definition 2.3.8. Given a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ in NNF, a *Hintikka set* for φ is a subset $H \subseteq cl(\varphi)$ such that:

(i) for every atom $\psi \in cl(\varphi)$, H contains either ψ or $\neg\psi$;

(ii) the set of literals from H is T-satisfiable;

(iii) if $\psi_1 \wedge \psi_2 \in H$, then ($\psi_1 \in H$ and $\psi_2 \in H$);

(iv) if $\psi_1 \vee \psi_2 \in H$, then ($\psi_1 \in H$ or $\psi_2 \in H$);

(v) if $\psi_1 U \psi_2 \in H$, then ($\psi_2 \in H$ or ($\psi_1 \in H$ and $X(\psi_1 U \psi_2) \in H$));

(vi) if $\Box\psi \in H$, then ($\psi \in H$ and $X\Box\psi \in H$).

[6] For simplicity (and ignoring efficiency problems), we include $\Box$ but not the 'release operator' $\varphi R \psi := \neg(\neg\varphi U \neg\psi)$ (this operator can be defined in terms of $\Box$ and U as $\Box\psi \vee (\psi U(\varphi \wedge \psi))$).

We are now in the position to define a Hintikka graph, which will be used as the key data structure to define the tableau procedure.

Definition 2.3.9. The *Hintikka graph* $\mathcal{H}(\varphi)$ of φ is the directed graph having as nodes the Hintikka sets for φ and as edges the pairs $H \rightarrow H'$ such that

(i) $H' \supseteq \{\psi \mid X\psi \in H\}$;

(ii) H and H' contain the same ground $\Sigma_r^{\underline{c}_0}$-literals.

Definition 2.3.10. A *strongly connected subgraph* (scs) of $\mathcal{H}(\varphi)$ is a set $\mathcal{C}$ of nodes of $\mathcal{H}(\varphi)$ such that for every $H, H' \in \mathcal{C}$ there is a (non-empty) $\mathcal{H}(\varphi)$-path from H to H' whose nodes all belong to $\mathcal{C}$.[7]

Definition 2.3.11 (Manna and Pnueli, 1995)**.** An scs $\mathcal{C}$ is *fulfilling* iff for every $\psi_1 U \psi_2 \in cl(\varphi)$ there is $H \in \mathcal{C}$ such that either $\psi_1 U \psi_2 \notin H$ or $\psi_2 \in H$.

A node H in $\mathcal{H}(\varphi)$ is *initial* iff $\varphi \in H$.

Corollary 2.3.12. *A ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ in NNF is satisfiable in an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate for $\mathcal{T}$ iff there is a $\mathcal{H}(\varphi)$-path leading from an initial node into a fulfilling scs.*

An observation about the complexity of the lazy procedure is in order. When the set of representative $\Sigma_r^{\underline{c}_0}$-atoms has polynomial size, one can derive a PSPACE-decision procedure (provided that the T-constraint satisfiability problem is also PSPACE) from the above Corollary. The crucial point is to avoid the explicit construction of the Hintikka graph and explore it 'on-the-fly' by using well-known techniques of the PLTL literature (see, e.g., Sistla and Clarke, 1985).

2.4 Decidability and Noetherian Compatible Data-Flow Theories

Below, we focus on the ground satisfiability problem in the Noetherian compatible case. Before developing our decision procedure, a preliminary notion is required.

Definition 2.4.1 (φ-Guessing)**.** Let φ be a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence. A *$\varphi$-guessing* is a Boolean assignment to literals of φ (we view a guessing as the set $\{\ell \mid \ell$ is an atom occurring in φ and ℓ is assigned to true$\} \cup \{\neg\ell \mid \ell$ is an atom occurring in φ and ℓ is assigned to false$\}$).

[7] In particular, for $H = H'$, we see that an scs cannot consist of a single not self-accessible node.

Algorithm 2 The satisfiability procedure for the Noetherian compatible case

Require: φ ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence

```
1:  procedure NSat(φ)
2:      for all φ-compatible sets of φ-guessings G_(φ) do
3:          B ← ∅
4:          repeat
5:              B' ← B
6:              for all G_i ∈ G_(φ) do
7:                  B_i ← Res_T^c(G_i ∪ B)
8:              end for
9:              B ← ⋃_i B_i
10:         until Dp-t(B' ∧ ¬B) = "unsatisfiable"
11:         if Dp-t(B) = "satisfiable" then
12:             return "satisfiable"
13:         end if
14:     end for
15:     return "unsatisfiable"
16: end procedure
```

We say that a (non-empty) set of φ-guessings $\mathcal{G}_{(\varphi)} := \{G_1, \ldots, G_k\}$ is *φ-compatible* if and only if $[\![\varphi \wedge \Box \bigvee_{i=1}^{k} G_i]\!]$ is PLTL-satisfiable.

Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a Noetherian compatible data-flow theory. The procedure NSat (cf. Algorithm 2) takes a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ as input and returns *"satisfiable"* if there is an appropriate LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ for $\mathcal{T}$ such that $\mathcal{M} \models \varphi$; otherwise, it returns *"unsatisfiable"*. The procedure relies on a decision procedure for the PLTL-satisfiability problem in order to recognize the φ-compatible sets of φ-guessings (cf. the outer loop of NSat). Moreover, Dp-t is a decision procedure for the satisfiability problem of arbitrary Boolean combinations of atoms of the theory T (i.e., it is capable of checking the T-satisfiability of sets of ground $\Sigma^{\underline{a},\underline{c}}$-clauses and not only of ground $\Sigma^{\underline{a},\underline{c}}$-literals). Notice that Dp-t can be implemented by Satisfiability Modulo Theories solvers (see, e.g., Nieuwenhuis et al., 2006 or the survey in Ranise and Tinelli, 2006). Finally, $Res_T^{\underline{c}}$ is the T-residue enumerator for T_r w.r.t. $\underline{c}$.

In the outer loop of NSat, all the possible φ-compatible sets of φ-guessings are enumerated. Let $\mathcal{G}_{(\varphi)} := \{G_1, \ldots, G_n\}$ be the current set of φ-guessings. The local variable $\mathcal{B}$ is initialized to the empty set (line 3) and then updated in the inner loop (lines 4-10) as follows: the T_r-bases $\mathcal{B}_i$ for $G_i \cup \mathcal{B}$ w.r.t. $\underline{c}$ are computed (for $i = 1, \ldots, n$), and the new value of $\mathcal{B}$ is set to $\bigcup_i \mathcal{B}_i$ (line 5 saves in $\mathcal{B}'$ the old value of

$\mathcal{B}$). The inner loop is iterated until $\mathcal{B}$ is logically equivalent to $\mathcal{B}'$ modulo T. At this point, if $\mathcal{B}$ is T-consistent, the procedure stops and returns *"satisfiable"*; otherwise it tries another φ-compatible set of φ-guessings. If for all φ-compatible sets of φ-guessings the $\mathcal{B}$'s returned after the execution of the inner loop are T-inconsistent, the procedure returns *"unsatisfiable"*.

Indeed, the termination of NSat is a consequence of the Noetherianity of the underlying theory of $\mathcal{T}$ by using the fact that every infinite ascending chain of sets of positive ground Σ_r^c-clauses is eventually constant for logical consequence; formally, is stated by the following

Lemma 2.4.2. *The procedure* NSat *always terminates.*

Proof. Since the number of literals occurring in φ is finite, there is only a finite number of φ-guessings, and thus there is a finite number of sets of φ-guessings $\mathcal{G}_{(\varphi)}$. So, it remains to prove that the inner loop of lines 4-10 of Algorithm 2 terminates; to this aim we recall the fact (proved in Lemma 1.2.7) that every infinite ascending chain of sets of positive ground Σ_r^c-clauses is eventually constant for logical consequence w.r.t. a Noetherian theory T_r. The test on line 10 eventually have to succeed by the following reason: if we let $\mathcal{B}^0, \mathcal{B}^1, \mathcal{B}^2, \ldots$ be the values of the local variable $\mathcal{B}$ after each execution of the loop, we have that $T_r \cup \mathcal{B}^{i+1} \models \mathcal{B}^i$, for each i, by Definition 1.2.4(ii). Thus, if we let $\mathcal{D}_i := \bigcup_{j \leq i} \mathcal{B}_j$, then the sequence

$$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \ldots$$

is increasing and hence eventually constant modulo $T_r \subseteq T$, which means that also the above mentioned test eventually succeeds. □

The following two lemmas state the correctness of the procedure NSat.

Lemma 2.4.3 (Soundness)**.** *Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a Noetherian compatible data-flow theory and φ be a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence. If* NSat*($\varphi$) returns "satisfiable", then there is an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate for $\mathcal{T}$ such that $\mathcal{M} \models \varphi$.*

Proof. If NSat(φ) returns *"satisfiable"*, then there exists a (non-empty) set of φ-guessings $\mathcal{G}_{(\varphi)} := \{G_1, \ldots, G_n\}$ that are φ-compatible, i.e. such that $[\![\varphi \wedge \Box(\bigvee_i G_i)]\!]$ is satisfiable (as usual, with a little abuse of notation, we confuse the set G_i with the conjunction of the literals occurring in it). NSat(φ) will produce the list of sets

of positive ground $\Sigma^{\underline{c}}_r$-clauses

$$\mathcal{B}^0_1, \ldots, \mathcal{B}^0_n, \mathcal{B}^1_1, \ldots, \mathcal{B}^1_n, \ldots, \mathcal{B}^h_1, \ldots, \mathcal{B}^h_n,$$

such that:

- $\mathcal{B}^0, \ldots, \mathcal{B}^h, \mathcal{B}^{h+1}$ are the values of the local variable $\mathcal{B}$ in the iterations of the inner loop (we have $\mathcal{B}^0 = \emptyset, \mathcal{B}^1 = \bigcup_i \mathcal{B}^0_i, \ldots, \mathcal{B}^{h+1} = \bigcup_i \mathcal{B}^h_i$);
- for $j = 0, \ldots, h$ and for $i = 1, \ldots, n$, the set $\mathcal{B}^j_i$ is a T_r-basis for $G_i \cup \mathcal{B}^j$ w.r.t. $\underline{c}$,
- $\mathcal{B}^{h+1}$ is T-consistent and logically equivalent to $\mathcal{B}^h$ modulo T.

Let $\mathcal{B}^\star := \{C \mid T \cup \mathcal{B}^h \models C$ and C is a positive ground $\Sigma^{\underline{c}}_r$-clause$\}$; notice that $\mathcal{B}^\star$ does not contain the empty clause, moreover we claim that for every positive ground $\Sigma^{\underline{c}}_r$-clause C and for each $i \in \{1, \ldots, n\}$, we have

$$T \cup G_i \cup \mathcal{B}^\star \models C \quad \Rightarrow \quad C \in \mathcal{B}^\star. \tag{2.5}$$

In fact, if $T \cup G_i \cup \mathcal{B}^\star \models C$, then $T \cup G_i \cup \mathcal{B}^h \models C$ and so, by Definition 1.2.4(ii) $T_r \cup \mathcal{B}^h_i \models C$; but then $T_r \cup \mathcal{B}^{h+1} \models C$, meaning that $T \cup \mathcal{B}^h \models C$ (because $\mathcal{B}^{h+1}$ is logically equivalent to $\mathcal{B}^h$) and finally $C \in \mathcal{B}^\star$ by the definition of the latter. Let $V := V_0 \to V_1 \to \cdots \to V_n \to \ldots$ be the infinite sequence of Boolean assignments that is a PLTL model for $[\![\varphi \wedge \Box(\bigvee_i G_i)]\!]$. Let us consider the infinite sequence $\{G'_n\}_{n \in \mathbb{N}}$ of guessings such that $G'_n := G_i$ and $V \models_n [\![G_i]\!]$ (this is well-set since for every $n \geq 0$ there exists only one G_i such that $V \models_n [\![G_i]\!]$). By (2.5) and by Lemma 1.3.9,[8] we obtain an infinite sequence $\mathcal{M}_0, \ldots, \mathcal{M}_i, \ldots$ of $\Sigma^{\underline{a},\underline{c}}$-structures such that (i) they all have the same support M and $\mathcal{M}_{i|\Sigma^{\underline{c}}_r} = \mathcal{M}_{j|\Sigma^{\underline{c}}_r}$ $(i, j \in \mathbb{N})$; (ii) $\mathcal{M}_i \models T \cup G'_i$. These $\mathcal{M}_i$ consequently form an LTL$(\Sigma^{\underline{a},\underline{c}})$-structure $\mathcal{M} := \{\mathcal{M}_i\}_{i \in \mathbb{N}}$ that, by construction, for every atom ℓ occurring in φ satisfies the condition: $\mathcal{M} \models_i \ell$ iff $V \models_i [\![\ell]\!]$. Applying Lemma 2.3.3 we have that $\mathcal{M} \models_0 \varphi$, because $V \models_0 [\![\varphi]\!]$, thus $\mathcal{M} \models \varphi$ obtains. □

[8] Lemma 1.3.9 is used with $I := \mathbb{N}$, and $T_i := T$, but symbols from $\Sigma \setminus \Sigma_r$ are disjointly renamed when building the signature Σ_i for the i^{th} copy of T (the same observation applies also to the flexible constants $\underline{a}$). In this way, a model of $\bigcup_i T_i$ is the same thing as a sequence of models $\{\mathcal{M}'_n\}_{n \in \mathbb{N}}$ of T whose $\Sigma^{\underline{c}}_r$-reducts coincide.

Lemma 2.4.4 (Completeness)**.** *Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a Noetherian compatible data-flow theory and φ be a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence. If there is an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure $\mathcal{M}$ appropriate for $\mathcal{T}$ such that $\mathcal{M} \models \varphi$, then* NSAT*($\varphi$) returns "satisfiable".*

Proof. Let $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n \in \mathbb{N}}$ be an LTL($\Sigma^{\underline{a},\underline{c}}$)-structure appropriate for $\mathcal{T}$ such that $\mathcal{M} \models \varphi$. Let us consider the set of φ-guessings $\mathcal{G}_{(\varphi)} := \{G_1, \dots G_k\}$ defined as follows: $G_i \in \mathcal{G}_{(\varphi)}$ iff there exists an n such that $\mathcal{M} \models_n G_i$. It is easy to verify that $\mathcal{G}_{(\varphi)}$ is φ-compatible, i.e. that $[\![\varphi \wedge \Box(\bigvee_i G_i)]\!]$ is satisfiable (here $G_i \in \mathcal{G}_{(\varphi)}$). In fact, the PLTL structure V that satisfies the condition $V \models_n [\![\ell]\!]$ iff $\mathcal{M} \models_n \ell$ for every atom ℓ occurring in φ is a model for $[\![\varphi \wedge \Box(\bigvee_i G_i)]\!]$ by Lemma 2.3.3.

When examining the set of φ-guessings $\mathcal{G}_{(\varphi)}$, the procedure DP-LTL produces a (finite, by Lemma 2.4.2) list of sets of positive ground $\Sigma^{\underline{c}}_r$-clauses

$$\mathcal{B}^0_1, \dots, \mathcal{B}^0_k, \mathcal{B}^1_1, \dots, \mathcal{B}^1_k, \dots, \mathcal{B}^h_1, \dots, \mathcal{B}^h_k,$$

such that:

- $\mathcal{B}^0, \dots, \mathcal{B}^h, \mathcal{B}^{h+1}$ are the values of the local variable $\mathcal{B}$ in the iterations of the inner loop (we have $\mathcal{B}^0 = \emptyset, \mathcal{B}^1 = \bigcup_i \mathcal{B}^0_i, \dots, \mathcal{B}^{h+1} = \bigcup_i \mathcal{B}^h_i$);
- for $j = 0, \dots, h$ and for $i = 1, \dots, k$, the set $\mathcal{B}^j_i$ is a T_r-basis for $G_i \cup \mathcal{B}^j$ w.r.t. $\underline{c}$;
- $\mathcal{B}^{h+1}$ is logically equivalent to $\mathcal{B}^h$ modulo T.

We need to show that $\mathcal{B}^h$ is T-consistent. To this aim it is sufficient to observe (by induction on $j \leq h$) that the a $\Sigma^{\underline{c}}_r$-clause belonging to $\mathcal{B}^j$ is true in $\mathcal{M}_0$ (in fact in all the $\mathcal{M}_n$, because the symbols of $\Sigma^{\underline{c}}_r$ are rigidly interpreted): this is obvious for $j = 0$ and for $j > 0$ it is a direct consequence of the fact that every G_i is true in some $\mathcal{M}_n$, by induction hypothesis and Definition 1.2.4(i). □

The lemmas above yield our main decidability result.

Theorem 2.4.5. *The ground satisfiability problem for Noetherian compatible data-flow theories is decidable.*

The theories considered in the previous section (especially, those in Section 1.5) satisfy the hypothesis of the theorem above.

2.5 Extensions to Abstract Temporal Logics

By considering the proof of the correctness of NSAT, it becomes evident that only very few of the characteristic properties of LTL are used. It turns out that a simple generalization of NSAT can be used to decide satisfiability problems of "temporalized" extensions of Noetherian theories whose flow of time is not linear, for example branching as in CTL.

In order to formalize the observation above, we regard modal/temporal operators as functions operating on powerset Boolean algebras. In this way, logics for various flows of time, such as CTL, Propositional Dynamic Logic (PDL), and the μ-calculus fall within the scope of our result (see Baader et al., 2002 for a similar approach).

Definition 2.5.1. An *abstract temporal signature*[9] I is a purely functional signature extending the signature BA of Boolean algebras.[10] An *abstract temporal logic* L is a class of I-structures, whose Boolean reducts are powerset Boolean algebras. Given an I-term t, deciding whether $t \neq 0$ is satisfied in some member of L is the *satisfiability problem* for L. Given I-terms t, u, deciding whether $u = 1 \ \& \ t \neq 0$ is satisfied in some member of L is the *relativized satisfiability problem* for L.

In many cases (such as the one of LTL, CTL, PDL, and μ-calculus), it is possible to reduce the relativized satisfiability problem to that of satisfiability (by using the so-called "master modality"); however, there are logics for which the latter is decidable whereas the former is undecidable (see Gabbay et al., 2003).

Definition 2.5.2 ($I(\Sigma^{\underline{a}})$-sentence)**.** Given a signature Σ, a (finite or infinite) set of free constants $\underline{a}$, and an abstract temporal signature I, the set of $I(\Sigma^{\underline{a}})$-sentences is inductively defined as follows: (a) if φ is a first-order $\Sigma^{\underline{a}}$-sentence, then φ is an $I(\Sigma^{\underline{a}})$-sentence, (b) if φ_1, φ_2 are $I(\Sigma^{\underline{a}})$-sentences, so are $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \neg\varphi_1$, and (c) if $\psi_1, \ldots, \psi_n$ are $I(\Sigma^{\underline{a}})$-sentences and $O \in I \setminus BA$ has arity n, then $O(\psi_1, \ldots, \psi_n)$ is a $I(\Sigma^{\underline{a}})$-sentence.

When I is LTL, $I(\Sigma^{\underline{a},\underline{c}})$-sentences coincide with LTL$(\Sigma^{\underline{a},\underline{c}})$-sentences (cf. Definition 2.1.1). We defined an abstract temporal logic L (based on I) as a class

[9] From the modal/temporal literature viewpoint, the adjective "intensional" might be preferable to "abstract temporal". We have chosen the latter, in order to emphasize that our results are deemed as significant for a class of logics whose modalities concern flows of time.

[10] This signature contains two binary function symbols for meet and join, a unary function symbol for complement, and two constants for zero and one (the latter are denoted with 0 and 1, respectively).

of I-structures based on powerset Boolean algebras: such structures (also called I-frames) will be denoted with $\mathcal{F} = (\wp(F), \{O^{\mathcal{F}}\}_{O \in I \setminus BA})$.

Definition 2.5.3. Let a signature Σ, a set $\underline{a}$ of free constants, and an abstract temporal signature I be given; an $I(\Sigma^{\underline{a}})$-*structure* (or simply a *structure*) is a pair formed by an I-frame $\mathcal{F} = (\wp(F), \{O^{\mathcal{F}}\}_{O \in I \setminus BA})$ and a collection $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n \in F}$ of $\Sigma^{\underline{a}}$-structures (all based on the same domain).

An $I(\Sigma^{\underline{a}})$-sentence φ is true in the $I(\Sigma^{\underline{a}})$-structure $(\mathcal{F}, \mathcal{M})$ at $t \in F$ (noted $\mathcal{F}, \mathcal{M} \models_t \varphi$) iff the following holds: (a) if φ is a first-order sentence, then $\mathcal{F}, \mathcal{M} \models_t \varphi$ holds iff $\mathcal{M}_t \models \varphi$ and (b) if the main operator of φ is a Boolean connective, truth of φ is defined in a truth-table manner; (c) if φ is of the kind $O(\psi_1, \ldots, \psi_n)$, then $\mathcal{F}, \mathcal{M} \models_t \varphi$ holds iff $t \in O^{\mathcal{F}}(\{u \mid \mathcal{F}, \mathcal{M} \models_u \psi_1\}, \ldots, \{u \mid \mathcal{F}, \mathcal{M} \models_u \psi_n\})$.

If a data-flow theory $\mathcal{T}$ is given, we say that an $I(\Sigma^{\underline{a}})$-structure is appropriate for $\mathcal{T}$ iff it satisfies the requirements of Definition 2.1.5. The (ground) satisfiability problem for an abstract temporal logic L (based on I) and for a data-flow theory $\mathcal{T}$ is now the following: given a (ground) $I(\Sigma^{\underline{a}})$-sentence φ, decide whether there is a $I(\Sigma^{\underline{a}})$-structure $(\mathcal{F}, \mathcal{M})$ appropriate for $\mathcal{T}$, such that $\mathcal{F} \in L$ and such that $\mathcal{F}, \mathcal{M} \models_t \varphi$ holds for some t.

Theorem 2.5.4. *The ground satisfiability problem for $\mathcal{T}$ and L is decidable if (i) $\mathcal{T}$ is Noetherian compatible and (ii) the relativized satisfiability problem for L is decidable.*

When I is LTL, this result simplifies to Theorem 2.4.5. To prove Theorem 2.5.4, it is possible to re-use NSAT (cf. Algorithm 2) almost 'off-the-shelf', by preliminarily adapting the definition of PLTL-abstraction function $[\![\cdot]\!]$ (cf. Definition 2.3.2) to L in the following (obvious) way.

Definition 2.5.5 (I-Abstraction)**.** Given a signature $\Sigma^{\underline{a}}$ and an abstract temporal signature I containing a set of constants $\mathcal{K}$ (of the same cardinality as the set of ground $\Sigma^{\underline{a}}$-atoms), let $[\![\cdot]\!]$ be a bijection from the set of ground $\Sigma^{\underline{a}}$-atoms into $\mathcal{K}$. By translating Boolean and temporal operators into the appropriate functions of I, the map is inductively extended to a bijective map (also called $[\![\cdot]\!]$) from the set of ground $I(\Sigma^{\underline{a}})$-sentences onto the set of I-terms.

It turns out that only the compatibility of guessings should be changed: a finite set of φ-guessings $\mathcal{G}_{(\varphi)} := \{G_1, \ldots, G_k\}$ is φ-compatible if and only if the relativized

satisfiability problem

$$[\![\,\varphi\,]\!] \neq 0 \quad \& \quad [\![\,\bigvee_{i=1}^{k} G_i\,]\!] = 1$$

is satisfiable in L.

The following lemmas prove Theorem 2.5.4.

Lemma 2.5.6 (Soundness). *Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c}\rangle$ be a Noetherian compatible data-flow theory, L be an abstract temporal logic and φ be a ground $I(\Sigma^{\underline{a},\underline{c}})$-sentence. If* NSAT*$(\varphi)$ returns "satisfiable", then there is an $I(\Sigma^{\underline{a},\underline{c}})$-structure $(\mathcal{F}, \mathcal{M})$ appropriate for $\mathcal{T}$ such that $\mathcal{F} \in L$ and $\mathcal{F}, \mathcal{M} \models_t \varphi$ holds for some t.*

Proof. If NSAT(φ) returns *"satisfiable"*, then there is a (non-empty) set of φ-guessings $\mathcal{G}_{(\varphi)} := \{G_1, \ldots, G_n\}$ that are φ-compatible, i.e. such that the relativized satisfiability problem $[\![\,\varphi\,]\!] \neq 0$ & $[\![\,\bigvee_{i=1}^{k} G_i\,]\!] = 1$ is satisfiable in the abstract temporal logic L. The proof runs as in the case of Lemma 2.4.3, with the exception that we consider (instead of a PLTL model) an I-frame $\mathcal{F} = (\wp(F), \{O^{\mathcal{F}}\}_{O \in I \setminus BA})$ for L that satisfies $[\![\,\varphi\,]\!] \neq 0$ and $[\![\,\bigvee_{i=1}^{k} G_i\,]\!] = 1$. Let us consider the set $\{G'_n\}_{n \in F}$ of guessings such that $G'_n := G_i$ and $n \in [\![\,G_i\,]\!]^{\mathcal{F}}$ (this is well-set since for every $n \in \mathcal{F}$ there exists only one G_i such that n belongs to $[\![\,G_i\,]\!]^{\mathcal{F}}$). By Lemma 1.3.9,[11] we obtain a set $\mathcal{M} := \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n \in F}$ of $\Sigma^{\underline{a},\underline{c}}$-structures such that (i) $\mathcal{M}_{i\,|\Sigma_r^{\underline{c}}} = \mathcal{M}_{j\,|\Sigma_r^{\underline{c}}}$ $(i, j \in F)$ and (ii) $\mathcal{M}_i \models T \cup G'_i$ $(i \in F)$.

$(\mathcal{F}, \mathcal{M})$ is an $I(\Sigma^{\underline{a},\underline{c}})$-structure appropriate for $\mathcal{T}$ that, by construction, for every atom ℓ occurring in φ satisfies the condition: $\mathcal{M} \models_n \ell$ iff $i \in [\![\,\ell\,]\!]^{\mathcal{F}}$. We want to show that $\mathcal{F}, \mathcal{M} \models_n \varphi$ iff $n \in [\![\,\varphi\,]\!]^{\mathcal{F}}$ (the thesis follows since there exists at least an element $n \in F$ such that $n \in [\![\,\varphi\,]\!]^{\mathcal{F}}$, because $\mathcal{F}$ satisfies $[\![\,\varphi\,]\!] \neq 0$). This can be done by induction on the complexity of the subformulae ψ occurring in φ. The condition is obvious if ψ is an atom, and follows directly from the induction hypothesis if it is of the kind $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\neg\psi$. Suppose now ψ is of the kind $O(\psi_1, \ldots, \psi_n)$, we want to show that $\mathcal{F}, \mathcal{M} \models_i O(\psi_1, \ldots, \psi_n)$ iff $i \in [\![\,O(\psi_1, \ldots, \psi_n)\,]\!]^{\mathcal{F}}$. The 'only if' case is argued as follows: $\mathcal{F}, \mathcal{M} \models_i O(\psi_1, \ldots, \psi_n)$ implies $i \in O^{\mathcal{F}}(\{u \mid \mathcal{F}, \mathcal{M} \models_u \psi_1\}, \ldots, \{u \mid \mathcal{F}, \mathcal{M} \models_u \psi_n\})$, thus, by induction hypothesis, $i \in O^{\mathcal{F}}([\![\,\psi_1\,]\!]^{\mathcal{F}}, \ldots, [\![\,\psi_n\,]\!]^{\mathcal{F}})$, hence $i \in \big(O([\![\,\psi_1\,]\!], \ldots, [\![\,\psi_n\,]\!])\big)^{\mathcal{F}}$, and finally, by Definition 2.5.5 of I-Abstraction, $i \in [\![\,O(\psi_1, \ldots, \psi_n)\,]\!]^{\mathcal{F}}$. The 'if' case is

[11] Lemma 1.3.9 is used with $I := F$, and $T_i := T$, but symbols from $\Sigma \setminus \Sigma_r$ are disjointly renamed when building the signature Σ_i for the i^{th} copy of T (the same observation applies also to the flexible constants $\underline{a}$). In this way, a model of $\bigcup_i T_i$ is the same thing as a set of models $\{\mathcal{M}'_n\}_{n \in F}$ of T whose $\Sigma_r^{\underline{c}}$-reducts coincide.

analogous. □

Lemma 2.5.7 (Completeness). *Let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a Noetherian compatible data-flow theory, L be an abstract temporal logic and φ be a ground $I(\Sigma^{\underline{a},\underline{c}})$-sentence. If there is an $I(\Sigma^{\underline{a},\underline{c}})$-structure $(\mathcal{F}, \mathcal{M})$ appropriate for $\mathcal{T}$ such that $\mathcal{F} \in L$ and such that $\mathcal{F}, \mathcal{M} \models_t \varphi$ holds for some t, then* NSAT*(φ) returns "satisfiable".*

Proof. The proof can be easily obtained from the proof of Lemma 2.4.4: indeed, using the new definition of compatible guessing, observe that the condition stating that, for every atom ℓ occurring in φ, $n \in [\![\ell]\!]^{\mathcal{F}}$ iff $\mathcal{F}, \mathcal{M} \models_n \ell$ still holds; this implies, by the induction argument used in the proof of the previous lemma, that $\mathcal{F}$ satisfies $[\![\varphi]\!] \neq 0$ and $[\![\bigvee_i G_i]\!] = 1$. □

While Theorem 2.4.5 is relevant to augment the degree of mechanization of deductive approaches for the verification of reactive systems based on LTL (e.g., the one put-forward in Manna and Pnueli, 1995), one may wonder about the relevance of its generalization, i.e. Theorem 2.5.4. To see its usefulness, consider Temporal Logic of Actions (TLA, see Lamport, 1994). For such a specification formalism, it is difficult to reuse techniques and tools for (classic) temporal/modal logic since TLA features some non-standard characteristics which are quite useful for practitioners (see Merz, 2003 for an extensive discussion on this and related issues). On the other hand, deductive verification of TLA specifications can be supported by proof assistants (see, e.g., Merz, 1999). While applying the inference rules of TLA, it has been observed in Merz (2003) that some of the resulting sub-goals may belong to a fragment of TLA which is equivalent to the modal logic $S4.2$ (see, e.g., Blackburn et al., 2002). Now, the relativized satisfiability problem for this logic is decidable (see again Blackburn et al., 2002) so that NSAT can be used to automatically discharge some of the sub-goals, whenever the data-flow theory formalizing the data structure manipulated by the system modelled in TLA is Noetherian compatible.

2.6 Conclusions and Related Work

In this chapter, we have considered first-order LTL. We have studied the decidability of the "temporalized" satisfiability problem for quantifier-free formulae modulo a background first-order theory axiomatizing the extensional part of the language. The key technique to obtain our results is a reduction to constraint satisfiability problems in unions of first-order theories over non-disjoint signature: this reduction

allowed us to derive undecidability results, but also decidability results, through suitable adaptations of extensions of the Nelson-Oppen schema (see, e.g., Bonacina et al., 2006; Ghilardi, 2004). In case a local finiteness requirement of the theory over the rigid signature is fulfilled, we have reduced the satisfiability problem for "temporalized" first-order theories to satisfiability in propositional LTL. When Noetherianity comes into play, instead, the combination turns out to be more complex and needs an exchange mechanism which relies on the use of residue enumerators. Finally, the decidability result has been extended to any modal/temporal logic whose propositional relativized satisfiability problem is decidable.

The undecidability of quantified modal logic over a discrete flow was discovered by D. Scott already in the sixties. Recent works isolated quite interesting fragments of quantified LTL which are computationally better behaved (see Gabbay et al., 2003 for a survey). In this chapter, we have taken a similar approach to the one in Manna and Pnueli (1995) by forbidding the interplay between quantifiers and temporal operators and by enriching the extensional part of the language so to be able to model infinite data structures manipulated by systems. An approach similar to ours was already taken in the seminal paper Plaisted (1986), where the author established a decidability result when the quantifier-free fragment of T is decidable and the flexible symbols are considered as free symbols by the theory T. By using recent techniques and results from the combination literature, we were able to attack the problem in its full generality and derive both the undecidability in the unrestricted case and the decidability under the 'combinability' hypotheses for T of Ghilardi (2004).

Chapter 3

Model Checking

Although it is well-known that full first-order temporal logics are highly undecidable, in Manna and Pnueli (1995) the authors have shown that a combination of first-order logic and LTL is needed in order to precisely state verification problems for the class of reactive systems. In that kind of combined formalism, the role of first-order theories is to describe (possibly infinite) data structures used by the system, whereas LTL is used to specify the behavior of the system during the flow of time. It turns out that this combined formalism is expressive enough to write concise and abstract specifications. In this context the problem of identifying useful fragments amenable of automated analysis immediately arises.

In this chapter, after proving an undecidability result, we concentrate on the problem of deriving sufficient condition for guaranteeing the decidability of the model checking problem. To this aim, we rely on the framework introduced in the previous chapter enriched with the capability of expressing transition system through LTL-system specification.

3.1 LTL-System Specifications and the Model Checking Problem

In order to introduce the model checking problems, we need some preliminary notions.

Definition 3.1.1. Given two signatures Σ_r and Σ such that $\Sigma_r \subseteq \Sigma$, we define the

one-step signature as follows:

$$\Sigma \underset{\Sigma_r}{\oplus} \Sigma \quad := \quad ((\Sigma \setminus \Sigma_r) \uplus (\Sigma \setminus \Sigma_r)) \cup \Sigma_r,$$

where $\uplus$ denotes disjoint union.

In order to build the one-step signature $\Sigma \oplus_{\Sigma_r} \Sigma$, we first consider two copies of the symbols in $\Sigma \setminus \Sigma_r$; the two copies of $s \in \Sigma \setminus \Sigma_r$ are denoted by s^0 and s^1, respectively. Notice that the symbols in Σ_r are not renamed. Also, arities in the one-step signature $\Sigma \oplus_{\Sigma_r} \Sigma$ are defined in the obvious way: the arities of the symbols in Σ_r are unchanged and if n is the arity of $s \in \Sigma \setminus \Sigma_r$, then n is the arity of both s^0 and s^1. The one-step signature $\Sigma \oplus_{\Sigma_r} \Sigma$ will be also written as $\bigoplus^2_{\Sigma_r} \Sigma$; similarly, we can define the n-step signature $\bigoplus^{n+1}_{\Sigma_r} \Sigma$ for $n > 1$ (our notation for the copies of $(\Sigma \setminus \Sigma_r)$-symbols extends in the obvious way, that is we denote by $s^0, s^1, \ldots, s^n$ the $n+1$ copies of s).

Given a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$, the one-step signature $\Sigma^{\underline{a},\underline{c}} \oplus_{\Sigma^{\underline{c}}_r} \Sigma^{\underline{a},\underline{c}}$ is appropriate to express constraints about the dynamic behavior of a system during one time unit. In fact, the transition relation of a system must be able to simultaneously refer to the structures representing the state of the system in two consecutive time instants. In this respect, non-rigid symbols are duplicated whereas rigid symbols are left unchanged.

We now define the concepts of one-step structure and one-step theory, which are the semantic counterparts of the one-step signature introduced above (cf. Definition 3.1.1).

Definition 3.1.2. Given two signatures Σ_r and Σ such that $\Sigma_r \subseteq \Sigma$, two Σ-structures $\mathcal{M}_0 = \langle M, \mathcal{I}_0 \rangle$ and $\mathcal{M}_1 = \langle M, \mathcal{I}_1 \rangle$ whose Σ_r-reducts are the same,[1] the *one-step* $(\Sigma \oplus_{\Sigma_r} \Sigma)$*-structure*

$$\mathcal{M}_0 \underset{\Sigma_r}{\oplus} \mathcal{M}_1 := \langle M, \mathcal{I}_0 \underset{\Sigma_r}{\oplus} \mathcal{I}_1 \rangle$$

is defined as follows:

- for each function or predicate symbol $s \in \Sigma \setminus \Sigma_r$, we have $(\mathcal{I}_0 \oplus_{\Sigma_r} \mathcal{I}_1)(s^0) := \mathcal{I}_0(s)$ and $(\mathcal{I}_0 \oplus_{\Sigma_r} \mathcal{I}_1)(s^1) := \mathcal{I}_1(s)$;
- for each function or predicate symbol $s \in \Sigma_r$, we have $(\mathcal{I}_0 \oplus_{\Sigma_r} \mathcal{I}_1)(s) := \mathcal{I}_0(s)$.

[1] Recall from Section 1.1 that this means that $\mathcal{I}_0(s) = \mathcal{I}_1(s)$ for all $s \in \Sigma_r$.

If φ is a Σ-formula, the $(\Sigma \oplus_{\Sigma_r} \Sigma)$-formulae φ^0, φ^1 are obtained from φ by replacing each symbol $s \in \Sigma \setminus \Sigma_r$ by s^0 and s^1, respectively. The one-step theory is nothing but a combination of a theory T with a partially renamed copy of itself.

Definition 3.1.3. Given two signatures Σ_r and Σ such that $\Sigma_r \subseteq \Sigma$, the theory $T \oplus_{\Sigma_r} T$ is defined by $\{\varphi^0 \wedge \varphi^1 \mid \varphi \in T\}$.

We will write $\bigoplus^2_{\Sigma_r} T$ instead of $T \oplus_{\Sigma_r} T$; the n-step theories $\bigoplus^{n+1}_{\Sigma_r} T$ (for $n > 1$) are similarly defined.

Let now $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a data-flow theory whose parameters and whose system variables are *finite*. A *transition relation* for the data-flow theory $\mathcal{T}$ is a $(\Sigma^{\underline{a},\underline{c}} \oplus_{\Sigma^{\underline{c}}_r} \Sigma^{\underline{a},\underline{c}})$-sentence δ: we usually write such formula as $\delta(\underline{a}^0, \underline{a}^1)$ to emphasize that it contains the two copies of the system variables $\underline{a}$ (on the other hand, the system parameters $\underline{c}$ that are not duplicated will never be displayed). Examples of transition relations are the *tautological* transition $\delta_\top := \top$ and the *idle* transition:

$$\delta_I \quad := \quad \bigwedge_a (a^0 = a^1) \wedge \bigwedge_P \forall \underline{x} (P^0(\underline{x}) \leftrightarrow P^1(\underline{x})) \wedge \bigwedge_f \forall \underline{x} (f^0(\underline{x}) = f^1(\underline{x})),$$

where a ranges over free constants in $\underline{a}$, P over predicate symbols in $\Sigma \setminus \Sigma_r$, and f over function symbols in $\Sigma \setminus \Sigma_r$.

An *initial state description* for a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ – with finitely many system variables and parameters – is simply a $\Sigma^{\underline{a},\underline{c}}$-sentence $\iota(\underline{a})$ (as it was the case for transition relations, the system parameters $\underline{c}$ will not be displayed also for state descriptions).

Definition 3.1.4 (LTL-System Specification and Model Checking)**.** An *LTL-system specification* is a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ (having finitely many system variables and system parameters) endowed with a transition relation $\delta(\underline{a}^0, \underline{a}^1)$ and with an initial state description $\iota(\underline{a})$. An LTL$(\Sigma^{\underline{a},\underline{c}})$-structure $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n \in \mathbb{N}}$ is *a run* for such an LTL-system specification iff it is appropriate for $\mathcal{T}$ and moreover it obeys the transition δ and the initial state description ι, meaning that:

(i) $\mathcal{M}_n \oplus_{\Sigma^{\underline{c}}_r} \mathcal{M}_{n+1} \models \delta(\underline{a}^0, \underline{a}^1)$, for every $n \geq 0$;

(ii) $\mathcal{M}_0 \models \iota(\underline{a})$.

The *model checking problem* for the LTL-system specification $(\mathcal{T}, \delta, \iota)$ is the following: given an LTL$(\Sigma^{\underline{a},\underline{c}})$-sentence φ, decide whether there is a run $\mathcal{M}$ for $(\mathcal{T}, \delta, \iota)$

such that $\mathcal{M} \models \varphi$.[2] The *ground model checking problem for* $(\mathcal{T}, \delta, \iota)$ is similarly introduced, but φ is assumed to be ground.

Roughly speaking, the satisfiability problem for data-flow theories (cf. Definition 2.1.5) is equivalent to the model checking problem for LTL-system specifications endowed with tautological transition and tautological initial state description (there is a little difference, however, due to the fact that the satisfiability problem is relative to data-flow theories having possibly infinitely many system parameters and variables, whereas LTL-system specifications must have finitely many system variables and parameters).

An important subclass of model checking problems is the following: the (syntactic) *safety model checking problem* is the model checking problem for formulae of the form

$$\Diamond \upsilon,$$

where υ is a $\Sigma^{\underline{a},\underline{c}}$-sentence. Since υ is meant to describe the set of *unsafe* states, we say that the LTL-system specification $(\mathcal{T}, \delta, \iota)$ is *safe for* υ iff the model checking problem for $\Diamond\upsilon$ has a negative solution. This implies that $\Box\neg\upsilon$ is true for all runs of $(\mathcal{T}, \delta, \iota)$.

3.1.1 The Seriality Property

In the literature about model checking (especially, for finite-state systems), it is usually assumed the seriality of the transition relation, i.e. that every state of the system must have at least one successor state (see, e.g., Clarke et al., 1999 for more details). Unfortunately, it is difficult to find an effective formulation of such a requirement in our framework because the states of the system $(\mathcal{T}, \delta, \iota)$ are the models of the (first-order) theory underlying $\mathcal{T}$. Below, we give a non-effective formulation for seriality in our framework. Fortunately, as we shall see, there exist simple and effective methods to ensure it.

Definition 3.1.5. An LTL-system specification $(\mathcal{T}, \delta, \iota)$, based on the data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$, is said to be *serial* iff for every $\Sigma^{\underline{a},\underline{c}}$-structure $\mathcal{M}_0 =$

[2] Notice that usually the model checking problem is taken to be the complement of our model checking problem, i.e. it is taken to be the problem of deciding whether a given sentence is true in all runs. As far as we are concerned with decidability/undecidability issues, the difference is immaterial (for complexity questions, one must take the complementary classes). Our choice is motivated by the need of having a homogeneous terminology with satisfiability problems.

$(M, \mathcal{I}_0)$ which is a model of T, there is another $\Sigma^{\underline{a},\underline{c}}$-structure $\mathcal{M}_1 = (M, \mathcal{I}_1)$ (still a model of T) such that $(\mathcal{M}_1)_{|\Sigma^{\underline{c}}_r} = (\mathcal{M}_2)_{|\Sigma^{\underline{c}}_r}$ and $\mathcal{M}_0 \oplus_{\Sigma^{\underline{c}}_r} \mathcal{M}_1 \models \delta(\underline{a}^0, \underline{a}^1)$.

In order to be able to ensure the above requirement for concrete situations, the following observations are useful:

(i) if the transition relation δ consists of the conjunction of (possibly guarded) assignments of the form $P \wedge a^1 = t^0(\underline{a}^0)$ where P is the condition under which the assignment is executed, then δ is serial (this is the case, for instance, of the water level controller example discussed in Section 3.4);

(ii) δ is serial when it is implied by the idle transition, i.e. in case $T \oplus_{\Sigma_r} T \models \delta_I \rightarrow \delta$ (this is equivalent to $T \models \delta^\sharp$, where $\delta^\sharp$ is obtained from δ by replacing the copies s^1, s^2 of every flexible symbol by s);[3]

(iii) every transition δ can be 'adjusted' in order to make it serial; to this end, it is sufficient to add a fresh 0-ary relational symbol E (standing for 'error') to Σ and replace δ by

$$\delta_E \quad := \quad (\neg E^0 \wedge \delta \wedge \neg E^1) \vee (\neg E^0 \wedge E^1) \vee (E^0 \wedge E^1).$$

3.1.2 Some Classes of LTL-Systems and Further Assumptions

In Subsection 2.1.2 of Chapter 2, we have introduced three different classes of data-flow theories of increasing expressiveness so to study the satisfiability problem for data-flow theories. Here, we introduce the corresponding classes of LTL-systems so to study the decidability of the safety model checking problem.

Definition 3.1.6. An LTL-system specification based on a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is said to be *finite state* iff $\mathcal{T}$ is totally rigid and T is an enumerated datatype theory.

Finite state LTL-system specifications are investigated by traditional symbolic model checking literature (see Clarke et al., 1999) and are efficiently handled by state-of-the-art tools like NuSMV (see Cimatti et al., 2002).

Definition 3.1.7. An LTL-system specification based on a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is said to be *locally finite compatible* iff there is a Σ_r-universal and effectively locally finite theory T_r such that T is T_r-compatible.

[3] If the constraint satisfiability problem of T is decidable and if δ is ground (as it is the case for some of the examples considered in this chapter), the condition $T \models \delta^\sharp$ can be effectively checked.

As for compatible theories, from our discussion in Section 1.2 in Chapter 1, it follows that an LTL-system based on totally flexible data-flow theory is locally finite compatible in case its underlying theory is stably infinite.

Definition 3.1.8. An LTL-system specification based on a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ is said to be *Noetherian compatible* iff there is a Σ_r-universal theory T_r such that T is an effectively Noetherian and T_r-compatible extension of T_r.

Since we are interested in positive results for safety model checking problems, we need to make some restrictions; we first of all assume the following (which completes Assumption 2.1.9 from Subsection 2.1.2 in Chapter 2):

Assumption 3.1.9. For any LTL-system specification $(\mathcal{T}, \delta, \iota)$ (considered in the rest of this chapter) we assume that:

(i) the underlying theory of $\mathcal{T}$ has decidable constraint satisfiability problem;

(ii) the transition relation δ and the initial state description ι are ground sentences;

(iii) $(\mathcal{T}, \delta, \iota)$ is serial.

3.2 Undecidability and Noetherian Data-Flow Theories

Even under the above assumption, the ground model checking problem for an LTL-system specification based on a totally rigid data-flow theory is undecidable: this is a standard result that can be obtained through a simple reduction to the (undecidable) reachability problem of Minsky machines (see Ebbinghaus et al., 1994; Minsky, 1961). We give details below for the sake of completeness.

A *two registers Minsky machine* is a finite set $\mathbf{P}$ of instructions (also called a program) for manipulating configurations seen as triples (s, m, n) of natural numbers, where s represents the machine state and m, n the contents of the two registers. There are four possible kinds of instructions, inducing transformations on the configurations as explained in Table 3.1.

A $\mathbf{P}$-transformation is a transformation induced by an instruction of $\mathbf{P}$ on a certain configuration. For a Minsky machine $\mathbf{P}$, we write $(s, m, n) \rightarrow_{\mathbf{P}}^{\star} (s', m', n')$ to say that it is possible to reach configuration (s', m', n') from (s, m, n) by applying finitely many $\mathbf{P}$-transformations. Given a Minsky machine $\mathbf{P}$ and an initial configuration (s_0, m_0, n_0), the problem of checking whether a configuration (s', m', n') is

N.	Instruction	Transformation
I	$s \to (t, 1, 0)$	$(s, m, n) \to (t, m+1, n)$
II	$s \to (t, 0, 1)$	$(s, m, n) \to (t, m, n+1)$
III	$s \to (t, -1, 0)[(t', 0, 0)]$	`if` $m \neq 0$ `then` $(s, m, n) \to (t, m-1, n)$ `else` $(s, m, n) \to (t', m, n)$
IV	$s \to (t, 0, -1)[(t', 0, 0)]$	`if` $n \neq 0$ `then` $(s, m, n) \to (t, m, n-1)$ `else` $(s, m, n) \to (t', m, n)$

Table 3.1: Instructions and related transformations for (two-registers) Minsky Machines

reachable from (s_0, m_0, n_0) (i.e., if $(s_0, m_0, n_0) \to^\star_{\mathbf{P}} (s', m', n')$ holds or not) is called *the (second) reachability (configuration) problem*. It is well-known (see Chagrov and Zakharyaschev, 1997) that there exists a (two-register) Minsky machine $\mathbf{P}$ and a configuration (s_0, m_0, n_0) such that the second reachability configuration problem is undecidable.

Theorem 3.2.1. *There exists a totally rigid and Noetherian compatible LTL-system specification $(\mathcal{T}, \delta, \iota)$, whose ground safety model checking problem is undecidable.*

Proof. The proof consists of two steps. First, we need to define a totally rigid data-flow theory $\mathcal{T}$ which is expressive enough to encode unbounded counters and which satisfies our Assumption 3.1.9(i) above. Second, we must define the encoding of a Minsky machine into an LTL-system specification based on $\mathcal{T}$ so that the second reachability problem of such machine can be represented as a safety model checking problem. This immediately gives the undecidability of the latter, as desired.

Let us consider the Σ_C-theory T_C, where

- Σ_C consists of two unary function symbols s, p and a constant 0;
- T_C contains all Σ_C-sentences which are true in the structure $(\mathbb{Z}, s, p, 0)$ of the Integers with zero, successor, and predecessor.[4]

Notice that T_C is Noetherian, though not locally finite. Indeed, the Noetherianity of T_C can be argued from the following arguments: (i) the pure theory of equality over the signature containing a unary function symbol is Noetherian (cf. Proposition 1.5.1 from Section 1.5 in Chapter 1); (ii) any extension (over a signature augmented of a

[4] It is possible to use also the structure given by $\mathbb{N}$, 0, successor, and predecessor (the latter is turned into a total function by putting $p(0) := 0$).

finite number of constant symbols) of a Noetherian theory remains Noetherian; (iii) every Σ_C-formula is T_C-logically equivalent to a $(\Sigma_C \setminus \{p\})$-formula).[5] Moreover, the constraint satisfiability problem of T_C is decidable by quantifier elimination (it is straightforward to adapt the algorithm for the naturals in Enderton, 1972). T_C can be seen as a 'minimal' theory where to encode an unbounded counter as it is required in order to express the instructions of the Minsky machines of Table 3.1 (below, we abbreviate $\underbrace{s(\cdots(s}_{n \text{ times}}(0)\cdots))$ with the numeral $\overline{n}$).

We define the totally rigid data-flow theory $\mathcal{T}$ as follows: T_C is underlying theory, there are three system variables $\{a_1, a_2, a_3\}$, and no parameters. Since $\mathcal{T}$ is totally rigid, it is completely determined by its underlying theory, its systems variables, its parameters, and there is no need to specify a rigid subsignature, because all predicate and function symbols are rigid.

We are now in the position to define the encoding of a second reachability problem for a Minsky machine into an LTL-system based on $\mathcal{T}$: we do it for a Minsky machine $\mathbf{P}$ and for a configuration (s_0, m_0, n_0) such that $\mathbf{P}$-reachability from (s_0, m_0, n_0) is undecidable.

The transition δ is the disjunction of the following ground sentences:

– for each $\mathbf{P}$-instruction $s \to (t, 1, 0)$ of the first kind, δ contains the disjunct

$$a_1^0 = \overline{s} \wedge a_1^1 = \overline{t} \wedge a_2^1 = s(a_2^0) \wedge a_3^1 = a_3^0;$$

– for each $\mathbf{P}$-instruction $s \to (t, 0, 1)$ of the second kind, δ contains the disjunct

$$a_1^0 = \overline{s} \wedge a_1^1 = \overline{t} \wedge a_2^1 = a_2^0 \wedge a_3^1 = s(a_3^0);$$

– for each $\mathbf{P}$-instruction $s \to (t, -1, 0)[(t', 0, 0)]$ of the third kind, δ contains the disjuncts

$$\begin{aligned} &\left(a_2^0 \neq 0 \wedge a_1^0 = \overline{s} \wedge a_1^1 = \overline{t} \wedge a_2^1 = p(a_2^0) \wedge a_3^1 = a_3^0\right) \vee \\ &\vee \left(a_2^0 = 0 \wedge a_1^0 = \overline{s} \wedge a_1^1 = \overline{t'} \wedge a_2^1 = a_2^0 \wedge a_3^1 = a_3^0\right); \end{aligned}$$

– for each $\mathbf{P}$-instruction $s \to (t, 0, -1)[(t', 0, 0)]$ of the fourth kind, δ contains

[5] In particular, every chain of sets of Σ_C-atoms is T_C-equivalent to a chain of sets of $(\Sigma_C \setminus \{p\})$-atoms. Since this latter has to be eventually constant for logical consequence w.r.t. T_C, so it is the former.

the disjuncts

$$\big(a_3^0 \neq 0 \wedge a_1^0 = \overline{s} \wedge a_1^1 = \overline{t} \wedge a_2^1 = a_2^0 \wedge a_3^1 = p(a_3^0)\big) \vee$$
$$\vee \big(a_3^0 = 0 \wedge a_1^0 = \overline{s} \wedge a_1^1 = \overline{t'} \wedge a_2^1 = a_2^0 \wedge a_3^1 = a_3^0\big);$$

– finally, δ contains also the idle disjunct

$$a_1^0 = a_1^1 \wedge a_2^1 = a_2^0 \wedge a_3^1 = a_3^0$$

(this disjunct is added in order to make the transition serial).

Let ι be the ground sentence $a_1 = \overline{s_0} \wedge a_2 = \overline{m_0} \wedge a_3 = \overline{n_0}$. We claim that, for a given configurations (s', m', n'), we have that $(s_0, m_0, n_0) \rightarrow_{\mathbf{P}}^{\star} (s', m', n')$ iff the formula

$$\Diamond(a_1 = \overline{s'} \wedge a_2 = \overline{m'} \wedge a_3 = \overline{n'})$$

is satisfied in a run of $(\mathcal{T}, \delta, \iota)$. The 'only if' implication of the claim is trivial. For the converse, suppose that there is a run $\mathcal{M}$ of $(\mathcal{T}, \delta, \iota)$ such that

$$\mathcal{M} \models_k a_1 = \overline{s'} \wedge a_2 = \overline{m'} \wedge a_3 = \overline{n'}$$

for some $k \geq 0$. First, notice that one may freely assume that a non-idle disjunct of δ is true in the i^{th} transition step for $0 \leq i \leq k-1$ (otherwise we can simply remove that step and get a smaller k). Second, as the data-flow theory $\mathcal{T}$ is totally rigid, only the interpretation of the system variables a_1, a_2, a_3 can be different at each time instant – the Σ_C-reduct of the various $\mathcal{M}_i$ being always the same. Such a reduct contains an (elementary) substructure which is isomorphic to the standard model $(\mathbb{Z}, s, p, 0)$ of integers (this is the substructure whose support is the collection of the interpretations of the numerals); moreover, as the system variables take values in the positive subset of that substructure at the initial instant, it is impossible for them to get values outside it for the whole run (to see this, just make an inspection to the definition of the transition δ). This immediately yields $(s_0, m_0, n_0) \rightarrow_{\mathbf{P}}^{\star} (s', m', n')$, as desired. □

3.3 Decidability and Locally Finite Data-Flow Theories

In order to be able to give decidability results for the (safety) model checking problem, we need to introduce the following preliminary definitions about one-step formulae.

Definition 3.3.1. A ground $(\Sigma^{\underline{a},\underline{c}} \oplus_{\Sigma_r^{\underline{c}}} \Sigma^{\underline{a},\underline{c}})$-sentence δ is said to be *purely left* (*purely right*) iff for each symbol $s \in \Sigma \setminus \Sigma_r$, we have that s^1 (s^0, resp.) does not occur in δ. We say that δ is *pure* iff it is a Boolean combination of purely left or purely right atoms.

Given a formula $\delta(\underline{a}^0, \underline{a}^1)$, it is always possible (see, e.g., Ghilardi, 2004) to obtain an equisatisfiable formula $\hat{\delta}(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ which is pure by introducing "fresh" constants that we call $\underline{d}^0$ (i.e., $\underline{d}^0 \cap (\underline{a}^0 \cup \underline{a}^1) = \emptyset$) to name "impure" subterms. Usually, $\hat{\delta}$ is called the *purification* of δ. Let $A_1, \ldots, A_k$ be the atoms occurring in $\hat{\delta}(\underline{a}^0, \underline{a}^1, \underline{d}^0)$.

Definition 3.3.2 ($\hat{\delta}$-assignment)**.** A $\hat{\delta}$-*assignment* is a conjunction $B_1 \wedge \cdots \wedge B_k$ (where B_i is either A_i or $\neg A_i$, for $1 \leq i \leq k$), such that $B_1 \wedge \cdots \wedge B_k \rightarrow \hat{\delta}$ is a propositional tautology.

Since $\hat{\delta}$ is pure, we can represent a $\hat{\delta}$-assignment V in the form $V^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V^r(\underline{a}^0, \underline{a}^1, \underline{d}^0)$, where V^l is a purely left conjunction of literals and V^r is a purely right conjunction of literals.

3.3.1 Safety Model Checking

Fortunately, the safety model checking problem is decidable for locally finite compatible LTL-system specifications. In the rest of this section, let $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ be a locally finite compatible data-flow theory, $(\mathcal{T}, \delta, \iota)$ be an LTL-system specification based on $\mathcal{T}$, and $\upsilon(\underline{a})$ be a ground $\Sigma^{\underline{a},\underline{c}}$-sentence. The related safety model checking problem amounts to checking whether there exists a run $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$ for $(\mathcal{T}, \delta, \iota)$ such that $\mathcal{M} \models_n \upsilon(\underline{a})$ for some $n \geq 0$: if this is the case, we say that the system is *unsafe* since there is a *bad run of length* n.

We can ignore bad runs of length $n = 0$, because the existence of such runs can be preliminarily decided by checking the ground sentence $\iota(\underline{a}) \wedge \upsilon(\underline{a})$ for T-satisfiability. So, for $n \geq 1$, taking into account the seriality of the transition, a bad run of length $n + 1$ exists iff the ground $(\bigoplus_{\Sigma_r^{\underline{c}}}^{n+2} \Sigma^{\underline{a},\underline{c}})$-sentence

$$\iota^0(\underline{a}^0) \wedge \delta^{0,1}(\underline{a}^0, \underline{a}^1) \wedge \delta^{1,2}(\underline{a}^1, \underline{a}^2) \wedge \cdots \wedge \delta^{n,n+1}(\underline{a}^n, \underline{a}^{n+1}) \wedge \upsilon^{n+1}(\underline{a}^{n+1}) \qquad (3.1)$$

is $\bigoplus_{\Sigma_r^c}^{n+2} T$-satisfiable, where $\iota^0(\underline{a}^0)$ is obtained by replacing each flexible symbol $s \in \Sigma \setminus \Sigma_r$ with s^0 in $\iota(\underline{a})$ (the system variables $\underline{a}$ are similarly renamed as $\underline{a}^0$); $\delta^{i,i+1}(\underline{a}^i, \underline{a}^{i+1})$ is obtained by replacing in $\delta(\underline{a}^0, \underline{a}^1)$ the copy s^0 and s^1 of each flexible symbol $s \in \Sigma \setminus \Sigma_r$ with s^i and s^{i+1} respectively (the two copies $\underline{a}^0, \underline{a}^1$ of the system variables $\underline{a}$ are similarly renamed as $\underline{a}^i, \underline{a}^{i+1}$); and $\upsilon^{n+1}(\underline{a}^{n+1})$ is obtained by replacing each flexible symbol $s \in \Sigma \setminus \Sigma_r$ with s^{n+1} in $\upsilon(\underline{a})$ (the system variables $\underline{a}$ are similarly renamed as $\underline{a}^{n+1}$). For the sake of simplicity, we will write formula (3.1) by omitting the superscripts of ι, δ, and υ (but we maintain those of the system variables $\underline{a}$).

Now, for a given $n+1$, an iterated application of the main combination result in Ghilardi (2004) and the fact that T_0-compatibility is a modular property (see again Ghilardi, 2004) yield the decidability of the satisfiability of formula (3.1). Unfortunately, this is not sufficient to solve the model checking problem for LTL-system specifications since the length of a run is not known apriori. To solve this problem, instead of considering the transition relation δ we focus on its purification $\hat{\delta}$ (cf. Section 3.3). By the fact that δ and $\hat{\delta}$ are equisatisfiable, a bad run of length $n+1$ exists iff the ground sentence

$$\iota(\underline{a}^0) \wedge \bigwedge_{i=0}^{n} (V_{i+1}^l(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i) \wedge V_{i+1}^r(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)) \wedge \upsilon(\underline{a}^{n+1}) \tag{3.2}$$

is $\bigoplus_{\Sigma_r}^{n+2} T$-satisfiable, where $\underline{d}^0, \underline{d}^1, \ldots, \underline{d}^n$ are $n+1$ copies of the fresh constants $\underline{d}^0$ and $V_1, \ldots, V_{n+1}$ range over the set of $\hat{\delta}$-assignments. Since T_r is locally finite, there are finitely many ground $\Sigma_r^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}$-literals which are representative (modulo T_r-equivalence) of all $\Sigma_r^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}$-literals. A guessing $G(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ (cf. Definition 2.3.4) over such literals will be called a *transition* Σ_r*-guessing*.

Definition 3.3.3. The *safety graph* associated to the LTL-system specification $(\mathcal{T}, \delta, \iota)$ based on the locally finite compatible data-flow theory $\mathcal{T}$ is the directed graph defined as follows:

- the nodes are the pairs (V, G) where V is a $\hat{\delta}$-assignment and G is a transition Σ_r-guessing;
- there is an edge $(V, G) \to (W, H)$ iff the ground sentence

$$G(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge W^l(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge H(\underline{a}^1, \underline{a}^2, \underline{d}^1) \tag{3.3}$$

is T-satisfiable.

The *initial nodes* of the safety graph are the nodes (V, G) such that $\iota(\underline{a}^0) \wedge V^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ is T-satisfiable; the *terminal nodes* of the safety graph are the nodes (V, G) such that $V^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge \upsilon(\underline{a}^1) \wedge G(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ is T-satisfiable.

In formula (3.3) we follow our convention of writing only the system variable renamings (flexible symbols being renamed accordingly). More in detail: we make three copies s^0, s^1, s^2 of every flexible symbol $s \in \Sigma \setminus \Sigma_r$. Both V^r and W^l might contain in principle two copies s^0, s^1 of s: the two copies in V^r keep their original names, whereas the two copies in W^l are renamed as s^1, s^2, respectively. However, V^r is a right formula (hence it does not contain s^0) and W^l is a left formula (hence it does not contain s^1): the moral of all this is that only the copy s^1 of s occurs after renaming, which means that (3.3) is after all just a plain $\Sigma^{\underline{a}^0, \underline{a}^1, \underline{a}^2, \underline{d}^0, \underline{d}^1}$-sentence (thus, it makes sense to test it for T-satisfiability). Notice that the Skolem constants $\underline{d}^0$ of V^r are renamed as $\underline{d}^1$ in W^l.

The decision procedure for safety model checking relies on the following fact.

Proposition 3.3.4. *The system is unsafe iff either $\iota(\underline{a}) \wedge \upsilon(\underline{a})$ is T-satisfiable or there is a path in the safety graph from an initial to a terminal node.*

Proof. Recall from Subsection 3.3.1 that a bad run of length $n+1$ exists iff the ground sentence

$$\iota(\underline{a}^0) \wedge \bigwedge_{i=0}^{n} (V_{i+1}^l(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i) \wedge V_{i+1}^r(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)) \wedge \upsilon(\underline{a}^{n+1}) \tag{3.2}$$

is $(\bigoplus_{\Sigma_r}^{n+2} T)$-satisfiable, where the V_{i+1} range over the set of $\hat{\delta}$-assignments.

Preliminary to the main argument of the proof, which is based on interpolations, let us better analyze the shape of the formula (3.2) with particular attention to symbols occurring in the various literals. In formula (3.2), each symbol $s \in \Sigma \setminus \Sigma_r$ can occur in $n+2$ copies $s^0, s^1, \ldots, s^{n+1}$ and the locations of these copies are the following:

(i) s^0 can only occur in $\iota(\underline{a}^0) \wedge V_1^l(\underline{a}^0, \underline{a}^1, \underline{d}^0)$;

(ii) s^i can only occur in $V_i^r(\underline{a}^{i-1}, \underline{a}^i, \underline{d}^{i-1}) \wedge V_{i+1}^l(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)$, for $i = 1, \ldots, n$;

(iii) s^{n+1} can only occur in $V_{n+1}^r(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge \upsilon(\underline{a}^{n+1})$.

Now, we are ready to develop the main argument of the proof. Suppose that the system is unsafe. Then, either there is a bad run of length 0 or the formula (3.2) is satisfiable in a model $\mathcal{N}$ of $\bigoplus_{\Sigma_r}^{n+2} T$ for some $n > 0$. For $i = 0, \ldots, n$, let $G_{i+1}(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ be the Σ_r-transition guessing realized by $(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)$ in $\mathcal{N}$ (by this, we mean the set of representative $\Sigma_r^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}$-literals $\psi(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ such that $\mathcal{N} \models \psi(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)$). With this choice for the G_i's, the satisfiability of (3.2) in $\mathcal{N}$ guarantees the existence of the path

$$(V_1, G_1) \to (V_2, G_2) \to \cdots \to (V_{n+1}, G_{n+1}) \tag{3.4}$$

from the initial node (V_1, G_1) to the terminal node (V_{n+1}, G_{n+1}) within the safety graph.

Viceversa, suppose that there is a path such as (3.4) and that, by contradiction, the system is safe. In particular, this means that the formula

$$\begin{gathered}\iota(\underline{a}^0) \wedge V_1^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge \cdots \\ \cdots \wedge V_{n+1}^l(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge V_{n+1}^r(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge \upsilon(\underline{a}^{n+1})\end{gathered}$$

is not $(\bigoplus_{\Sigma_r}^{n+2} T)$-satisfiable. If we apply the interpolation Lemma 1.3.10 to the T_0-compatible theories T and $\bigoplus_{\Sigma_r}^{n+1} T$ (the hypotheses of Lemma 1.3.10 hold by the modularity Lemma 1.3.11), we get a ground $\Sigma_r^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}$-sentence $\psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ such that

$$T \models \iota(\underline{a}^0) \wedge V_1^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \to \psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0) \tag{3.5}$$

and such that

$$\psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge \cdots \wedge V_{n+1}^l(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge V_{n+1}^r(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge \upsilon(\underline{a}^{n+1}) \tag{3.6}$$

is not $(\bigoplus_{\Sigma_r}^{n+1} T)$-satisfiable. Since $G_1(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ is a transition Σ_r-guessing, G_1 represents a maximal choice of representative $\Sigma_r^{\underline{a}^0,\underline{a}^1,\underline{d}^0}$-literals, hence we must have either $T \models G_1 \to \psi_1$ or $T \models G_1 \to \neg\psi_1$ (that is, $T \models \psi_1 \to \neg G$). The latter contradicts (3.5) and the fact that the node (V_1, G_1) is initial in the safety graph. The former, together with (3.6) implies that the formula

$$G_1(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge \cdots \wedge V_{n+1}^l(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge V_{n+1}^r(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge \upsilon(\underline{a}^{n+1})$$

is not $(\bigoplus_{\Sigma_r}^{n+1} T)$-satisfiable. We now repeat the argument: we apply the interpolation

Lemma 1.3.10 to the T_0-compatible theories T and $\bigoplus^n_{\Sigma_r} T$ and we get a ground $\Sigma_r^{\underline{c},\underline{a}^1,\underline{a}^2,\underline{d}^1}$-sentence $\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)$ such that

$$T \models G_1(\underline{a}^0,\underline{a}^1,\underline{d}^0) \wedge V_1^r(\underline{a}^0,\underline{a}^1,\underline{d}^0) \wedge V_2^l(\underline{a}^1,\underline{a}^2,\underline{d}^1) \rightarrow \psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1) \tag{3.7}$$

and such that

$$\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1) \wedge V_2^r(\underline{a}^1,\underline{a}^2,\underline{d}^1) \wedge \cdots \wedge V_{n+1}^l(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n) \wedge V_{n+1}^r(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n) \wedge \upsilon(\underline{a}^{n+1}) \tag{3.8}$$

is not $(\bigoplus^n_{\Sigma_r} T)$-satisfiable. Since $G_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)$ is a transition Σ_r-guessing, we must have that either $T \models G_2 \rightarrow \psi_2$ or $T \models G_2 \rightarrow \neg\psi_2$. The latter contradicts (3.7) and the existence of an edge $(V_1,G_1) \rightarrow (V_2,G_2)$. The former, together with (3.8) implies that the formula

$$G_2(\underline{a}^1,\underline{a}^2,\underline{d}^1) \wedge V_2^r(\underline{a}^1,\underline{a}^2,\underline{d}^1) \wedge \cdots \wedge V_{n+1}^l(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n) \wedge V_{n+1}^r(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n) \wedge \upsilon(\underline{a}^{n+1})$$

is not $(\bigoplus^n_{\Sigma_r} T)$-satisfiable. Continuing in this way, we obtain the T-unsatisfiability of the formula

$$G_{n+1}(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n) \wedge V_{n+1}^r(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n) \wedge \upsilon(\underline{a}^{n+1})$$

thus contradicting the fact that the node (V_{n+1},G_{n+1}) is final in the safety graph. □

Theorem 3.3.5. *The ground safety model checking problem for a locally finite compatible LTL-system specification is decidable.*

For complexity, the same remarks given at the end of Subsection 2.3.3 in Chapter 2 apply here too.

3.3.2 Model Checking

This section extend the results of the previous section by showing that the model checking problem is indeed decidable for locally finite compatible LTL-system specifications. To this aim we simply enrich the safety graph with Hintikka sets in order to check whether a run satisfies the property expressed by an LTL$(\Sigma^{\underline{a},\underline{c}})$-sentence φ. The proofs are quite similar to the ones in the previous section; an additional argument based on compactness is however needed to obtain the result.

We briefly recall some notational conventions from the previous section. The

formula $\varphi^i(\underline{a}^i)$ is obtained from the formula $\varphi(\underline{a})$ by replacing each flexible symbol $s \in \Sigma \setminus \Sigma_r$ with s^i (the system variables $\underline{a}$ are similarly renamed as $\underline{a}^i$). Analogously, $\delta^{i,i+1}(\underline{a}^i, \underline{a}^{i+1})$ is obtained by replacing in $\delta^{0,1}(\underline{a}^0, \underline{a}^1)$ the copy s^0 and s^1 of each flexible symbol $s \in \Sigma \setminus \Sigma_r$ with s^i and s^{i+1} respectively (again, the two copies $\underline{a}^0, \underline{a}^1$ of the system variables $\underline{a}$ are similarly renamed as $\underline{a}^i, \underline{a}^{i+1}$). The notational convention applies also to set of literals (meaning that, for example, $V^{i,i+1}(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)$ is obtained by replacing each literal $\ell^{0,1}(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ occurring in $V^{0,1}(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ with $\ell^{i,i+1}(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)$). For the sake of readability, we will usually omit the superscripts of formulae and sets of formulae (but we maintain those of the system variables).

We are now ready to introduce our main definition:

Definition 3.3.6. The *LTL($\Sigma^{\underline{a},\underline{c}}$)-graph* for the ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence φ and associated to the LTL-system specification $(\mathcal{T}, \delta, \iota)$ based on the locally finite compatible data-flow theory $\mathcal{T}$ is the directed graph defined as follows:

- the nodes are the pairs (H, V, G) where H is a Hintikka set for φ, V is a $\hat{\delta}$-assignment and G is a transition Σ_r-guessing;
- there is an edge $(H_1, V_1, G_1) \to (H_2, V_2, G_2)$ iff
  (i) the ground sentence
  $$G_1(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_2^l(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge G_2(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge Lt(H_2(\underline{a}^1)) \tag{3.9}$$
  is T-consistent, where $Lt(H(\underline{a}))$ is the set of $\Sigma^{\underline{a},\underline{c}}$-literals occurring in $H(\underline{a})$;
  (ii) $H_2 \supseteq \{\varphi \mid X\varphi \in H_1\}$.

The *initial nodes* of the graph are the nodes (H, V, G) such that $\varphi \in H$ and $\iota(\underline{a}^0) \wedge Lt(H(\underline{a}^0)) \wedge V^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ is T-consistent.

Recalling Definition 2.3.9 of Hintikka graph and 2.3.10 of strongly connected subgraph (scs), we introduce the following

Definition 3.3.7. An scs $\mathcal{C}$ of the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ is *fulfilling* if and only if the set of nodes $\{H_i \mid (H_i, V_i, G_i) \in \mathcal{C}\}$ is a fulfilling scs of the Hintikka graph $\mathcal{H}(\varphi)$ of φ.

Lemma 3.3.8 (Completeness)**.** *Let φ be a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence and $(\mathcal{T}, \delta, \iota)$ be an LTL-system specification. If there is a run $\mathcal{M}$ for $(\mathcal{T}, \delta, \iota)$ such that $\mathcal{M} \models \varphi$*

then there exists a path without repeated nodes into the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ leading from an initial node into a fulfilling scs.

Proof. Let $\mathcal{M}$ be a run for $(\mathcal{T}, \delta, \iota)$ such that $\mathcal{M} \models \varphi$; without loss of generality, we can consider $\mathcal{M}$ a run for $\langle \mathcal{T}, \hat{\delta}, \iota \rangle$ by appropriately interpreting the fresh constants introduced by the purification of δ. Let us define the following sets

- $H_i = \{\psi \in cl(\varphi) \mid \mathcal{M} \models_i \psi\}$;
- $V_i = \{A_j \in At(\hat{\delta}) \mid \mathcal{M}_i \oplus_{\Sigma_r} \mathcal{M}_{i+1} \models A_j\} \cup \{\neg A_j \mid A_j \in At(\hat{\delta}) \text{ and } \mathcal{M}_i \oplus_{\Sigma_r} \mathcal{M}_{i+1} \models \neg A_j\}$;
- $G_i = \{\ell \mid \ell \text{ is a } \Sigma_r^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}\text{-literal and } \mathcal{M}_i \oplus_{\Sigma_r} \mathcal{M}_{i+1} \models \ell\}$.

Consider the sequence $\{N_i\}$ where $N_i = (H_i, V_i, G_i)$. We want to show that $N_0 \to N_1 \to \dots$ is a path in the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ. First of all, we show that the LTL($\Sigma^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{a}^2,\underline{d}^0,\underline{d}^1}$)-sentence

$$G_i(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_i^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_{i+1}^l(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge G_{i+1}(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge Lt(H_{i+1}(\underline{a}^1)) \tag{3.10}$$

is T-satisfiable for every $i \in \mathbb{N}$. By definition of H_i, V_i, and G_i it follows that

$$\mathcal{M}_i \underset{\Sigma_r}{\oplus} \mathcal{M}_{i+1} \underset{\Sigma_r}{\oplus} \mathcal{M}_{i+2} \models (3.10).$$

Let $\mathcal{N} = (\mathcal{M}_i \oplus_{\Sigma_r} \mathcal{M}_{i+1} \oplus_{\Sigma_r} \mathcal{M}_{i+2})_{|\Sigma_{i+1}^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{a}^2,\underline{d}^0,\underline{d}^1}}$, where $\Sigma_{i+1} = \{s^{i+1} \mid s \in \Sigma \setminus \Sigma_r\} \cup \Sigma_r$; since, with a little abuse of notation, $\mathcal{N}$ is a $\Sigma^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{a}^2,\underline{d}^0,\underline{d}^1}$-structure that is a model for T and that verifies (3.10), it follows that (3.10) is T-satisfiable. Secondly, $X\psi \in H_i$ iff $\mathcal{M} \models_i X\psi$ iff $\mathcal{M} \models_{i+1} \psi$ iff $\psi \in H_{i+1}$, thus $H_{i+1} \supseteq \{\psi \mid X\psi \in H_i\}$ for each i.

We show that $N_0 = (H_0, V_0, G_0)$ is an initial node. In fact, $\varphi \in H_0$ because $\mathcal{M} \models_0 \varphi$ and, obviously, $\varphi \in cl(\varphi)$. Moreover, $\iota(\underline{a}^0) \wedge Lt(H_0(\underline{a}^0)) \wedge V_0^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ is T-consistent. In fact, by construction, $\mathcal{M}_0 \oplus_{\Sigma_r} \mathcal{M}_1 \models \iota(\underline{a}^0) \wedge Lt(H_0(\underline{a}^0)) \wedge V_0^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0)$. Again, let $\mathcal{N} = (\mathcal{M}_0 \oplus_{\Sigma_r} \mathcal{M}_1)_{|\Sigma_0^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}}$, where $\Sigma_0 = \{s^{i+1} \mid s \in \Sigma \setminus \Sigma_r\} \cup \Sigma_r$; since $\mathcal{N}$ is a $\Sigma^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}$-structure that is a model for T and that verifies the constraint we are considering, it follows that the latter is T-satisfiable.

Since the nodes of the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ are in a finite number, there exists a node $N_k = (H_k, V_k, G_k)$ that occurs infinitely often in the path $N_0 \to N_1 \to \dots$ we

are considering. Notice that $\mathcal{C} = \{N_k, N_{k+1}, \ldots\}$ is an scs in LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ because the node N_k occurs infinitely often. Moreover, $\mathcal{C}_{\mathcal{H}(\varphi)} = \{H_i \mid (H_i, V_i, G_i) \in \mathcal{C}\}$ is an scs of $\mathcal{H}(\varphi)$ because $H_k \rightarrow H_{k+1} \rightarrow \cdots$ is a path in $\mathcal{H}(\varphi)$ (by our definition of edge between N_k and N_{k+1}) and because H_k occurs infinitely often. Finally, $\mathcal{C}_{\mathcal{H}(\varphi)}$ is fulfilling because if $\psi_1 U \psi_2 \in cl(\varphi)$, then either $\mathcal{M} \not\models_k \psi_1 U \psi_2$, or there exists $j \geq k$ such that $\mathcal{M} \models_j \psi_2$, i.e. such that $\psi_2 \in H_j$.

The path $N_0 \rightarrow \cdots \rightarrow N_k$ is therefore a path into the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ leading from an initial node into a fulfilling scs; finally, it can be easily turned into a path without repeated nodes simply by discarding all the nodes between two repeated nodes (notice that the path so obtained is still a path in in the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ). □

Lemma 3.3.9 (Soundness)**.** *Let φ be a ground LTL($\Sigma^{\underline{a},\underline{c}}$)-sentence and $(\mathcal{T}, \delta, \iota)$ be an LTL-system specification. If there exists a path without repeated nodes into the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph for φ leading from an initial node into a fulfilling scs, then there is a run $\mathcal{M}$ for $(\mathcal{T}, \delta, \iota)$ such that $\mathcal{M} \models \varphi$.*

Proof. Let $N_0 \rightarrow \cdots \rightarrow N_k$ be the path from an initial node into a fulfilling scs $\mathcal{C}$; moreover, let $N_k \rightarrow \cdots \rightarrow N_{k+s}$ a path covering (possibly with repetitions) all the nodes in $\mathcal{C}$. Consider the path

$$N_0 \rightarrow \cdots \rightarrow N_k \rightarrow \cdots \rightarrow N_{k+s} \rightarrow \cdots \rightarrow N_n \rightarrow \cdots$$

within the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph obtained by cyclically repeating $N_k, \ldots, N_{k+s}$ in the tail (that is, we take, for $i > k + s$, the node N_i to be N_{k+p}, where p is the reminder of the integer division between $i - k$ and $s + 1$).

Let us consider the following set of formulae

$$\Theta_i = T^i \cup Lt(H_i(\underline{a}^i)) \cup G_i(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i) \cup V_i(\underline{a}^i, \underline{a}^{i+1}, \underline{d}^i)$$

where $T^i = \{\psi^i \mid \psi \in T\}$.

We want to prove that $\Theta = \{\iota(\underline{a}^0)\} \cup \bigcup_i \Theta_i$ is consistent. By contradiction, suppose not; hence, by compactness for first-order logic, there exists a finite subset

of Θ which is inconsistent. This implies that there exists n such that the formula

$$\begin{aligned}
&\iota(\underline{a}^0) \wedge Lt(H_0(\underline{a}^0)) \wedge V_0^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0) \quad \wedge \\
\wedge \quad & V_0^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^l(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge G_1(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge Lt(H_1(\underline{a}^1)) \quad \wedge \\
\wedge \quad & V_1^r(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge V_2^l(\underline{a}^2, \underline{a}^3, \underline{d}^2) \wedge G_2(\underline{a}^2, \underline{a}^3, \underline{d}^2) \wedge Lt(H_2(\underline{a}^2)) \quad \wedge \\
& \qquad \cdots \\
\wedge \quad & V_{n-1}^r(\underline{a}^{n-1}, \underline{a}^n, \underline{d}^{n-1}) \wedge \; V_n^l(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge G_n(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge Lt(H_n(\underline{a}^n))
\end{aligned}$$

is $(\bigoplus_{\Sigma_r}^{n+1} T)$-unsatisfiable. Since N_0 is initial, the formula

$$\iota(\underline{a}^0) \wedge Lt(H_0(\underline{a}^0)) \wedge V_0^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0)$$

is T-satisfiable. If we apply the interpolation Lemma 1.3.10 to the T_0-compatible theories T and $\bigoplus_{\Sigma_r}^{n+1} T$ (the hypotheses of Lemma 1.3.10 hold by the modularity Lemma 1.3.11), we get a ground $\Sigma_r^{\underline{c}, \underline{a}^0, \underline{a}^1, \underline{d}^0}$-sentence $\psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ such that

$$T \models \iota(\underline{a}^0) \wedge Lt(H_0(\underline{a}^0)) \wedge V_0^l(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0) \rightarrow \psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0)$$

and such that

$$\begin{aligned}
&\psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_0^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^l(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge G_1(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge Lt(H_1(\underline{a}^1)) \quad \wedge \\
& \qquad \wedge \quad V_1^r(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge V_2^l(\underline{a}^2, \underline{a}^3, \underline{d}^2) \wedge G_2(\underline{a}^2, \underline{a}^3, \underline{d}^2) \wedge Lt(H_2(\underline{a}^2)) \quad \wedge \\
& \qquad\qquad \cdots \\
& \qquad \wedge \quad V_{n-1}^r(\underline{a}^{n-1}, \underline{a}^n, \underline{d}^{n-1}) \wedge \; V_n^l(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge G_n(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge Lt(H_n(\underline{a}^n))
\end{aligned}$$

is $(\bigoplus_{\Sigma_r}^{n} T)$-unsatisfiable. Being $\psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ a ground $\Sigma_r^{\underline{c}, \underline{a}^0, \underline{a}^1, \underline{d}^0}$-sentence and representing $G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ a maximal choice of representative ground $\Sigma_r^{\underline{c}, \underline{a}^0, \underline{a}^1, \underline{d}^0}$-literals, it follows that $G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0) \rightarrow \psi_1(\underline{a}^0, \underline{a}^1, \underline{d}^0)$, hence

$$\begin{aligned}
&G_0(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_0^r(\underline{a}^0, \underline{a}^1, \underline{d}^0) \wedge V_1^l(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge G_1(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge Lt(H_1(\underline{a}^1)) \quad \wedge \\
& \qquad \wedge \quad V_1^r(\underline{a}^1, \underline{a}^2, \underline{d}^1) \wedge V_2^l(\underline{a}^2, \underline{a}^3, \underline{d}^2) \wedge G_2(\underline{a}^2, \underline{a}^3, \underline{d}^2) \wedge Lt(H_2(\underline{a}^2)) \quad \wedge \\
& \qquad\qquad \cdots \\
& \qquad \wedge \quad V_{n-1}^r(\underline{a}^{n-1}, \underline{a}^n, \underline{d}^{n-1}) \wedge \; V_n^l(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge G_n(\underline{a}^n, \underline{a}^{n+1}, \underline{d}^n) \wedge Lt(H_n(\underline{a}^n))
\end{aligned}$$

is $(\bigoplus_{\Sigma_r}^{n} T)$-unsatisfiable. Being the nodes N_0 and N_1 connected in the LTL$(\Sigma^{\underline{a}, \underline{c}})$-

graph, it follows that

$$G_0(\underline{a}^0,\underline{a}^1,\underline{d}^0)\wedge V_0^r(\underline{a}^0,\underline{a}^1,\underline{d}^0)\wedge V_1^l(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge G_1(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge Lt(H_1(\underline{a}^1))$$

is T-satisfiable. Again, if we apply the interpolation Lemma 1.3.10 to the T_0-compatible theories T and $\bigoplus_{\Sigma_r}^n T$ we we get a ground $\Sigma_r^{\underline{c},\underline{a}^0,\underline{a}^1,\underline{d}^0}$-sentence $\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)$ such that

$$T\models G_0(\underline{a}^0,\underline{a}^1,\underline{d}^0)\wedge V_0^r(\underline{a}^0,\underline{a}^1,\underline{d}^0)\wedge V_1^l(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge G_1(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge Lt(H_1(\underline{a}^1))\rightarrow \\ \rightarrow\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)$$

and such that

$$\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge V_1^r(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge V_2^l(\underline{a}^2,\underline{a}^3,\underline{d}^2)\wedge G_2(\underline{a}^2,\underline{a}^3,\underline{d}^2)\wedge Lt(H_2(\underline{a}^2))\quad\wedge \\ \dots \\ \wedge\quad V_{n-1}^r(\underline{a}^{n-1},\underline{a}^n,\underline{d}^{n-1})\wedge\ V_n^l(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n)\wedge G_n(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n)\wedge Lt(H_n(\underline{a}^n))$$

is $(\bigoplus_{\Sigma_r}^{n-1}T)$-unsatisfiable. Being $\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)$ a ground $\Sigma_r^{\underline{c},\underline{a}^1,\underline{a}^2,\underline{d}^1}$-sentence and representing $G_1(\underline{a}^1,\underline{a}^2,\underline{d}^1)$ a maximal choice of representative ground $\Sigma_r^{\underline{c},\underline{a}^1,\underline{a}^2,\underline{d}^1}$-literals, it follows that $G_1(\underline{a}^1,\underline{a}^2,\underline{d}^1)\rightarrow\psi_2(\underline{a}^1,\underline{a}^2,\underline{d}^1)$, hence

$$G_1(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge V_1^r(\underline{a}^1,\underline{a}^2,\underline{d}^1)\wedge V_2^l(\underline{a}^2,\underline{a}^3,\underline{d}^2)\wedge G_2(\underline{a}^2,\underline{a}^3,\underline{d}^2)\wedge Lt(H_2(\underline{a}^2))\quad\wedge \\ \dots \\ \wedge\quad V_{n-1}^r(\underline{a}^{n-1},\underline{a}^n,\underline{d}^{n-1})\wedge\ V_n^l(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n)\wedge G_n(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n)\wedge Lt(H_n(\underline{a}^n))$$

is $(\bigoplus_{\Sigma_r}^{n-1}T)$-unsatisfiable. By repeatedly applying the above argument, we obtain that the formula

$$G_{n-1}(\underline{a}^{n-1},\underline{a}^n,\underline{d}^{n-1})\wedge V_{n-1}^r(\underline{a}^{n-1},\underline{a}^n,\underline{d}^{n-1})\wedge \\ \wedge\ V_n^l(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n)\wedge G_n(\underline{a}^n,\underline{a}^{n+1},\underline{d}^n)\wedge Lt(H_n(\underline{a}^n))$$

is T-unsatisfiable. But, if $n\leq k+s$, this contradicts the hypothesis that $(H_{n-1},V_{n-1},G_{n-1})\rightarrow(H_n,V_n,G_n)$ is an edge in the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph; if $n>k+s$ it contradicts the hypothesis that $(H_{k+p_1},V_{k+p_1},G_{k+p_1})\rightarrow(H_{k+p_2},V_{k+p_2},G_{k+p_2})$ is an edge in the LTL($\Sigma^{\underline{a},\underline{c}}$)-graph, where p_1 (resp. p_2) is the reminder of the integer

division between $n-1-k$ (resp. $n-k$) and $s+1$.

Thus there exists a structure $\mathcal{M}$ such that $\mathcal{M} \models \Theta$. This structure can be seen as an LTL($\Sigma^{\underline{a},\underline{d},\underline{c}}$)-structure $\mathcal{M} = \{\mathcal{M}_n = (M, \mathcal{I}_n)\}_{n\in\mathbb{N}}$ appropriate for the data-flow theory $\mathcal{T}$, such that $\mathcal{M}_n \oplus_{\Sigma_r} \mathcal{M}_{n+1} \models \hat{\delta}(\underline{a}^0, \underline{a}^1, \underline{d}^0)$ (hence $\mathcal{M}_n \oplus_{\Sigma_r} \mathcal{M}_{n+1} \models \delta(\underline{a}^0, \underline{a}^1)$) for every $n \geq 0$, and such that $\mathcal{M}_0 \models \iota(\underline{a})$.

It remains to prove that $\mathcal{M} \models \varphi$, i.e. we prove by induction on the complexity of ψ (where $\psi \in cl(\varphi)$) that for every i it holds that:

$$\psi \in H_i \quad \Rightarrow \quad \mathcal{M} \models_i \psi \tag{3.11}$$

where H_i is the first component of the node $N_i = (H_i, V_i, G_i)$ in the (infinite) path we are considering. In particular, we get $\mathcal{M} \models_0 \varphi$, because $\varphi \in H_0$ (since H_0 is initial). The condition (3.11) is obvious if ψ is a literal or if it is of the kind $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$ (by definition of Hintikka set and since, by construction, $\mathcal{M}_i \models Lt(H_i(\underline{a}))$).

If ψ is of the kind $X\psi_1$, then $X\psi_1 \in H_i$ implies that $\psi_1 \in H_{i+1}$, so it follows that $\mathcal{M} \models_{i+1} \psi_1$ by induction hypothesis, and thus $\mathcal{M} \models_i X\psi_1$ obtains. If ψ is of the kind $\Box\psi_1$, then $\Box\psi_1 \in H_i$ implies $\psi_1 \in H_j$ for each $j \geq i$, so it follows that $\mathcal{M} \models_j \psi_1$ for each $j \geq i$ by induction hypothesis, and thus $\mathcal{M} \models_i \Box\psi_1$.

Suppose now ψ is of the kind $\psi_1 U \psi_2$. Let us consider the following two cases:

- If $i < k$ there are two subcases to consider: (i) $\psi_1 U \psi_2 \in H_k$ and $\psi_1 \in H_j$ for every $i \leq j < k$; (ii) there exists $l < k$ such that $\psi_2 \in H_l$ and $\psi_1 \in H_j$ for every $i \leq j < l$. For the case (i) we can conclude that $\mathcal{M} \models_i \psi_1 U \psi_2$ by induction hypothesis and by the fact that $\mathcal{M} \models_k \psi_1 U \psi_2$ (see the case $i \geq k$ below), whereas for (ii) we can conclude by induction hypothesis that $\mathcal{M} \models_i \psi_1 U \psi_2$;

- Let $\mathcal{C}_{\mathcal{H}(\varphi)} = \{H_i \mid (H_i, V_i, G_i) \in \mathcal{C}\}$ (we remind that $\mathcal{C}_{\mathcal{H}(\varphi)}$ is a fulfilling scs of $\mathcal{H}(\varphi)$ by hypothesis). If $i \geq k$, since $\psi_1 U \psi_2 \in H_i$ and since the scs $\mathcal{C}_{\mathcal{H}(\varphi)}$ is fulfilling, there exists $H \in \mathcal{C}_{\mathcal{H}(\varphi)}$ such that $\psi_2 \in H$.[6] Such an H occurs in the infinite list $H_i, H_{i+1}, \ldots$, because this list includes all the nodes from $\mathcal{C}_{\mathcal{H}(\varphi)}$. Thus there exists the minimum $j \geq i$ such that $\psi_2 \in H_j$; for this j, the definition of a Hintikka set and of an edge in the Hintikka graph gives $\psi_1 \in H_l$ for every $i \leq l < j$, thus by induction hypothesis $\mathcal{M} \models_i \psi_1 U \psi_2$ obtains.

[6] This is by the definition of a Hintikka set and of an edge in the graph $\mathcal{H}(\varphi)$: notice that $\psi_1 U \psi_2$ is inherited by all the nodes of a path within $\mathcal{H}(\varphi)$ starting with H_i, unless the path meets a node to which ψ_2 belongs. Now a path covering the whole $\mathcal{C}_{\mathcal{H}(\varphi)}$ must meets such a node, because $\mathcal{C}_{\mathcal{H}(\varphi)}$ is fulfilling.

□

As an immediate corollary of the Lemmas 3.3.8 and 3.3.9 we obtain the following

Theorem 3.3.10. *The ground model checking problem for a locally finite compatible LTL-system specification is decidable.*

3.4 Some Examples

In this section, we provide examples to which the algorithm suggested by Proposition 3.3.4 can be successfully applied in order to formally verify safety properties. For the convenience of the reader, we recall the axioms of the theory T_{dlo} of dense linear order since the examples below rely on suitable extensions of it (here and in the following $x < y$ stands for $x \leq y \wedge x \neq y$)

$$\forall x \forall y \forall z \, (x \leq y \wedge y \leq z \rightarrow x \leq y)$$
$$\forall x \forall y \, (x \leq y \vee y \leq x)$$
$$\forall x \forall y \, (x \leq y \wedge y \leq x \rightarrow x = y)$$
$$\forall x \forall y \, (x < y \rightarrow \exists z \, (x < z \wedge z < y))$$

Example 3.4.1 (Sofronie-Stokkermans, 2006)**.** Consider a water level controller modeled as follows:

- changes in the water level by inflow/outflow are represented as functions *in* and *out* depending on the water level l and on the time instant; alarm and overflow levels $l_{\text{alarm}} < l_{\text{overflow}}$ are known;
- if the water level l is such that $l \geq l_{\text{alarm}}$ at a given state, then a valve is opened and the water level changes at the next observable time by $l' = in(out(l))$;
- if $l < l_{\text{alarm}}$ then the valve is closed; the water level changes at the next observable time by $l' = in(l)$.

The dependency of the functions *in* and *out* on the time instant means precisely that they can be modeled as *flexible* function symbols depending only on the water level. However, functions *in* and *out* cannot be completely uninterpreted, we impose

the following restrictions on them:

$$\forall x\,(x < l_{\text{alarm}} \rightarrow in(x) < l_{\text{overflow}}) \tag{3.12}$$

$$\forall x\,(x < l_{\text{overflow}} \rightarrow out(x) < l_{\text{alarm}}) \tag{3.13}$$

Under such restrictions we want to show that from an initial state where $l < l_{\text{alarm}}$ the water level always remains below l_{overflow}.

Let us fix the notation in order to formalize the problem in our framework. We consider the data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ such that

- $\Sigma = \{in, out, l_{\text{alarm}}, l_{\text{overflow}}, <\}$ where in, out are two unary function symbols, $l_{\text{alarm}}, l_{\text{overflow}}$ are two constant symbols, $<$ is a binary predicate symbol;
- $\Sigma_r = \{l_{\text{alarm}}, l_{\text{overflow}}, <\}$;
- $T = T_r^\star \cup \{(3.12), (3.13)\}$ where $T_r^\star$ is the theory of dense linear orders without endpoints endowed with the further axiom $l_{\text{alarm}} < l_{\text{overflow}}$. In other words, $T_r^\star$ is made of the axioms of T_{dlo} and of the following axioms:

$$\forall x \exists y\; x < y$$
$$\forall x \exists y\; y < x$$
$$l_{\text{alarm}} < l_{\text{overflow}}$$

- l is the only system variable and there are no system parameters (that is, $\underline{a} := \{l\}$ and $\underline{c} := \emptyset$).

It can be shown that the constraint satisfiability problem for T is decidable, that $T_r^\star$ admits quantifier elimination (thus it is the model completion of its universal fragment T_r), and that T_r is effectively locally finite: hence it follows that $\mathcal{T}$ is a locally finite compatible data-flow theory. We consider now the LTL-system specification $(\mathcal{T}, \delta, \iota)$ where δ is

$$\begin{aligned} \delta \quad :\equiv \quad & \big(l_{\text{alarm}} \leq l^0 \rightarrow l^1 = in^0(out^0(l^0))\big) \quad \wedge \\ \wedge \quad & \big(l^0 < l_{\text{alarm}} \rightarrow l^1 = in^0(l^0)\big) \end{aligned}$$

and ι is $l < l_{\text{alarm}}$. Finally, notice that δ is a purely left $(\Sigma^{\underline{a}} \oplus_{\Sigma_r} \Sigma^{\underline{a}})$-formula.

We are interested in the safety model checking problem in which the unsafe state is described by the formula υ given by $l_{\text{overflow}} < l$. Using the procedure suggested

by Theorem 3.3.5 we can prove that the system is safe, i.e. that there is no run $\mathcal{M}$ for $(\mathcal{T}, \delta, \iota)$ such that $\mathcal{M} \models \Diamond v$. We can observe that the task in practice is not extremely hard from a computational point of view, even if, accordingly to Definition 3.3.3, the graph is made of $2^{32} \times 21$ nodes. In fact, since our transition relation δ is a purely left formula, we can consider only T-consistent nodes (i.e, nodes (V, G) such that $V \wedge G$ is T-consistent); indeed, recalling Definition 3.3.3 of safety graph, T-inconsistent nodes (i) cannot be initial nodes and (ii) cannot be reached by any path in the safety graph (it is easy to see that such nodes cannot have any incoming edge). Since there are just 50 nodes (modulo T-equivalence) which are T-consistent, at most 50^2 satisfiability tests are required to check whether a terminal node is reachable from an initial one. Moreover, by using suitable heuristics and strategies, the problem becomes computationally even easier: indeed, instead of considering all the edges of the safety graph, it is sufficient to build just the paths starting from the initial nodes or ending in a terminal node (namely applying a forward/backward search strategy). In the former case, it turns out that 26 nodes (modulo T-equivalence) of the safety graph are reachable from an initial node, none of them being a terminal node. In the latter, just 12 nodes are reachable from a terminal node, obviously none of them being an initial node. Hence the dimension of the problem is tractable (other details can be found in the Appendix).

One might ask if the axioms (3.12) and (3.13) are really needed in order to guarantee the safety of the system, or, instead, if it is sufficient to consider just the instantiations of the two axioms above to the water level at the current time. In such a case, T is simply the theory of dense linear order without endpoints endowed with the axiom $l_{\text{alarm}} < l_{\text{overflow}}$; moreover, we have to insert the instances into the transition in such a way they are always satisfied during the flow of time, thus obtaining the new transition

$$\begin{aligned}
\delta' \quad :\equiv \quad & l_{\text{alarm}} \leq l^0 \rightarrow l^1 = in^0(out^0(l^0)) \quad \wedge \\
\wedge \quad & l^0 < l_{\text{alarm}} \rightarrow l^1 = in^0(l^0) \quad \wedge \\
\wedge \quad & l^0 < l_{\text{alarm}} \rightarrow in^0(l^0) < l_{\text{overflow}} \quad \wedge \\
\wedge \quad & l^0 < l_{\text{overflow}} \rightarrow out^0(l^0) < l_{\text{alarm}}
\end{aligned}$$

In such a system, it is straightforward to see that there is a path into the safety

graph from an initial to a terminal node. Consider for example the following path:

$$(V_0, G_0) \longrightarrow (V_1, G_1)$$

where

$$\begin{aligned} V_0(\underline{a}^0, \underline{a}^1) &:\equiv l^0 < l_{\text{alarm}} \wedge l^0 < l_{\text{overflow}} \wedge l^1 = in^0(out^0(l^0)) \wedge l^1 = in^0(l^0) \wedge \\ &\qquad \wedge in^0(l^0) < l_{\text{overflow}} \wedge out^0(l^0) < l_{\text{alarm}} \\ G_0(\underline{a}^0, \underline{a}^1) &:\equiv l^0 < l_{\text{alarm}} < l^1 < l_{\text{overflow}} \end{aligned}$$

and

$$\begin{aligned} V_1(\underline{a}^0, \underline{a}^1) &:\equiv l_{\text{alarm}} < l^0 \wedge l^0 < l_{\text{overflow}} \wedge l^1 = in^0(out^0(l^0)) \wedge l^1 = in^0(l^0) \wedge \\ &\qquad \wedge \neg(in^0(l^0) < l_{\text{overflow}}) \wedge out^0(l^0) < l_{\text{alarm}} \\ G_1(\underline{a}^0, \underline{a}^1) &:\equiv l_{\text{alarm}} < l^0 < l_{\text{overflow}} < l^1. \end{aligned}$$

It is easy to check that (V_0, G_0) is an initial node and that (V_1, G_1) is a terminal node; moreover $G_0(\underline{a}^0, \underline{a}^1) \wedge V_1(\underline{a}^1, \underline{a}^2) \wedge G_1(\underline{a}^1, \underline{a}^2)$ is T-consistent (when checking details, remember that our transition δ is a purely left formula).

The aim of the following three examples is to use our techniques to analyze the safety of the well-known Lamport's mutual exclusion "Bakery" algorithm. This algorithm can be modeled by a locally finite compatible (and also totally rigid) LTL-system specification in case the number of processors is known (finite state LTL-system specifications are – at least in principle – not enough because the number of tickets is unbounded). Examples 3.4.2 and 3.4.3 give a first formalization which directly fit into our framework in case the number of processor is known; these examples can be useful to have a better insight into Example 3.4.4 that shows how to deal with the case of an unknown number of processor.

Example 3.4.2. Consider the Lamport's mutual exclusion "Bakery" algorithm and fix a number n of processors or individuals. In a first approximation, we consider that every individual is always in the queue, waiting to be served (as soon as an individual is served, it goes back in the last position of the queue).

Let us fix the notation in order to formalize the problem in our framework. We consider the data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ such that

– $\Sigma = \{S, <\}$ where S is a unary predicate symbol and $<$ is a binary predicate

symbol;

– $\Sigma_r = \{<\}$ and T_r is the theory of dense linear order without endpoints; in other words, T_r is made of the axioms of T_{dlo} and of the following axioms:

$$\forall x \exists y\ x < y$$
$$\forall x \exists y\ y < x$$

– $T = T_r$;

– a_i's are the system variables and there are no system parameters (i.e., $\underline{a} = \{a_1, \ldots, a_n\}$, $\underline{c} = \emptyset$).

Intuitively, a_i represents the ticket associated to the i^{th} individual and $S(x)$ formalizes that x is served. It is straightforward to see that the constraint satisfiability problem for T is decidable, T is T_r-compatible (since T_r admits quantifier elimination) and T_r is effectively locally finite, thus $\mathcal{T}$ is a locally finite compatible data-flow theory.

The initial state condition $\iota(\underline{a})$ is in the form

$$a_{i_1} < \cdots < a_{i_n} \wedge S(a_{i_1}) \wedge \neg S(a_{i_2}) \wedge \cdots \wedge \neg S(a_{i_n})$$

where $i_j \in \{1, \ldots, n\}$ and if $j \neq k$ then $i_j \neq i_k$. The unsafe states are described by the formula $\upsilon(\underline{a}) := \bigvee_{i \neq j} \big(S(a_i) \wedge S(a_j)\big)$ which says that at least two individuals are in the critical section at the same time. Finally, the transition relation $\delta(\underline{a}^0, \underline{a}^1)$ is obtained from the conjunction of the following

$$\bigwedge_{i=1}^{n} \Big(S^0(a_i^0) \rightarrow \neg S^1(a_i^1) \wedge \bigwedge_{j=1}^{n} a_j^0 < a_i^1 \Big) \tag{3.14}$$

$$\bigwedge_{i=1}^{n} \Big(\neg S^0(a_i^0) \rightarrow a_i^1 = a_i^0 \Big) \tag{3.15}$$

$$\bigwedge_{i=1}^{n} \Big(\bigvee_{i \neq j} a_j^0 < a_i^0 \rightarrow \neg S^1(a_i^1) \Big) \tag{3.16}$$

whose intuitive meaning is the following

(3.14) if an individual is served, it goes back in the last position of the queue;

(3.15) if an individual is not served, it maintains its position in the queue;

(3.16) if an individual is not first in the queue, is not served.

Example 3.4.3. Consider the Lamport's mutual exclusion "Bakery" algorithm and fix a number n of processors or individuals. This time, individuals are not forced to be always in the queue: more precisely, as soon as one is served, gets out of the queue and can non-deterministically choose when come back in the last position of the queue.

Let us fix the notation in order to formalize the problem in our framework. We consider the data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ such that

- $\Sigma = \{S, <, 0\}$ where S is a unary predicate symbol, $<$ is a binary predicate symbol and 0 is a constant;
- $\Sigma_r = \{<, 0\}$ and T_r is the theory of dense linear orders with a minimum named 0;
- $T = T_r \cup \{\neg S(0)\}$;
- a_i's are the system variables and there are no system parameters (i.e., $\underline{a} = \{a_1, \ldots, a_n\}$, $\underline{c} = \emptyset$).

Intuitively, a_i represents the ticket associated to the i^{th} individual (or being out of the queue if the ticket is equal to 0) and $S(x)$ formalizes that x is served. Again, it is straightforward to see that the constraint satisfiability problem for T is decidable, T is T_r-compatible (since T_r admits quantifier elimination) and T_r is effectively locally finite, thus $\mathcal{T}$ is a locally finite compatible data-flow theory.

The initial state condition is a formula of the kind

$$a_1 = 0 \wedge \cdots \wedge a_n = 0.$$

The unsafe states are described by the formula $\upsilon(\underline{a}) := \bigvee_{i \neq j} \big(S(a_i) \wedge S(a_j)\big)$ which says that at least two individuals are in the critical section at the same time. Finally, the transition relation $\delta(\underline{a}^0, \underline{a}^1)$ is obtained from the conjunction of the following

$$\bigwedge_{i=1}^{n} \Big(S^0(a_i^0) \rightarrow a_i^1 = 0 \Big) \tag{3.17}$$

$$\bigwedge_{i=1}^{n} \Big(\neg S^0(a_i^0) \wedge a_i^0 \neq 0 \rightarrow a_i^1 = a_i^0 \Big) \tag{3.18}$$

$$\bigwedge_{i=1}^{n}\Big(\bigvee_{i\neq j}(a_j^0 < a_i^0 \wedge a_j^0 \neq 0) \rightarrow \neg S^1(a_i^1)\Big) \tag{3.19}$$

$$\bigwedge_{i=1}^{n}\Big(a_i^0 = 0 \rightarrow a_i^1 = 0 \vee \bigwedge_{j\neq i} a_j^0 < a_i^1\Big) \tag{3.20}$$

$$\bigwedge_{i\neq j=1}^{n}\Big(a_i^0 = a_j^0 \wedge a_i^0 = 0 \rightarrow a_i^1 = 0 \vee a_j^1 = 0 \vee a_i^1 \neq a_j^1\Big) \tag{3.21}$$

whose meaning is the following

(3.17) if an individual is served, it gets rid of the ticket (thus quits the queue);

(3.18) if an individual is not served while is in the queue, the ticket is preserved;

(3.19) if an individual is not first in the queue, is not served;

(3.20) if an individual is out of the queue, it non-deterministically choose to remain out of the queue or to join the queue in the last position.

(3.21) if two individuals are out of the queue, then either one of them remains out of the queue or they will assigned different tickets ((3.20) ensures that they will join the queue in the last positions).

Example 3.4.4. In case the number of involved processors is unknown, we can build for the problem an appropriate $\mathcal{T}$, which is 'almost' a locally finite compatible (not totally rigid anymore) LTL-system specification. We said 'almost' because $\mathcal{T}$ violates our Assumption 3.1.9 from Subsection 3.1.2 in that it has a non-ground transition (some first-order variables are universally quantified in it). We then produce out of $\mathcal{T}$ (by skolemization and instantiation) a locally finite compatible LTL-system specification $\mathcal{T}'$ which is safe iff $\mathcal{T}$ is safe. Safety of $\mathcal{T}'$ can then be easily checked through our techniques. Before analyzing formal details, we point out that the peculiar features of $\mathcal{T}$ that make the whole construction to work are *purely syntactic* in nature and do not need human intervention to be noticed: they basically consist of the finiteness of the set of terms of certain sorts in the skolemized Herbrand universe.

We deal with a sorted language:[7] indeed, we have two sorts, namely P and O. The former is the sort representing the individuals (i.e the involved processes),

[7]There are no problems in extending our results to the many-sorted case.

whereas the latter is used in order to represent tickets. Let us consider the following data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$:

- Σ is a sorted signature containing a unary predicate symbol S of sort P, a binary predicate symbol $<: O \times O$, two constant symbols 0 and 1 of sort O, and a unary function symbol $f : P \to O$;
- T axiomatizes, over the sort O, the theory of dense total orders with named distinct endpoints; in other words, T is made of the axioms of T_{dlo} and of the following axioms

$$\forall x\,(0 \leq x)$$
$$\forall x\,(x \leq 1)$$
$$0 < 1$$

  Moreover, the behavior of the function f is constrained by the following further axioms for T:

$$\forall x \forall y\,(f(x) = f(y) \to x = y \vee f(x) = 1) \qquad (f \text{ is "almost-injective"})$$
$$\forall x\,(f(x) = 1 \to \neg S(x))$$

- Σ_r contains the symbols $\{0, 1, <\}$;
- there are no system parameters (i.e. $\underline{c} := \emptyset$) and there is just one system variable t, which is of sort O (i.e. $\underline{a} := \{t\}$).

In order to give an intuitive explanation of what we are modeling, we can think of the values of t at two consecutive instants as the range in which the values of the tickets produced by the "ticket machine" in that interval of time can vary, whereas f can be seen as the function that associates to every individual its current ticket (f is time-dependent, hence flexible, because the ticket is changed after it has been used). We have at our disposal an infinite amounts of tickets whose values are in the interval $[0, 1]$; every individual is inserted into a queue according to the value of its ticket (the value 1 has the meaning of being out of the queue). Finally, the predicate S models the set of the individuals that are in the critical section.

We leave the reader to check that the constraint satisfiability problem for T is decidable and that T is T_r-compatible for a suitable universal locally finite Σ_r-theory

T_r:[8] it follows that $\mathcal{T}$ is a locally finite compatible data-flow theory.

We can associate to $\mathcal{T}$ an LTL-system specification $(\mathcal{T}, \delta, \iota)$ in the following manner: the initial condition is described by the formula

$$\iota \ :\equiv \ \forall x\,(f(x) = 1) \wedge t = 0,$$

whereas the transition δ is obtained from the conjunction of the following (implicitly universally quantified) formulae:

$$t^0 < t^1 < 1 \tag{3.22}$$

$$S^0(x) \to f^1(x) = 1 \tag{3.23}$$

$$\neg S^0(x) \wedge f^0(x) \neq 1 \to f^1(x) = f^0(x) \tag{3.24}$$

$$f^0(x) < f^0(y) \to \neg S^1(y) \tag{3.25}$$

$$f^0(x) = 1 \to f^1(x) = 1 \vee (t^0 \leq f^1(x) \wedge f^1(x) < t^1 \wedge \neg S^1(x)) \tag{3.26}$$

The meaning of the above formulae is the following:

(3.22) the range of the values of the tickets produced by the "ticket machine" is strictly increasing during the flow of time;

(3.23) an individual is removed from the queue immediately after having joined the critical section;

(3.24) if an individual is in the queue and it is not in the critical section, then its ticket is preserved;

(3.25) if an individual is not the first in the queue, it cannot enter the critical section;

(3.26) if an individual is not in the queue, it can remain out of the queue or it can take a ticket (without being immediately served).

The unsafe states are described by the formula

$$\upsilon \ :\equiv \ \exists x \exists y\,(x \neq y \wedge S(x) \wedge S(y)).$$

[8] Take as T_r the theory of linear orders with named distinct endpoints (this admits as a model completion $T_r^\star$, which is the theory of an infinite set over the sort P and of dense linear orders with named distinct endpoints over the sort O).

Since ι, δ, υ all violate our Assumption 3.1.9 because they are not ground, the problem needs to be reformulated (in a *safety/unsafety preserving* way!) in order to become tractable with our techniques.

Consider the data-flow theory $\mathcal{T}' = \langle \Sigma, T, \Sigma_r, \{t\}, \{c_1, c_2\} \rangle$, which is like $\mathcal{T}$ except that two new system parameters c_1, c_2 of sort P have been added. We first skolemize the formula υ into the ground formula

$$\upsilon' \ :\equiv \ c_1 \neq c_2 \wedge S(c_1) \wedge S(c_2),$$

then we instantiate the initial condition ι obtaining

$$\iota' \ :\equiv \ t = 0 \wedge f(c_1) = 1 \wedge f(c_2) = 1.$$

Finally we instantiate also the transition δ, thus getting the ground formula δ' which is the conjunctions of (3.27)-(3.33) below:[9]

$$t^0 < t^1 < 1 \tag{3.27}$$

$$(S^0(c_1) \rightarrow f^1(c_1) = 1) \wedge (S^0(c_2) \rightarrow f^1(c_2) = 1) \tag{3.28}$$

$$\begin{aligned} &(\neg S^0(c_1) \wedge f^0(c_1) \neq 1 \rightarrow f^1(c_1) = f^0(c_1)) \wedge \\ &\wedge (\neg S^0(c_2) \wedge f^0(c_2) \neq 1 \rightarrow f^1(c_2) = f^0(c_2)) \end{aligned} \tag{3.29}$$

$$f^0(c_1) < f^0(c_2) \rightarrow \neg S^1(c_2) \tag{3.30}$$

$$f^0(c_2) < f^0(c_1) \rightarrow \neg S^1(c_1) \tag{3.31}$$

$$f^0(c_1) = 1 \rightarrow f^1(c_1) = 1 \vee (t^0 \leq f^1(c_1) \wedge f^1(c_1) < t^1 \wedge \neg S^1(c_1)) \tag{3.32}$$

$$f^0(c_2) = 1 \rightarrow f^1(c_2) = 1 \vee (t^0 \leq f^1(c_2) \wedge f^1(c_2) < t^1 \wedge \neg S^1(c_2)) \tag{3.33}$$

$(\mathcal{T}', \delta', \iota')$ is now an LTL-system specification matching Assumption 3.1.9; moreover $(\mathcal{T}', \delta', \iota')$ is locally finite compatible for the reasons explained above.

It is not difficult to see that there exists a bad run for $(\mathcal{T}, \iota, \delta)$ (w.r.t. υ) if and only if there exists a bad run for $(\mathcal{T}', \iota', \delta')$ (w.r.t. υ'): the key observation to show this is that one can restrict the interpretation of the sort P in a bad run for $(\mathcal{T}', \iota', \delta')$ so that it consists only on the two individuals c_1, c_2. By applying the algorithm from Proposition 3.3.4, since $\iota' \wedge \upsilon'$ is T-inconsistent and since $\delta' \wedge \upsilon'$

[9] Observe that all quantifiers in ι, δ are of sort P and that there are no ground terms in the signature of $\mathcal{T}'$ of that sort, apart from the Skolem constants c_1, c_2. Notice that some instances of δ have been removed, because they are tautological modulo T.

is $(T \oplus_{\Sigma_r} T)$-inconsistent, it follows that $(\mathcal{T}', \iota', \delta')$ is safe w.r.t. υ': consequently, $(\mathcal{T}, \iota, \delta)$ is safe w.r.t. υ too.

Remark. Suppose that an LTL-system specification $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ satisfies the following requirements:

(i) the initial state condition, the transition relation and the formula representing the safe state are universal closures of open formulae (for the sake of simplicity, we assume that all the quantifiers bind variables over a unique sort, say S);

(ii) the axioms of the theory T are of the kind $\forall \underline{x}^S \varphi$ where φ does not contain any quantifier binding variables over the sort S;

(iii) the terms of sort S contained in the Herbrand universe obtained from $\Sigma^{\underline{a},\underline{c}}$ augmented with the symbols from the Skolemization of the negation of the formula representing the safe states are in a finite number and made of rigid symbols only.

Then the technique used in Example 3.4.4 applies in order to produce out of $\mathcal{T}$ (by skolemization and instantiation) a locally finite compatible LTL-system specification $\mathcal{T}'$ which is safe iff $\mathcal{T}$ is safe. Again, notice that the peculiar features of $\mathcal{T}$ that make the whole construction to work are *purely syntactic* in nature and do not need human intervention to be noticed.

3.5 Conclusions and Related Work

In this chapter, we have studied the decidability of the model checking problem for quantifier-free formulae modulo a background first-order theory axiomatizing the extensional part of the language. We have also recalled the undecidability of the model checking problem by a reduction to the reachability problem of Minsky machines (see Minsky, 1961). Moreover, we have given the decidability of the (ground) model checking problem, when this is restricted to safety properties modulo background theories that are compatible (see Ghilardi, 2004) with a locally finite theory over the rigid signature, and then we have extended this result in order to be able to take into account any temporal property. We have also exemplified our techniques on some examples.

Since the literature on model checking is extremely vast (see the introduction of this thesis to have but a few references), we shall make here a comparison only

with works that are somewhat related to our model-theoretic viewpoint inspired by combination. The paper Demri (2006) makes an extensive review on constrained LTL that can be seen as a form of model checking for possibly infinite-state systems. This form of model checking does not allow flexible symbols (apart from system variables); moreover, specific fixed purely relational structures plays there the role played by the models of the underlying first-order theories in our approach. In this context, *some* of the results in Demri (2006) could be seen as specializations of our results to totally rigid LTL-system specifications. Other results and techniques from Demri (2006) (and also from the recent paper Demri et al., 2006) should nevertheless be seriously taken into account for integration in our settings. A similar observation applies to the rewriting techniques used in Cyrluk and Narendran (1994) in order to obtain decision procedures for interesting (but very special) classes of formulae.

An integration of classic tableaux and automated deduction techniques is presented in Sipma et al. (1999). While sharing the goal of combining model checking algorithms and deductive techniques, Sipma et al. (1999) provides a uniform framework in which performing such combination with no guarantee on the complete automation of the resulting combination. Similarly, Maidl (2001) describes a combination of tableaux and automated deduction techniques to automatically solve the model checking problem of classes of parametrized theories. Although we share some use of tableaux and automated deduction techniques, Maidl (2001) does not reduce the problem to combination problems in first-order theories.

The approach in de Moura et al. (2002) shares an important distinguishing feature with ours, namely the reduction of safety model checking problems to constraint satisfiability modulo first-order theories. Our main contribution (Theorems 3.3.5 and 3.3.10) identifies precise conditions under which this reduction, not necessarily limited to safety properties, yields a complete decision procedure (notice however that our safety graph is not just an approximation of the graph of the states of the system, because *pairs* of states are taken into account when building it).

Finally, a long line of research in model checking infinite-state systems, begun with the seminal work in Graf and Saïdi (1997), goes under the name of "abstract-check-refine", featuring a combination of finite-state model checking and decision procedures for first-order theories. A common feature with our work is the emphasis on using decision procedures for the satisfiability problem in first-order theories. However, we are more concerned with precisely characterizing the termination of the model checking algorithm while the abstract-check-refine techniques focus on practi-

cal usability. Furthermore, for such techniques to scale-up, the decision procedures are required to compute interpolants (see, e.g., Henzinger et al., 2002; McMillan, 2005) and this may be indeed a difficult task. Instead, our approach should allow one to more easily leverage SMT solvers by designing suitable refinements of the algorithm suggested by Proposition 3.3.4 and Lemmas 3.3.8 and 3.3.9.

Chapter 4

Arrays with Dimension

This chapter is devoted to the study of extensions of the theory of arrays in order to derive decidability result for their universal fragments. Since its introduction in McCarthy (1962), the theory of arrays has played a very important role in Computer Science. Hence, it is not surprising that many papers (see, e.g., Armando et al., 2003; Bradley et al., 2006; Downey and Sethi, 1978; Jaffar, 1981; Mateti, 1981; Reynolds, 1979; Stump et al., 2001; Suzuki and Jefferson, 1980) have been devoted to its study in the context of verification and many reasoning techniques, both automatic (see, e.g., Armando et al., 2003) and manual (see, e.g., Reynolds, 1979), have been developed to reason in such a theory.

Unfortunately, as many previous works (see, e.g., Bradley et al., 2006; Jaffar, 1981; Mateti, 1981; Suzuki and Jefferson, 1980) have already observed, the theory of array alone or even extended with extensional equality between arrays (as in Armando et al., 2003; Stump et al., 2001) is not sufficient for many applications of verification. For example, the works in Jaffar (1981); Mateti (1981); Suzuki and Jefferson (1980) tried to extend the theory to reason about sorted arrays. More recently, works in Bradley (2007); Bradley et al. (2006) have shown the decidability of the satisfiability problem for a restricted class of (possibly quantified) first-order formulae that allows one to express many important properties about arrays.

In this chapter we prove decidability results for extensions of the theory of arrays with dimension, being the decidability of the universal fragment of the latter already given in Nicolini (2007). We properly extend that result considering more expressive fragments and taking into account issues related to the implementation of the developed procedures and their integration into tools that are already available.

4.1 Arrays with Dimension

An array is a data structure that consists of a group of elements having a single name. Elements in the array are usually numbered and individual elements are accessed by their index (i.e. numeric position). We consider two main types of arrays which are natively supported by imperative languages (such as C): *fixed-size* and *dynamically-allocated* arrays. A fixed-size array occupies a contiguous area of storage that never changes during run-time and whose fixed dimension is known at compile-time. In contrast, the size of the memory reserved to dynamically-allocated arrays can be unknown at compile-time and may change at runtime, even though this may be an expensive operation involving the copy of the entire content of an array (consider, e.g., the C's function `realloc` applied to a `malloc`'ed array). Actually, there exists a third type of arrays called *dynamic*, which are supported by interpreted (such as, for example, the Perl language) and object-oriented programming languages (see, e.g., the C++'s `std::vector` or the `ArrayList` classes of Java API and .NET Framework) in which memory handling is usually hidden. A detailed discussion of such a data structure is beyond the scope of this chapter. Here, it is sufficient to observe that dynamic arrays can be efficiently implemented by imposing an appropriate memory allocation policy on dynamically-allocated arrays (see, e.g., Brodnik et al., 1999). For all types of arrays, their elements have usually the same type.

After the declaration/allocation, the content of an array is in general not initialized, both in the case of fixed-size or dynamically-allocated arrays (in this context, recall the difference between the C's functions `malloc` and `calloc`). To formalize this, we introduce a distinguished element $\perp$ (for undefined), which is distinct from every other element in arrays, and assume that any array contains $\perp$ at every position except one, after creation. This distinguished position is the capacity of an array a (minus 1, since 0 is used to identify the first element of a), i.e. how many elements a will be able to store. Under this assumption, the situation where a predefined element is used to fill the array after declaration can be simulated by using an appropriate sequence of assignments. In our formal model, we abstract from memory and efficiency issues and assume the capability of storing an element e at an arbitrary index i of an array a, by allocating (only) the necessary extra space when i is bigger than the actual size of a; the resulting array is denoted with $\mathsf{store}(a, i, e)$. In this way, we can formalize the capacity of an array as the function dim returning the smallest index, after which no more elements of the array exist. For simplicity, we will talk about the 'dimension' of an array instead of its capacity.

To summarize, we have chosen to formalize dynamically-allocated arrays while abstracting away any considerations about memory handling. The reader may wonder why we have taken such a decision. The answer is twofold. First, dynamically-allocated arrays are at the core of many algorithms and abstract datatypes (such as heaps, queues, and hash tables). So, the availability of a procedure (cf. Section 4.2) to reason about such a type of arrays would greatly help the task of verifying many programs. The second reason is that dynamically-allocated arrays more accurately model heaps, i.e. the areas of memory where pointer-based data structures are dynamically allocated. For example, as observed in McPeak and Necula (2005), the absence of aliasing in linked lists can be specified by using an axiom for injectivity of the function modelling the heap. It is possible to extend dynamic arrays with a recognizer for "injective arrays", where $\perp$ models the null-pointer, and obtain a decision procedure also for this theory (cf. Subsection 4.3.1). As another example, consider Separation Logic as introduced in Reynolds (1979). The key feature of this logic is its capability to support "local reasoning" by formalizing heaps as partial function from addresses to values and introducing new logical connectives, such as the separating conjunction $P \star Q$ that asserts that P and Q hold for *disjoint* portions of a certain heap. Indeed, the partial function modelling heaps can be turned into total functions by using the standard trick of returning an undefined value whenever they are undefined. In this sense, heaps can naturally be seen as dynamic arrays, which can be extended with a "domain" function, returning the set of non-$\perp$ elements. We will see that also this extension of the theory of arrays with dimension is decidable (cf. Subsection 4.3.2); this can also be seen as a first step in the direction of providing automatic support for Separation Logic by decision procedures developed in first-order logic.

We are now in the position to discuss the simple mathematical model underlying dynamic arrays. Given a set A, by $Arr(A)$ we denote the set of finite arrays with natural numbers as indexes and whose elements are from A. An element of $Arr(A)$ is a sequence $a : \mathbb{N} \longrightarrow A \cup \{\perp\}$ eventually equal to $\perp$ (here $\perp$ is an element not in A denoting an "undefined" value). In this way, for every array $a \in Arr(A)$ there is a smallest index $n \geq 0$, called the *dimension* of a, such that the value of a at index j is equal to $\perp$ for $j \geq n$. We do not require any value of a at $k < n$ to be distinct from $\perp$: this is also the reason to use the word 'dimension' rather than 'length'. There is just one array whose dimension is zero which we indicate by ε and call it the *empty array*. Since many applications of verification require arithmetic expressions

on indexes of arrays, we introduce Presburger Arithmetic $\mathcal{P}$ over indexes: any other decidable fragment of Arithmetic would be a good alternative. Thus the relevant operations on our arrays include *addition over indexes, read, write, and dimension*. Below, we will consider a theory, denoted by $\mathcal{ADP}$, capable of formally expressing the properties described above.

4.1.1 Arrays with Dimension as a Combined Theory

We work in *many-sorted first-order logic with equality* and we assume the basic syntactic and semantic concepts as in, e.g., Gallier (1986).

A *signature* Σ is a non-empty set of sort symbols together with a set of function symbols and a set of predicate symbols (both equipped with suitable lists of sort symbols as arity). The set of predicate symbols always contains a symbol $=_S$ for equality for every sort S (we usually omit its subscript). To avoid confusion, we use the symbol $\equiv$ (instead of $=$) in the metalanguage to mean identity of syntactic expressions.

From the semantic side, a Σ-*structure* $\mathcal{M}$ consists of non-empty and pairwise disjoint domains $S^{\mathcal{M}}$ for every sort S, and interprets each function symbol f and predicate symbol P as functions $f^{\mathcal{M}}$ and relations $P^{\mathcal{M}}$, respectively, according to their arities. We use $f^{\mathcal{M}}$ (resp. $P^{\mathcal{M}}$) to denote the interpretation of the function symbol f (resp. predicate symbol P) in the structure $\mathcal{M}$ (the equality predicate $=_S$ is always interpreted as the identity relation over the sort S). All the remaining notions from Subsection 1.1.1 in Chapter 1 can be easily adapted.

Formally, the theory $\mathcal{ADP}$ can be seen as a combination of two well-known theories: $\mathcal{P}$ and the theory $\mathcal{A}_e$ of arrays with extensionality (see, e.g., Armando et al., 2003), extended with a function for the dimension which takes an array and returns a natural number. Because of the function for dimension, the combination is *non-disjoint* and cannot be handled by classical combination schemas such as Nelson and Oppen (1979). Nevertheless, following Ghilardi (2004), it is convenient to see $\mathcal{ADP}$ as a combination of $\mathcal{P}$ with a theory of array with dimension $\mathcal{A}_{dim}$: $\mathcal{A}_{dim}$ extends $\mathcal{A}_e$ (both in the signature and in the axioms), but is contained in $\mathcal{ADP}$, because in $\mathcal{A}_{dim}$ indexes are only endowed with a discrete linear poset structure. In this way, we have that $\mathcal{ADP} = \mathcal{A}_{dim} \cup \mathcal{P}$ and the theories $\mathcal{A}_{dim}$ and $\mathcal{P}$ share the well-known complete theory $\mathcal{T}_0$ of natural numbers endowed with zero and successor (see, e.g., Enderton, 1972): this theory admits quantifier elimination, so that the $\mathcal{T}_0$-compatibility hypothesis of Ghilardi (2004) needed for the non-disjoint Nelson-

Oppen combination is satisfied. Unfortunately, the combination result in Ghilardi (2004) cannot be applied to $\mathcal{ADP}$ for mainly two reasons. First, $\mathcal{T}_0$ is not locally finite (see, e.g., Ghilardi, 2004 for details). Secondly, $\mathcal{A}_{dim}$ is a proper extension of the theory $\mathcal{A}_e$, hence the decision procedures for the $\mathcal{A}_e$-satisfiability problem (such as, e.g., the one in Armando et al., 2003) must be extended. In the rest of the chapter, we will show that it is sufficient to use decision procedures for the $\mathcal{P}$- and $\mathcal{A}_e$-satisfiability problem to solve the $\mathcal{ADP}$-satisfiability problem, provided that a suitable pre-processing of the input set of literals is performed.

We now introduce the basic theories of interests for this chapter.

$\boxed{\mathcal{T}_0}$ has just one sort symbol INDEX, the following function and predicate symbols: 0 : INDEX, s : INDEX $\rightarrow$ INDEX, and $<$: INDEX $\times$ INDEX. It is axiomatized by the following formulae:[1]

$$y \neq 0 \rightarrow \exists z(y = \mathsf{s}(z)) \tag{4.1}$$

$$x < \mathsf{s}(y) \leftrightarrow (x < y \vee x = y) \tag{4.2}$$

$$\neg(x < 0) \tag{4.3}$$

$$x < y \vee x = y \vee y < x \tag{4.4}$$

$$x < y \rightarrow \neg(y < x) \tag{4.5}$$

$$x < y \rightarrow (y < z \rightarrow x < z) \tag{4.6}$$

where x, y and z are variables of sort INDEX. This theory admits elimination of quantifiers and it is complete (see Enderton, 1972 for details).

$\boxed{\mathcal{P}}$ is the well-known Presburger Arithmetic (see, e.g., Enderton, 1972) over indexes. The signature is that of $\mathcal{T}_0$ extended with the function symbol for addition $+$: INDEX $\times$ INDEX $\rightarrow$ INDEX, written infix. Since $\mathcal{P}$ is not finitely axiomatizable (see again Enderton, 1972), we assume as axioms all the sentences valid in the standard model of natural numbers. Notice that $\mathcal{T}_0 \subset \mathcal{P}$.

$\boxed{\mathcal{A}}$ is the theory of arrays (see, e.g., Armando et al., 2003) which has the following signature:

- sort symbols: INDEX, ELEM, ARRAY and

[1] Here and in the following, we omit the outermost universal quantification for the sake of readability.

– function symbols: select : ARRAY × INDEX → ELEM and store : ARRAY × INDEX × ELEM → ARRAY

and it is axiomatized by the following formulae:

$$\mathsf{select}(\mathsf{store}(a, i, e), i) = e \tag{4.7}$$

$$i \neq j \rightarrow \mathsf{select}(\mathsf{store}(a, i, e), j) = \mathsf{select}(a, j) \tag{4.8}$$

$\boxed{\mathcal{A}_e}$ is the theory of arrays with extensionality (see, e.g., Armando et al., 2003) which has the same signature of $\mathcal{A}$ and it is axiomatized by (4.7), (4.8), and the axiom of extensionality:

$$\forall i(\mathsf{select}(a, i) = \mathsf{select}(b, i)) \rightarrow a = b \tag{4.9}$$

The converse implication is an obvious consequence of the congruence of equality; hence, there is no need to explicitly take it into account since we work in (many-sorted) first-order logic with equality. Notice also that $\mathcal{A} \subset \mathcal{A}_e$.

$\boxed{\mathcal{A}_{dim}}$ is the simple theory of arrays with dimension whose signature is the union of the signatures of $\mathcal{T}_0$ and $\mathcal{A}_e$ extended with the following three symbols: $\perp$: ELEM, ε : ARRAY, and dim : ARRAY → INDEX. It is axiomatized by the axioms in $\mathcal{T}_0$, those in $\mathcal{A}_e$, and the following formulae:

$$\mathsf{dim}(a) \leq i \rightarrow \mathsf{select}(a, i) = \perp \tag{4.10}$$

$$\mathsf{dim}(a) = \mathsf{s}(i) \rightarrow \mathsf{select}(a, i) \neq \perp \tag{4.11}$$

$$\mathsf{dim}(\varepsilon) = 0 \tag{4.12}$$

Notice that $\mathcal{T}_0 \subset \mathcal{A}_{dim}$ and $\mathcal{A}_e \subset \mathcal{A}_{dim}$.

$\boxed{\mathcal{ADP}}$ is the theory of arrays with dimension whose signature is the union of the signatures of $\mathcal{A}_{dim}$ and $\mathcal{P}$ and is axiomatized by the axioms in $\mathcal{A}_{dim}$ and all valid sentences in $\mathcal{P}$.

The theories $\mathcal{T}_0$ and $\mathcal{P}$ are decidable (see Enderton, 1972); moreover, the constraint satisfiability problem for the theories $\mathcal{A}$ and $\mathcal{A}_e$ is decidable (see Armando et al., 2003). These are important observations for the results presented in this chapter, since the decision procedure for $\mathcal{ADP}$-satisfiability will assume the availability of two decision procedures for the constraint satisfiability problems of $\mathcal{P}$ and

$\mathcal{A}$. The theories $\mathcal{A}_e$, $\mathcal{A}_{dim}$, and $\mathcal{ADP}$ admit a particular subclass of models, which we call the *standard* ones and are exactly those introduced above in order to motivate the definition of $\mathcal{ADP}$. Formally, a standard model is the model induced by a pair (A, κ), where A is a set of elements and κ is a distinguished element of A as explained in the following definition.

Definition 4.1.1. Let A be a set and κ be an element of A. The *standard model of $\mathcal{ADP}$ induced by the pair* (A, κ) is the $\Sigma_{\mathcal{ADP}}$-structure $\mathcal{M}$ such that

(i) the sort INDEX is interpreted in $\mathcal{M}$ as $\mathbb{N}$ and the symbols $0, <, s, +$ have their natural meaning;

(ii) the sort ELEM is interpreted in $\mathcal{M}$ as A and the constant $\bot$ is interpreted as κ;

(iii) the sort ARRAY is interpreted in $\mathcal{M}$ as the set of functions $a : \mathbb{N} \longrightarrow A$ such that there is some $n_a \in \mathbb{N}$ for which we have $a(m) = \kappa$ whenever $m \geq n_a$; moreover, the constant ε is interpreted as the constant function with value κ;

(iv) $dim^{\mathcal{M}}(a)$ is the smallest $n \in \mathbb{N}$ such that $a(m) = \kappa$ holds for all $m \geq n$;

(v) we have $\mathsf{select}^{\mathcal{M}}(a, i) := a(i)$ and

$$\mathsf{store}^{\mathcal{M}}(a, i, e)(n) := \begin{cases} a(n) & \text{if } n \neq i, \\ e & \text{otherwise.} \end{cases}$$

The standard models of $\mathcal{A}_e$ and $\mathcal{A}_{dim}$ can be defined in a similar way by taking the $\Sigma_{\mathcal{A}_e}$- and $\Sigma_{\mathcal{A}_{dim}}$-reduct (respectively) of $\mathcal{ADP}$-standard models; notice that the dimension of the empty array is 0 and the dimension of a non-empty array is the successor of the index of the last element different from $\bot$. Of course, when investigating constraint satisfiability we are mainly interested in satisfiability of constraints in standard models and we shall in fact prove that a constraint is satisfiable in a model of $\mathcal{ADP}$ if and only if it is satisfiable in a standard model (cf. Lemma 4.2.7 below).

4.2 A Decision Procedure for Arrays with Dimension

In the rest of the chapter, we assume the availability of two decision procedures solving the $\mathcal{A}$- and $\mathcal{P}$-satisfiability problems; we will see how to reduce to these

latter the $\mathcal{ADP}$-satisfiability problem. In order to introduce the reader into the details of the procedure, we consider an example which illustrates some key ideas.

Example 4.2.1. Consider the problem of checking the $\mathcal{ADP}$-satisfiability of

$$\begin{gathered}\mathsf{dim}(a) = n \ \wedge \ \mathsf{dim}(b) = m \ \wedge \ b = \mathsf{store}(a, n, e) \ \wedge \\ \wedge \ e \neq \bot \ \wedge \ m > 0 \ \wedge \ n = m + 1\end{gathered} \tag{4.13}$$

where a, b, m, n, e are free constants of appropriate sorts. To detect the unsatisfiability of (4.13), it is crucial to derive that $m < n$ in Presburger Arithmetic. In fact, we can detect the $\mathcal{A}_e$-unsatisfiability of $b = \mathsf{store}(a, n, e) \wedge e \neq \bot$ in (4.13) and $\mathsf{select}(b, n) = \bot$, which is a logical consequence of (4.10) and $\mathsf{dim}(b) = m < n = \mathsf{dim}(a)$. If we were not able to derive facts in Presburger Arithmetic, we would have failed to show the $\mathcal{ADP}$-unsatisfiability of (4.13).

The capability of deriving all facts entailed by a constraint can be problematic, since we only assume the availability of a decision procedure to solve the $\mathcal{P}$-satisfiability problem without further capabilities. To overcome this difficulty, we will transform the problem of checking a logical consequence into a satisfiability problem, i.e. if φ and ψ are two constraints in $\mathcal{P}$, then in order to check $\mathcal{P} \cup \{\varphi\} \models \psi$, we will check the $\mathcal{P}$-unsatisfiability of $\varphi \wedge \neg\psi$. Indeed, it will be necessary to *guess* the entailed constraint ψ. This is a standard technique in the field of combining decision procedures (see, e.g., Ghilardi, 2004), which allows us to abstractly describe our decision procedure and more easily prove its correctness.

4.2.1 The Architecture

The overall schema of the decision procedure for the $\mathcal{ADP}$-satisfiability problem is depicted in Figure 4.1.

The module Flatten pre-processes the literals in the input constraint so as to make them flat and easily recognizable as belonging to one theory among those used to define $\mathcal{ADP}$ (cf. Section 4.1.1), i.e. $\mathcal{T}_0$, $\mathcal{P}$, $\mathcal{A}_e$, or $\mathcal{A}_{dim}$. The module $\mathcal{E}$-instantiation produces suitable instances of the extensionality axiom, i.e. (4.9), so that a simpler decision procedure for the $\mathcal{A}$-satisfiability problem (with respect to one for $\mathcal{A}_e$) is assumed available. The module $\mathcal{G}$-instantiation is non-deterministic and guesses sufficiently many facts which are potentially entailed by the constraints in $\mathcal{P}$. The modules $\mathsf{DP}_{\mathcal{P}}$ and $\mathsf{DP}_{\mathcal{A}}$ implement the decision procedures for Presburger Arithmetic and for the constraint satisfiability problem for the theory of arrays

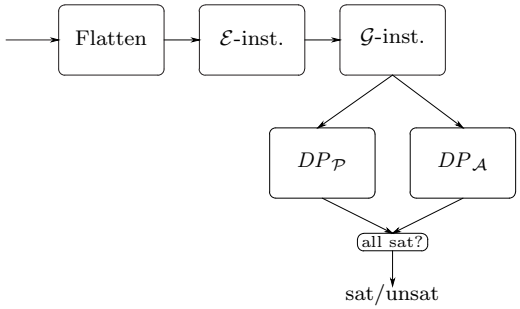

Figure 4.1: The architecture of the decision procedure for $\mathcal{ADP}$

(without extensionality). The module 'all sat?' returns *"satisfiable"* if both decision procedures for $\mathcal{P}$ and $\mathcal{A}$ returned *"satisfiable"*; otherwise returns *"unsatisfiable"*. Now, we are ready to describe the internal workings of each module in detail.

Flattening

It is well-known that it is possible to transform a constraint φ into an equisatisfiable constraint φ' containing only flat literals in linear time by introducing sufficiently many fresh constant symbols to name subterms (see, e.g., Armando et al., 2003). In our case, we assume that the module Flatten in Figure 4.1 transforms (in linear time) a set of arbitrary literals over the signature $\Sigma^{\underline{a}}_{\mathcal{ADP}}$ into an equisatisfiable set of flat literals on the signature $\Sigma^{\underline{c}}_{\mathcal{ADP}}$, for some set $\underline{c} \supseteq \underline{a}$ of constants (the constants in $\underline{c} \setminus \underline{a}$ are said to be fresh). Notice that a flattened set of literals L over a simple expansion of $\Sigma_{\mathcal{ADP}}$ can be represented as a set-theoretic union $L_{\mathcal{A}_{dim}} \cup L_{\mathcal{P}}$, where $L_{\mathcal{A}_{dim}}$ collects all the literals from L over a simple expansion of $\Sigma_{\mathcal{A}_{dim}}$ and $L_{\mathcal{P}}$ collects all the literals from L over a simple expansion of $\Sigma_{\mathcal{P}}$ (thus $L_{\mathcal{A}_{dim}} \cap L_{\mathcal{P}}$ contains precisely the literals from L over a simple expansion of $\Sigma_{\mathcal{T}_0}$).

$\mathcal{E}$-instantiation closure

The module $\mathcal{E}$-instantiation finds enough instances of the axiom (4.9) for extensionality of arrays so that it can be eliminated without compromising the correctness of the decision procedure for $\mathcal{ADP}$.

Definition 4.2.2 ($\mathcal{E}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{E}$-instantiation closed if and only if the following condition is satisfied:

1. if $a \neq b \in L$, with $a, b :$ ARRAY, then $\{\mathsf{select}(a, i) = e_1, \mathsf{select}(b, i) = e_2, e_1 \neq e_2\} \subseteq L$ for some constants $i :$ INDEX, $e_1, e_2 :$ ELEM;

It is not difficult to see that, given a set of ground flat literals L, there exists an $\mathcal{ADP}$-equisatisfiable set $L^{\mathcal{E}} \supseteq L$ that contains the Skolemization of some logical consequences of $\mathcal{A}_e \cup L$ and is $\mathcal{E}$-instantiation closed.

Lemma 4.2.3. *There exists a linear time algorithm which takes a set L of flat literals over the signature $\Sigma^{\underline{a}}_{\mathcal{ADP}}$ and returns an $\mathcal{E}$-instantiation closed set $L^{\mathcal{E}}$ of flat literals over the signature $\Sigma^{\underline{c}}_{\mathcal{ADP}}$ such that (i) $L \subseteq L^{\mathcal{E}}$, (ii) L and $L^{\mathcal{E}}$ are $\mathcal{ADP}$-equisatisfiable, and (iii) $\underline{a} \subseteq \underline{c}$.*

The signature $\Sigma^{\underline{c}}_{\mathcal{ADP}}$ of $L^{\mathcal{E}}$ is a proper simple expansion of the signature $\Sigma^{\underline{a}}_{\mathcal{ADP}}$ of L, because Skolem constants are fresh. It is straightforward to see that, if L contains n literals, at most $3n$ new literals are sufficient to obtain an $\mathcal{E}$-instantiation closed set of literals containing L. Under the assumption that producing a new literal takes constant time, there exists a linear time algorithm to compute $\mathcal{E}$-instantiation closed sets.

$\mathcal{G}$-instantiation closure

The module $\mathcal{G}$-instantiation is non-deterministic and it is responsible to produce suitable instances of the axioms about the dimension of arrays, i.e. (4.10) and (4.11), and to guess enough facts of $\mathcal{P}$ entailed by the input constraint so as to guarantee the correctness of the overall decision procedure for $\mathcal{ADP}$-satisfiability.

Definition 4.2.4 ($\mathcal{G}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{G}$-instantiation closed if and only if the following conditions are satisfied:

1. if ε occurs in L, then $\mathsf{dim}(\varepsilon) = 0 \in L$;
2. if $\mathsf{dim}(a) = i \in L$, with a : ARRAY and i : INDEX, then $\{i = 0\} \subseteq L$ or $\{e \neq \bot, \mathsf{select}(a, j) = e, s(j) = i\} \subseteq L$ for some constants j : INDEX and e : ELEM;
3. if i, j occur in L, with i, j : INDEX, then $i = j \in L$ or $i \neq j \in L$;
4. if i, j occur in L, with i, j : INDEX and $i \neq j \in L$, then $i < j \in L$ or $j < i \in L$;
5. if $\{\mathsf{dim}(a) = i, i \leq j\} \subseteq L$, with a : ARRAY and i, j : INDEX, then $\{\mathsf{select}(a, j) = \bot\} \subseteq L$ (here $i \leq j$ stands for $i < j$ or $i = j$).

Given a set of literals, it is always possible to compute an equisatisfiable $\mathcal{G}$-instantiation closed set in (non-deterministic) polynomial time.

Lemma 4.2.5. *There exists a non-deterministic polynomial time algorithm which takes as input a set L of ground flat literals over a signature $\Sigma^a_{\mathcal{ADP}}$ and returns a $\mathcal{G}$-instantiation closed set $L^{\mathcal{G}}$ of flat literals over the signature $\Sigma^c_{\mathcal{ADP}}$ such that (i) $L \subseteq L^{\mathcal{G}}$, (ii) L and $L^{\mathcal{G}}$ are $\mathcal{ADP}$-equisatisfiable, and (iii) $\underline{a} \subseteq \underline{c}$.*

Proof. Let m be the number of literals in L of the form $\mathsf{dim}(a_k) = d_{a_k}$ where d_{a_k} is a constant of sort INDEX. Let us consider a set $\underline{b} = \{j_1, \ldots, j_m, e_1, \ldots, e_m\}$ of fresh constants, where j_k : INDEX, e_k : ELEM, and $k \in \{1, \ldots, m\}$. A $\mathcal{G}$-instantiation $L^{\mathcal{G}}$ of L can be computed by sequentially executing the following three steps:

1. for each pair i, j of constants of sort INDEX in $\underline{a} \cup \underline{b} \cup \{0\}$, exactly one of the atoms $i = j$ and $i \neq j$ is added to $L^{\mathcal{G}}$, and in the latter case either $i < j$ or $j < i$ is also added;

2. for each literal $\mathsf{dim}(a_k) = d_{a_k} \in L^{\mathcal{G}}$, then:

   (a) if $0 = d_{a_k} \in L^{\mathcal{G}}$ or $0 \equiv d_{a_k}$, then add $\{j_k = 0, e_k = \bot\}$ to $L^{\mathcal{G}}$;

   (b) if $0 < d_{a_k} \in L^{\mathcal{G}}$, then add $\{\mathsf{s}(j_k) = d_{a_k}, \mathsf{select}(a_k, j_k) = e_k, e_k \neq \bot\}$ to $L^{\mathcal{G}}$.

3. if $\{\mathsf{dim}(a) = i, i \leq j\} \subseteq L^{\mathcal{G}}$, then add $\{\mathsf{select}(a, j) = \bot\}$ to $L^{\mathcal{G}}$.

There are two important observations. First, each new constant j_k : INDEX ($k \in \{1, \ldots, m\}$) denotes the predecessor of the dimension of a_k, when the latter is guessed to be different from 0 (if the dimension of a_k is guessed to be 0, then j_k is set to zero). Second, each new constant e_k : ELEM ($k \in \{1, \ldots, m\}$) denotes the result of reading the content of array a_k at position j_k.

These two observations together with the fact that the process described above to build $L^{\mathcal{G}}$ closely follows Definition 4.2.4 should make it clear that L is $\mathcal{ADP}$-satisfiable if and only if there exist a set $L^{\mathcal{G}}$ which is $\mathcal{G}$-instantiation closed and $\mathcal{ADP}$-satisfiable. The non-deterministic polynomial time result is obtained by a straightforward inspection of the process described above. □

It is easy to check that one obtains a set of both $\mathcal{E}$- and $\mathcal{G}$-instantiation closed set of literals by invoking first the $\mathcal{E}$- and then the $\mathcal{G}$-instantiation module.

4.2.2 The Algorithm

Algorithm 3 gives a (non-deterministic) decision procedure to solve the $\mathcal{ADP}$-satisfiability problem. Without loss of generality (cf. Subsection 4.2.1), we assume that L contains only flat literals.

Algorithm 3 The (extensible) decision procedure for $\mathcal{ADP}$

Require: L set of flat literals
$\mathsf{T} \longleftarrow \{\mathcal{A}, \mathcal{P}\}$
1: **procedure** $\mathsf{DP}_{\mathcal{ADP}}(L)$
2: $L^{\mathcal{E}} \leftarrow \mathcal{E}$-*instantiation*$(L)$
3: **for all** $L^{\mathcal{G}} \in \mathcal{G}$-*instantiation*$(L^{\mathcal{E}})$ **do**
4: **if** $\bigwedge_{T \in \mathsf{T}} (\mathsf{DP}_T(L^{\mathcal{G}}_T) =$ *"satisfiable"*$)$ **then**
5: **return** *"satisfiable"*
6: **end if**
7: **end for**
8: **return** *"unsatisfiable"*
9: **end procedure**

The function DP_T, for $T \in \{\mathcal{ADP}, \mathcal{A}, \mathcal{P}\}$, denotes a decision procedure to solve the T-satisfiability problem, i.e. DP_T takes a set L of literals over (a simple expansion of) the signature Σ_T and returns *"satisfiable"* when L is T-satisfiable; *"unsatisfiable"*, otherwise. If L is a set of flat literals, then

$$L_T := \{\ell \mid \ell \in L \text{ is a } \Sigma^a_T\text{-literal}\},$$

where $T \in \{\mathcal{A}, \mathcal{P}\}$. So, for example, $L^{\mathcal{G}}_{\mathcal{P}}$ is the subset of the literals in $L^{\mathcal{G}}$ over a simple expansion of the signature $\Sigma_{\mathcal{P}}$ (for the sake of readability, when it is clear from the context, the term "simple expansion" will be omitted). The set T in Algorithm 3 contains the names of the theories for which a decision procedure for the T-satisfiability problem is assumed available.

Let L be a set of flat literals over the signature $\Sigma_{\mathcal{ADP}}$ to be checked for $\mathcal{ADP}$-satisfiability. The decision procedure $\mathsf{DP}_{\mathcal{ADP}}$ first computes the $\mathcal{E}$-instantiation $L^{\mathcal{E}}$ of L (recall from Lemma 4.2.3 that this can be done in linear time). Then, it enumerates all possible $\mathcal{G}$-instantiations (cf. the **for each** loop in Algorithm 3). If it is capable of finding a $\mathcal{G}$-instantiation $L^{\mathcal{G}}$ such that its literals in $L^{\mathcal{G}}_{\mathcal{P}}$ over the signature $\Sigma_{\mathcal{P}}$ are $\mathcal{P}$-satisfiable and its literals in $L^{\mathcal{G}}_{\mathcal{A}}$ over the signature $\Sigma_{\mathcal{A}}$ are $\mathcal{A}$-satisfiable, then $\mathsf{DP}_{\mathcal{ADP}}$ returns the $\mathcal{ADP}$-satisfiability of the input set L of literals. Otherwise, if all possible $\mathcal{G}$-instantiations are enumerated and the test of the conditional in the body of the loop always fails, $\mathsf{DP}_{\mathcal{ADP}}$ returns the $\mathcal{ADP}$-unsatisfiability of the input set L of literals.

Theorem 4.2.6. *The constraint satisfiability problem for $\mathcal{ADP}$ is* NP*-complete.*

The proof is based on the following considerations: (i) the constraint satisfia-

bility problem for the theory of Presburger Arithmetic reduces to the Integer Linear Programming problem; (ii) both Integer Linear Programming and constraint satisfiability for the theory of array are known to be NP-complete problems (see, e.g., Schrijver, 1986; Stump et al., 2001 respectively); (iii) the size of the $\mathcal{E}$- and $\mathcal{G}$-instantiation closed set is polynomially bounded with respect to the size of the original constraint. From (i) and (ii) it follows the NP-hardness of the problem, whereas from (iii) and the correctness of $\mathsf{DP}_{\mathcal{ADP}}$ (Theorem 4.2.8) it follows that the problem is in NP, hence the thesis.

4.2.3 Correctness of the Procedure

The termination of $\mathsf{DP}_{\mathcal{ADP}}$ is obvious, since the computation of $L^{\mathcal{E}}$ terminates (cf. Lemma 4.2.3) and there are only finitely many possible sets $L^{\mathcal{G}}$ to be considered in the **for each** loop of Algorithm 3 (cf. Lemma 4.2.5).

The soundness and completeness of $\mathsf{DP}_{\mathcal{ADP}}$ are consequences of the following combination lemma:

Lemma 4.2.7 (Combination)**.** *Let L be an $\mathcal{E}$- and $\mathcal{G}$-instantiation closed set. Then the following conditions are equivalent:*

(i) L is satisfiable in a standard model of $\mathcal{ADP}$;

(ii) L is $\mathcal{ADP}$-satisfiable;

(iii) $L_{\mathcal{A}}$ is $\mathcal{A}$-satisfiable and $L_{\mathcal{P}}$ is $\mathcal{P}$-satisfiable.

Proof. Since the implications (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) are trivial, it is sufficient to show that (iii) $\Rightarrow$ (i) to conclude the proof.

Let $\mathcal{M}'$ be a structure such that $\mathcal{M}' \models \mathcal{A} \cup L_{\mathcal{A}}$ and $\mathcal{N}$ be a structure such that $\mathcal{N} \models \mathcal{P} \cup L_{\mathcal{P}}$. Since $\mathcal{P}$ is complete, we are entitled to assume that $\mathcal{N}$ is the standard structure of natural numbers $\mathbb{N}$. We are now ready to build a standard model $\mathcal{M}$ for $\mathcal{ADP} \cup L$ out of $\mathcal{M}'$ as follows. We take $\textsc{elem}^{\mathcal{M}}$ to be $\textsc{elem}^{\mathcal{M}'}$ and $\bot^{\mathcal{M}}$ to be $\bot^{\mathcal{M}'}$; the free constants occurring in L are interpreted as follows:

(A) for each constant $i : \textsc{index}$ occurring in $L_{\mathcal{P}}$, let $i^{\mathcal{M}} := i^{\mathcal{N}}$;

(B) for each constant $e : \textsc{elem}$ occurring in $L_{\mathcal{A}}$, let $e^{\mathcal{M}} := e^{\mathcal{M}'}$;

(C) for each constant $a : \text{ARRAY}$ occurring in $L_{\mathcal{A}}$, we define $a^{\mathcal{M}}$ to be the sequence $\{e_n\}$ such that

$$e_n := \begin{cases} \mathsf{select}(a, i)^{\mathcal{M}'} & \text{if } n = i^{\mathcal{M}} \text{ for some } i \text{ occurring in } L_{\mathcal{P}}, \\ \bot^{\mathcal{M}} & \text{otherwise.} \end{cases}$$

The construction in (C) is well-defined; indeed, if two constants i_1 and i_2 of sort INDEX occurring in $L_{\mathcal{P}}$ are interpreted into the same element in $\mathcal{M}$, then $i_1^{\mathcal{N}} = i_2^{\mathcal{N}}$; since L is $\mathcal{G}$-instantiation closed, the atom $i_1 = i_2$ is in $L_{\mathcal{P}}$ (and hence in $L_{\mathcal{A}}$) and so $\mathcal{M}' \models \mathsf{select}(a, i_1) = \mathsf{select}(a, i_2)$.

Now, we show that for each $\ell \in L$, we have $\mathcal{M} \models \ell$. This is obvious for $\ell \in L_{\mathcal{P}}$ and for ℓ of the form $e_1 = e_2$ or $e_1 \neq e_2$, with $e_1, e_2 : \text{ELEM}$. We are left to consider the following cases depending on the form of ℓ:

(i) $\mathsf{select}(a, i) = e$. $\mathcal{M} \models \ell$ because of (A), (B) and (C);

(ii) $a_1 = a_2$, with $a_1, a_2 : \text{ARRAY}$. $\mathcal{M} \models \ell$ because $a_1^{\mathcal{M}'} = a_2^{\mathcal{M}'}$, so $\mathsf{select}(a_1, i)^{\mathcal{M}'} = \mathsf{select}(a_2, i)^{\mathcal{M}'}$ for each constant $i : \text{INDEX}$ occurring in $L_{\mathcal{P}}$. Hence, $a_1^{\mathcal{M}} = a_2^{\mathcal{M}}$ by (C);

(iii) $\mathsf{store}(a_1, i, e) = a_2$. $\mathcal{M} \models \ell$ by considering an argument similar to that used for case (ii);

(iv) $a_1 \neq a_2$, with $a_1, a_2 : \text{ARRAY}$. $\mathcal{M} \models \ell$ since

$$\{\mathsf{select}(a_1, i) = e_1, \mathsf{select}(a_2, i) = e_2, e_1 \neq e_2\} \subseteq L_{\mathcal{A}}$$

by Definition 4.2.2 of $\mathcal{E}$-instantiation closed set of literals and $\mathcal{M}' \models L_{\mathcal{A}}$ and hence $\mathsf{select}(a_1, i)^{\mathcal{M}} \neq \mathsf{select}(a_2, i)^{\mathcal{M}}$ because of (i). As a consequence, we have $a_1^{\mathcal{M}} \neq a_2^{\mathcal{M}}$.

(v) $\mathsf{dim}(a) = i$. We consider two sub-cases, according to Definition 4.2.4(2):

- if $i = 0 \in L_{\mathcal{P}}$ or $i \equiv 0$, then it is sufficient to prove that for each integer n, e_n is equal to $\bot^{\mathcal{M}}$ where $\{e_n\} = a^{\mathcal{M}}$. If $n = j^{\mathcal{M}}$ for some constant $j : \text{INDEX}$ such that

$$\{i < j\} \subseteq L_{\mathcal{P}} \quad \text{or} \quad \{i = j\} \subseteq L_{\mathcal{P}},$$

then, since L is $\mathcal{G}$-instantiation closed, $\mathsf{select}(a, j) = \bot \in L_{\mathcal{A}}$ hence $e_n = \bot^{\mathcal{M}}$ by (C); otherwise, $e_n = \bot^{\mathcal{M}}$ by (C).

– if $i \neq 0 \in L_{\mathcal{P}}$, then for each integer $n \geq i^{\mathcal{M}}$, $e_n = \bot^{\mathcal{M}}$ by a similar argument to the one used for the previous sub-case. In fact, we observe that since L is $\mathcal{G}$-instantiation closed, $\mathsf{s}(j) = i$ is in $L_{\mathcal{P}}$ for some constant j : INDEX, and both $\mathsf{select}(a, j) = e$ and $e \neq \bot$ must also be in $L_{\mathcal{A}}$, therefore the thesis follows from (B), (C) and (i).

□

Now, we are able to state and prove the correctness of $\mathsf{DP}_{\mathcal{ADP}}$.

Theorem 4.2.8. *$\mathsf{DP}_{\mathcal{ADP}}$ is a decision procedure for the $\mathcal{ADP}$-satisfiability problem, i.e. for any set L of flat literals, L is $\mathcal{ADP}$-satisfiable if and only if $\mathsf{DP}_{\mathcal{ADP}}(L)$ returns "satisfiable". Furthermore, $\mathsf{DP}_{\mathcal{ADP}}$ decides the satisfiability problem in the standard models of $\mathcal{ADP}$.*

Proof. If L is $\mathcal{ADP}$-satisfiable, then it is obvious that $\mathsf{DP}_{\mathcal{ADP}}(L)$ returns *"satisfiable"*. We are left with the task of proving that the converse holds. We will prove that when $\mathsf{DP}_{\mathcal{ADP}}(L)$ returns *"satisfiable"*, then L is satisfiable in a standard model of $\mathcal{ADP}$. If $\mathsf{DP}_{\mathcal{ADP}}(L)$ returns *"satisfiable"*, then $\mathsf{DP}_{\mathcal{ADP}}$ has found a $\mathcal{G}$-instantiation $L^{\mathcal{G}}$ of $L^{\mathcal{E}}$ at some iteration of the **for each** loop in Algorithm 3. The set $L^{\mathcal{G}}$ is such that

$$L^{\mathcal{G}}_{\mathcal{A}} \text{ is } \mathcal{A}\text{-satisfiable and } L^{\mathcal{G}}_{\mathcal{P}} \text{ is } \mathcal{P}\text{-satisfiable.}$$

From these two facts, the existence of a standard $\mathcal{ADP}$-model of $L^{\mathcal{G}}$ immediately follows by using Lemma 4.2.7 above. □

4.3 Extensions of the Theory of Arrays with Dimension

We now show the decidability of the constraint satisfiability problem for some interesting extensions of $\mathcal{ADP}$.

As observed in McPeak and Necula (2005), certain properties of pointer-based data structures, such as *no-aliasing*, can be specified by using first-order axioms. The first extension of $\mathcal{ADP}$ is obtained by adding an axiom recognizing injective arrays (which, according to McPeak and Necula, 2005, may characterize memory configurations where pointers satisfy the no-aliasing property) and then showing how to extend the decision procedure for $\mathcal{ADP}$ by an instantiation strategy so as

to consider enough (ground) instances of the injectivity axiom. We notice that the decidability of a similar problem in Bradley et al. (2006) was left open and finally stated as undecidable in Bradley (2007): we are capable of deriving a decidability result since we use a richer theory that identifies a more restricted class of models.

The second extension of $\mathcal{ADP}$ we consider is again motivated by applications in program verification. As already observed in Reynolds (1979), it is quite helpful to regard arrays as functions equipped with an operator to compute their domains. This is used, for example, to define the semantics of separating connectives (supporting local reasoning) in Separation Logic (see Reynolds, 2002). So, we extend $\mathcal{ADP}$ with a set of axioms characterizing a function which, given an array a, returns the domain D of a, i.e. D is a set of indexes such that $\mathsf{select}(a, i) \neq \bot$ for i in D. We regard this as a first step in the direction of providing automatic support for Separation Logic by decision procedures developed in first-order logic.

The section concludes taking into account some other interesting extensions, which exemplify the flexibility of our approach and are all relevant for applications as discussed in, e.g., Bradley et al. (2006).

4.3.1 Injective Arrays

We extend the (empty) set of predicate symbols in $\mathcal{ADP}$ by the unary predicate symbol Inj : ARRAY which, intuitively, recognizes injective arrays, i.e. arrays containing unique elements, with the exception of the undefined element $\bot$. To formalize the intuitive meaning of Inj, we extend the set of axioms of $\mathcal{ADP}$ by the following definition:

$$\mathsf{Inj}(a) \;\leftrightarrow\; \forall i, j(\mathsf{select}(a, i) = \mathsf{select}(a, j) \rightarrow i = j \vee \mathsf{select}(a, i) = \bot) \tag{4.14}$$

where a is an implicitly universally quantified variable of sort ARRAY. Let $\mathcal{ADP}_{\mathsf{inj}}$ be the theory obtained by extending $\mathcal{ADP}$ with axiom (4.14). Notice that, since the new predicate Inj has an explicit definition in the theory $\mathcal{ADP}_{\mathsf{inj}}$, every model for $\mathcal{ADP}$ extends uniquely to a model for $\mathcal{ADP}_{\mathsf{inj}}$ (see, e.g., van Dalen, 1989). Furthermore, *standard models of* $\mathcal{ADP}_{\mathsf{inj}}$ will be those models of $\mathcal{ADP}_{\mathsf{inj}}$ whose reduct is a standard model of $\mathcal{ADP}$.

In order to obtain a decision procedure for $\mathcal{ADP}_{\mathsf{inj}}$, it is necessary to find suitable extensions of Definitions 4.2.2 and 4.2.4 so that enough instances of (4.14) are considered and the results of the available decision procedures for $\mathcal{A}$ and $\mathcal{P}$ are

conclusive about the satisfiability of the original constraint in the extended theory. We formalize the meaning of "enough instances" for this extension of $\mathcal{ADP}$ in the following two definitions.

Definition 4.3.1 ($\mathcal{E}_{\mathsf{inj}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{E}_{\mathsf{inj}}$-instantiation closed if and only if L is $\mathcal{E}$-instantiation closed (cf. Definition 4.2.2) and the following condition is satisfied:

1. if $\neg\mathsf{Inj}(a) \in L$, then $\{\mathsf{select}(a,i) = e, \mathsf{select}(a,j) = e, i < j, e \neq \bot\} \subseteq L$ for some constants $e : \text{ELEM}, i, j : \text{INDEX}$.

Definition 4.3.2 ($\mathcal{G}_{\mathsf{inj}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{G}_{\mathsf{inj}}$-instantiation closed if and only if L is $\mathcal{G}$-instantiation closed (cf. Definition 4.2.4) and the following conditions are satisfied:

1. if $\mathsf{Inj}(a) \in L$ then, for each constant i of sort INDEX occurring in L, $\mathsf{select}(a,i) = \bot \in L$ or $\{\mathsf{select}(a,i) = e, e \neq \bot\} \subseteq L$ for some constant $e : \text{ELEM}$;

2. if $\{\mathsf{Inj}(a), i < j, \mathsf{select}(a,i) = e_1, \mathsf{select}(a,j) = e_2, e_1 \neq \bot, e_2 \neq \bot\} \subseteq L$, then $e_1 \neq e_2 \in L$.

Lemmas 4.2.3 and 4.2.5 can be easily adapted to the theory $\mathcal{ADP}_{\mathsf{inj}}$, taking into consideration the additional requirements of Definitions 4.3.1 and 4.3.2. A decision procedure $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{inj}}}$ for $\mathcal{ADP}_{\mathsf{inj}}$ can be obtained from $\mathsf{DP}_{\mathcal{ADP}}$ by replacing the modules for $\mathcal{E}$- and $\mathcal{G}$-instantiation in Figure 4.1 with those taking into account Definitions 4.3.1 and 4.3.2. We are now ready to state and prove the correctness of $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{inj}}}$.

Theorem 4.3.3. $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{inj}}}$ *is a decision procedure for the* $\mathcal{ADP}_{\mathsf{inj}}$*-satisfiability problem. Furthermore,* $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{inj}}}$ *decides the constraint satisfiability problem in the standard models of* $\mathcal{ADP}_{\mathsf{inj}}$.

Proof. Soundness is trivial. Regarding the key point for completeness, suppose we are given an $\mathcal{E}_{\mathsf{inj}}$- and a $\mathcal{G}_{\mathsf{inj}}$-instantiation closed finite set of literals

$$L = L_{\mathcal{ADP}} \cup L_{\mathsf{inj}}$$

(here L_{inj} is the set of literals from L involving the predicate Inj) such that $L_{\mathcal{A}}$ is $\mathcal{A}$-consistent and $L_{\mathcal{P}}$ is $\mathcal{P}$-consistent. The construction of Lemma 4.2.7 yields a standard model $\mathcal{M}$ of $\mathcal{ADP}$ satisfying $L_{\mathcal{ADP}}$. We are left to prove that the expansion

of $\mathcal{M}$ to a model of $\mathcal{ADP}_{\mathsf{inj}}$ is a model of L_{inj}. But this is easy by Definitions 4.3.1 and 4.3.2. □

4.3.2 Arrays with Domain

We equip arrays with a function computing their domain, i.e. the set of indexes at which they store "defined" values, i.e. values distinct from $\bot$. To this end, we need to formalize a very simple theory of sets of indexes, which is a straightforward extension of that used in Armando et al. (2003). Let $\mathcal{S}^{\emptyset}$ be the theory whose sort symbols are BOOL and SET, whose function symbols are $\mathsf{true}, \mathsf{false} : \text{BOOL}$, $\emptyset : \text{SET}$, $\mathsf{mem} : \text{INDEX} \times \text{SET} \to \text{BOOL}$, $\mathsf{ins} : \text{INDEX} \times \text{SET} \to \text{SET}$, and whose axioms are the following:

$$\mathsf{mem}(i, \emptyset) = \mathsf{false} \tag{4.15}$$

$$\mathsf{mem}(i, \mathsf{ins}(i, s)) = \mathsf{true} \tag{4.16}$$

$$i_1 \neq i_2 \to \mathsf{mem}(i_1, \mathsf{ins}(i_2, s)) = \mathsf{mem}(i_1, s) \tag{4.17}$$

$$\mathsf{true} \neq \mathsf{false} \wedge \forall x : \text{BOOL}\ (x = \mathsf{true} \vee x = \mathsf{false}) \tag{4.18}$$

where i, i_1, i_2 are implicitly universally quantified variables of sort INDEX and s is an implicitly universally quantified variable of sort SET. Moreover, we will call $\mathcal{S}^{\emptyset}_{-}$ the theory given by the axioms (4.15), (4.16), and (4.17).

Intuitively, $\emptyset$ denotes the empty set, mem is the test for membership of an index to a set, ins adds an index to a set if it is not already in the set. The constants true and false allow us to encode the membership predicate with the Boolean valued function mem. It is possible to use Lemma 4.4.2 below to see that the constraint satisfiability problem for $\mathcal{S}^{\emptyset}$ is decidable (by Superposition Calculus).[2] Hence, from now on, we consider the availability of a decision procedure for the constraint satisfiability problem of $\mathcal{S}^{\emptyset}$ in addition to those for $\mathcal{A}$ and $\mathcal{P}$.

Since we want to be able to compare sets by using the membership predicate mem, we need to consider the theory $\mathcal{S}^{\emptyset}_{e}$ obtained from $\mathcal{S}^{\emptyset}$ by adding the following

[2] Indeed, given a $\Sigma^{a}_{\mathcal{S}^{\emptyset}}$-constraint L, we produce the set of clauses $I^{\star}_{L}$ by instantiating the axiom (4.18) to each constant of sort BOOL occurring in L; clearly $\mathcal{S}^{\emptyset} \cup L$ and $\mathcal{S}^{\emptyset}_{-} \cup I^{\star}_{L} \cup L$ are equisatisfiable. A straightforward inspection of the clauses produced in the saturation process given in the proof of Lemma 4.4.2 allows to conclude that $\mathcal{SP}$ terminates on $\mathcal{S}^{\emptyset}_{-} \cup I^{\star}_{L} \cup L$, hence the decidability result obtains.

axiom of extensionality for sets:

$$\forall i(\mathsf{mem}(i, s_1) = \mathsf{mem}(i, s_2)) \rightarrow s_1 = s_2 \tag{4.19}$$

where s_1, s_2 are implicitly universally quantified variables of sort SET. The *standard models* of the theory $\mathcal{S}_e^{\emptyset}$ are the models in which the sort SET is interpreted as the set of (characteristic functions of) finite subsets of the interpretation of the sort INDEX.

We are now in the position to give a precise definition of the extension of $\mathcal{ADP}$ by the domain function for arrays. Let $\mathcal{ADP}_{\mathsf{dom}}$ be the theory obtained by extending the (disjoint) union of $\mathcal{ADP}$ with $\mathcal{S}_e^{\emptyset}$ by the function symbol dom : ARRAY $\rightarrow$ SET together with the following axiom:

$$\mathsf{select}(a, i) = \bot \ \leftrightarrow \ \mathsf{mem}(i, \mathsf{dom}(a)) = \mathsf{false} \tag{4.20}$$

where i and a are implicitly universally quantified variables of sort INDEX and ARRAY, respectively. Again, notice that a standard model of $\mathcal{ADP} \cup \mathcal{S}_e^{\emptyset}$ can be expanded in a unique way to a model (called *standard* as well) of $\mathcal{ADP}_{\mathsf{dom}}$.

In order to obtain a decision procedure for the constraint satisfiability problem of $\mathcal{ADP}_{\mathsf{dom}}$, it is necessary to find suitable extensions of Definitions 4.2.2 and 4.2.4 so that enough instances of axioms (4.19) and (4.20) are considered and the results of the available decision procedures for the constraint satisfiability problems of $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{S}^{\emptyset}$ are conclusive about the satisfiability of the original constraint in the extended theory. We formalize the meaning of "enough instances" for axioms (4.19) and (4.20) in the following definitions.

Definition 4.3.4 ($\mathcal{E}_{\mathsf{set}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{E}_{\mathsf{set}}$-instantiation closed if and only if L is $\mathcal{E}$-instantiation closed (cf. Definition 4.2.2) and the following condition is satisfied:

1. if $s_1 \neq s_2 \in L$, with s_1, s_2 constants of sort SET, then $\{\mathsf{mem}(i, s_1) = b_1, \mathsf{mem}(i, s_2) = b_2, b_1 \neq b_2\} \subseteq L$ for some constants b_1, b_2 : BOOL, i : INDEX.

Definition 4.3.5 ($\mathcal{G}_{\mathsf{dom}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{G}_{\mathsf{dom}}$-instantiation closed if and only if L is $\mathcal{G}$-instantiation closed (cf. Definition 4.2.4) and the following conditions are satisfied:

1. if a literal of the kind $\mathsf{dom}(a) = s_a$ belongs to L, then for each constant i of sort INDEX occurring in L, $\mathsf{select}(a, i) = \bot \in L$ or $\{\mathsf{select}(a, i) = e, e \neq \bot\} \subseteq L$ for some constant e : ELEM;

2. if $\{\mathsf{select}(a,i) = \bot, \mathsf{dom}(a) = s_a\} \subseteq L$ then $\{\mathsf{mem}(i,s_a) = b, b \neq \mathsf{true}\} \subseteq L$ for some constant b : BOOL; otherwise, if $\{\mathsf{select}(a,i) = e, e \neq \bot, \mathsf{dom}(a) = s_a\} \subseteq L$ then $\mathsf{mem}(i,s_a) = \mathsf{true} \in L$;

Lemmas 4.2.3 and 4.2.5 can be easily adapted to the theory $\mathcal{ADP}_{\mathsf{dom}}$. The decision procedure $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{dom}}}$ for the theory $\mathcal{ADP}_{\mathsf{dom}}$ is obtained from $\mathsf{DP}_{\mathcal{ADP}}$ by (i) replacing the modules for $\mathcal{E}$- and $\mathcal{G}$-instantiation in Figure 4.1 with those taking into account Definitions 4.3.4 and 4.3.5 and by (ii) adding the decision procedure for $\mathcal{S}^{\emptyset}$ to the set of decision procedures available to the schema in Algorithm 3, i.e. by setting T to $\{\mathcal{A}, \mathcal{P}, \mathcal{S}^{\emptyset}\}$.

Theorem 4.3.6. $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{dom}}}$ *is a decision procedure for the* $\mathcal{ADP}_{\mathsf{dom}}$*-satisfiability problem. Furthermore,* $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{dom}}}$ *decides the satisfiability problem in the standard models of* $\mathcal{ADP}_{\mathsf{dom}}$.

Since the arguments used in the proof of the theorem above are quite similar to the ones used in Theorem 4.2.8, we omit the proof.

4.3.3 Further Extensions of $\mathcal{ADP}$

To show the flexibility of our approach, we consider here some further extensions of $\mathcal{ADP}$ whose satisfiability problem can be checked by augmenting the decision procedure of Section 4.2 with suitable instantiation strategies. The extensions considered below are all relevant for applications as discussed, e.g., in Bradley et al. (2006). It is remarkable that the decision procedures for the constraint satisfiability problem for the various extensions considered below can simply be obtained by modifying the modules for $\mathcal{E}$-instantiation and $\mathcal{G}$-instantiation in Figure 4.1.

Prefixes

We consider the new binary predicate symbol $\sqsubseteq$: ARRAY $\times$ ARRAY and we extend the set of axioms of $\mathcal{ADP}$ by adding the following sentence:

$$a \sqsubseteq b \quad \leftrightarrow \quad \forall i(i < \mathsf{dim}(a) \rightarrow \mathsf{select}(a,i) = \mathsf{select}(b,i)) \tag{4.21}$$

where i is a variable of sort INDEX, a and b are implicitly universally quantified variables of sort ARRAY. We denote the extended theory with $\mathcal{ADP}_{\mathsf{pfx}}$. Intuitively, a is a prefix of b whenever $a \sqsubseteq b$ holds.

In order to obtain a decision procedure for the $\mathcal{ADP}_{\mathsf{pfx}}$-satisfiability problem, we need to extend the definitions of $\mathcal{E}$- and $\mathcal{G}$-instantiation closed sets of literals.

Definition 4.3.7 ($\mathcal{E}_{\mathsf{pfx}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{E}_{\mathsf{pfx}}$-instantiation closed if and only if L is $\mathcal{E}$-instantiation closed (cf. Definition 4.2.2) and the following conditions are satisfied:

1. if $a \not\sqsubseteq b \in L$, then $\{\mathsf{select}(a,i) = e_1, \mathsf{select}(b,i) = e_2, e_1 \neq e_2, i < d_a, d_a = \mathsf{dim}(a)\} \subseteq L$ for some constants $i, d_a : \textsc{index}, e_1, e_2 : \textsc{elem}$;
2. if $a \sqsubseteq b \in L$, then $d_a = \mathsf{dim}(a) \in L$ for some constant $d_a : \textsc{index}$.

Definition 4.3.8 ($\mathcal{G}_{\mathsf{pfx}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{G}_{\mathsf{pfx}}$-instantiation closed if and only if L is $\mathcal{G}$-instantiation closed (cf. Definition 4.2.4) and the following condition is satisfied:

1. if $\{a \sqsubseteq b, i < d_a, d_a = \mathsf{dim}(a)\} \subseteq L$ then $\{\mathsf{select}(a,i) = e, \mathsf{select}(b,i) = e\} \subseteq L$ for some constant $e : \textsc{elem}$.

A decision procedure $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{pfx}}}$ for the constraint satisfiability problem of $\mathcal{ADP}_{\mathsf{pfx}}$ can be obtained from $\mathsf{DP}_{\mathcal{ADP}}$ by simply replacing the modules for $\mathcal{E}$-instantiation and $\mathcal{G}$-instantiation in Figure 4.1 with those taking into account Definitions 4.3.7 and 4.3.8 above.

The soundness and completeness of $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{pfx}}}$ are obtained with arguments which are similar to that for injective arrays in Subsection 4.3.1. As a consequence, here we only state the main result without providing proofs.

Theorem 4.3.9. *$\mathsf{DP}_{\mathcal{ADP}_{\mathsf{pfx}}}$ is a decision procedure for the $\mathcal{ADP}_{\mathsf{pfx}}$-satisfiability problem. Furthermore, $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{pfx}}}$ decides the satisfiability problem in the standard models of $\mathcal{ADP}_{\mathsf{pfx}}$.*

Iterators

We consider two finite sets $\{mapf_1, \ldots, mapf_n\}$ and $\{f_1, \ldots, f_n\}$ of fresh unary function symbols such that $mapf_k : \textsc{array} \rightarrow \textsc{array}$ and $f_k : \textsc{elem} \rightarrow \textsc{elem}$ ($k \in \{1, \ldots, n\}$). We extend the set of axioms of $\mathcal{ADP}$ by adding a finite number of sentences of the following form:

$$\mathsf{select}(mapf_k(a), i) = f_k(\mathsf{select}(a, i)) \tag{4.22}$$

$$f_k(\bot) = \bot, \tag{4.23}$$

where i and a are implicitly universally quantified variables of sort INDEX and ARRAY, respectively ($k \in \{1, \ldots, n\}$). We denote the extended theory with $\mathcal{ADP}_{\mathsf{map}}$. Intuitively, $mapf_k(a)$ can be seen as an application of the higher-order function map, which is routinely used in many functional languages, such as ML or Haskell, i.e. $mapf_k(a)$ is equivalent to $(\mathsf{map}\ \ f_k\ \ a)$.

In order to obtain a decision procedure for the $\mathcal{ADP}_{\mathsf{map}}$-satisfiability problem, we need to extend the definition of of $\mathcal{E}$-instantiation closed set of literals.

Definition 4.3.10 ($\mathcal{G}_{\mathsf{map}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{G}_{\mathsf{pfx}}$-instantiation closed if and only if L is $\mathcal{G}$-instantiation closed (cf. Definition 4.2.4) and the following conditions are satisfied:

1. if $b = mapf_k(a) \in L$, then $\{\mathsf{select}(a, i) = e_1, f_k(e_1) = e_2, \mathsf{select}(b, i) = e_2\} \subseteq L$ for some constants $e_1, e_2 :$ ELEM;
2. $f_k(\bot) = \bot \in L$ ($k \in \{1, \ldots, n\}$).

A decision procedure $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{map}}}$ for $\mathcal{ADP}_{\mathsf{map}}$ can be obtained from $\mathsf{DP}_{\mathcal{ADP}}$ by replacing the module for $\mathcal{G}$-instantiation in Figure 4.1 with that taking into account the Definition 4.3.10 above and by extending the decision procedure for $\mathcal{A}$-satisfiability to cope with the uninterpreted function symbols f_k's which have been added to the signature of $\mathcal{ADP}$. This latter modification can be obtained for free in the rewriting-based approach to satisfiability procedures (as explained in Armando et al., 2003) or by combining *à la* Nelson-Oppen (see Nelson and Oppen, 1979; Tinelli and Harandi, 1996) the decision procedure for $\mathcal{A}$ with one for the theory of equality (see, e.g., Nelson and Oppen, 1980).

The soundness and completeness of $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{map}}}$ are obtained with arguments which are similar to that for injective arrays in Subsection 4.3.1. As a consequence, here we only state the main result without providing proofs.

Theorem 4.3.11. $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{map}}}$ *is a decision procedure for the* $\mathcal{ADP}_{\mathsf{map}}$*-satisfiability problem. Furthermore,* $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{map}}}$ *decides the satisfiability problem in the standard models of* $\mathcal{ADP}_{\mathsf{map}}$*.*

Sorting

We consider the new binary predicate symbol $\preceq$: ELEM $\times$ ELEM and we extend the axioms of $\mathcal{ADP}$ by adding sentences stating that $\preceq$ is a total order over the sort ELEM. We also add the unary predicate symbol Sorted over the sort ARRAY,

recognizing those arrays which are sorted in ascending order according to the total order $\preceq$ (with the exception of $\perp$ element). We also extend the set of axioms by adding the following sentence:

$$\mathsf{Sorted}(a) \leftrightarrow \forall i, j \Big(i < j \rightarrow \begin{pmatrix} \mathsf{select}(a,i) \preceq \mathsf{select}(a,j) & \vee \\ \mathsf{select}(a,i) = \perp & \vee \\ \mathsf{select}(a,j) = \perp & \end{pmatrix} \Big) \tag{4.24}$$

where a is an implicitly universally quantified variable of sort ARRAY.

In order to obtain a decision procedure for the $\mathcal{ADP}_{\mathsf{ord}}$-satisfiability problem, we need to extend the definitions of $\mathcal{E}$- and $\mathcal{G}$-instantiation closed sets of literals.

Definition 4.3.12 ($\mathcal{E}_{\mathsf{ord}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{E}_{\mathsf{ord}}$-instantiation closed if and only if L is $\mathcal{E}$-instantiation closed (cf. Definition 4.2.2) and the following condition is satisfied:

1. if $\neg\mathsf{Sorted}(a) \in L$, then $\{\mathsf{select}(a,i) = e_1, \mathsf{select}(a,j) = e_2, e_1 \neq \perp, e_2 \neq \perp, i < j, e_1 \not\preceq e_2\} \subseteq L$ for some constants $e_1, e_2 :$ ELEM, $i, j :$ INDEX.

Definition 4.3.13 ($\mathcal{G}_{\mathsf{ord}}$-instantiation closed set of literals)**.** A set L of ground flat literals is $\mathcal{G}_{\mathsf{ord}}$-instantiation closed if and only if L is $\mathcal{G}$-instantiation closed (cf. Definition 4.2.4) and the following conditions are satisfied:

1. if $\mathsf{Sorted}(a) \in L$ then, for each constant i of sort INDEX occurring in L, $\mathsf{select}(a,i) = \perp \in L$ or $\{\mathsf{select}(a,i) = e, e \neq \perp\} \subseteq L$ for some constant $e :$ ELEM;

2. if $\{\mathsf{Sorted}(a), \mathsf{select}(a,i) = e_1, \mathsf{select}(a,j) = e_2, i < j, e_1 \neq \perp, e_2 \neq \perp\} \subseteq L$, then $e_1 \preceq e_2 \in L$.

A decision procedure $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{ord}}}$ for $\mathcal{ADP}_{\mathsf{ord}}$ can be obtained from $\mathsf{DP}_{\mathcal{ADP}}$ by replacing the modules for $\mathcal{E}$- and $\mathcal{G}$-instantiation in Figure 4.1 with those taking into account the Definitions 4.3.12 and 4.3.13 above and by replacing the decision procedure for $\mathcal{A}$ with a decision procedure obtained by combining *à la* Nelson-Oppen (see Nelson and Oppen, 1979; Tinelli and Harandi, 1996) the decision procedure for $\mathcal{A}$ with one for the theory of total order (see, e.g., Bjørner et al., 1997).

The soundness and completeness of $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{ord}}}$ are obtained with arguments which are similar to that for injective arrays in Subsection 4.3.1. As a consequence, here we only state the main result without providing proofs.

Theorem 4.3.14. $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{ord}}}$ *is a decision procedure for the* $\mathcal{ADP}_{\mathsf{ord}}$*-satisfiability problem. Furthermore,* $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{ord}}}$ *decides the satisfiability problem in the standard models of* $\mathcal{ADP}_{\mathsf{ord}}$*.*

All the extensions considered above can be combined together in order to obtain a decidable fragment which is very expressive and able to cope with many properties of interest for the field of software verification.

4.4 Implementation Issues

The following section is devoted to address some of the problems arising in the implementation of the procedures presented above. The key issue is how to efficiently handle the non-determinism introduced by the various $\mathcal{G}$-instantiation modules considered above (cf. Definitions 4.2.4, 4.3.2 and 4.3.5). An ad hoc solution to this problem for the theory of arrays with domains will be given by using the rewriting-approach to build satisfiability procedures. Unfortunately, this solution is not general since, for example, the theory of arrays augmented with the injective axiom does not seem to be amenable to such an approach without resorting to suitable extensions of the calculus to handle cancellation axioms (see, e.g., Rusinowitch, 1989) which are not implemented in state-of-the-art provers. A more general solution, relying on the use of Satisfiability Modulo Theories solvers will then be described which is capable of coping with all the extensions considered above.

4.4.1 A Rewriting-based Procedure for $\mathcal{ADP}_{\mathsf{dom}}$

An alternative to the model-theoretic approach described in Subsection 4.3.2 is represented by the rewriting-approach to satisfiability procedures described in Armando et al. (2003), which allows us to better handle the non-determinism introduced by the guessing. In fact, we can use the Superposition Calculus (from now on denoted by $\mathcal{SP}$, see Nieuwenhuis and Rubio, 2001 and also the Appendix for a very brief overview) to build a decision procedure for the constraint satisfiability problem in the union of the theories $\mathcal{A}_e$ and $\mathcal{S}_e^{\emptyset}$ extended with axiom (4.20). Such a procedure is then combined with a decision procedure for the constraint satisfiability problem in $\mathcal{P}$ to build a decision procedure for $\mathcal{ADP}_{\mathsf{dom}}$.

In Armando et al. (2003), it is shown how to use $\mathcal{SP}$ to build decision procedures for the constraint satisfiability problem of theories axiomatized by a finite set of first-order clauses. The key observation is that, in order to show that $\mathcal{SP}$ is a decision

procedure, it is sufficient to prove that $\mathcal{SP}$ terminates on the set of clauses obtained by the union of the axioms of the theory and an arbitrary set of ground and flat literals. According to Armando et al. (2003), $\mathcal{SP}$ terminates also for some of the theories considered in this chapter, e.g., $\mathcal{A}$ and $\mathcal{S}^{\emptyset}_{-}$ (when considered in isolation). Modularity results in Armando et al. (2007) allow us to conclude that $\mathcal{SP}$ also terminates for the union $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-}$. Unfortunately, this is not enough since our goal is to build a decision procedure for the $\mathcal{ADP}_{\mathsf{dom}}$-satisfiability problem whose set of axioms also contains (4.18) and (4.20).

As a preliminary step to applying $\mathcal{SP}$, we need to partially instantiate axioms (4.18) and (4.20) with the constants of sort ARRAY and BOOL occurring in L. This is so because $\mathcal{SP}$ does not seem to terminate on theories axiomatizing enumerated datatypes such as the Booleans (see Bonacina et al., 2006 for a discussion on this point).

Definition 4.4.1. Let L be a set of ground and flat $\Sigma_{\mathcal{A}_e \cup \mathcal{S}^{\emptyset}}$-literals; we define $\mathcal{I}_L$ to be the following set of (partial) instances of axioms (4.18) and (4.20):

$$\begin{aligned} \mathsf{select}(a,x) \neq \bot \vee \mathsf{mem}(x, \mathsf{dom}(a)) \neq \mathsf{true} \\ \mathsf{select}(a,x) = \bot \vee \mathsf{mem}(x, \mathsf{dom}(a)) = \mathsf{true} \\ b = \mathsf{true} \vee b = \mathsf{false} \\ \mathsf{true} \neq \mathsf{false} \end{aligned}$$

for each $\mathsf{dom}(a) = s$ in L and for each constant b : BOOL occurring in L.

Along the lines of Armando et al. (2003), to build a decision procedure for the $\mathcal{ADP}_{\mathsf{dom}}$-satisfiability problem it is necessary to show that $\mathcal{SP}$ terminates on the class of clauses obtained by the union of ground flat literals and the axioms which have not been completely instantiated, namely those in $\mathcal{A}$, $\mathcal{S}^{\emptyset}_{-}$, and those in $\mathcal{I}_L$.

Lemma 4.4.2. *$\mathcal{SP}$ terminates on $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L \cup L$ for every finite set L of ground and flat $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-literals.*

Proof. Let L be a set of ground and flat $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-literals. The clauses in the saturation of $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L \cup L$ by $\mathcal{SP}$ can only be of type

i) the empty clause;

ii) the clauses in $\mathcal{A}$, i.e.

a) $\mathsf{select}(\mathsf{store}(x, y, z), y) = z$;

b) $\mathsf{select}(\mathsf{store}(x, y, z), w) = \mathsf{select}(x, w) \vee y = w$;

iii) clauses of the following kind ($n, m \geq 0$):

a) $\mathsf{select}(a, x) = \mathsf{select}(a', x) \vee x = i_1 \vee \cdots \vee x = i_n \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

b) $\mathsf{select}(a, i) = e \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

c) $\mathsf{store}(a, i, e) = a' \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

d) $a \bowtie a' \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

e) $e \bowtie e' \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

f) $j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

derived by considering only $\mathcal{A} \cup L$, or of type

i') the empty clause;

ii') the clauses in $\mathcal{S}^{\emptyset}_{-}$, i.e.

a) $\mathsf{mem}(x, \mathsf{ins}(x, z)) = \mathsf{true}$;

b) $\mathsf{mem}(x, \mathsf{ins}(y, z)) = \mathsf{mem}(x, z) \vee x = y$;

c) $\mathsf{mem}(x, \emptyset)) \neq \mathsf{true}$;

iii') clauses of the following kind ($n, m \geq 0$):

a) $\mathsf{mem}(x, s) = \mathsf{mem}(x, s') \vee x = i_1 \vee \cdots \vee x = i_n \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

b) $\mathsf{mem}(i, s) = b \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

c) $\mathsf{ins}(i, s) = s' \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

d) $s \bowtie s' \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

e) $b \bowtie b' \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

f) $j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

derived by considering only $\mathcal{S}^{\emptyset}_{-} \cup L$, or of type

iv) clauses in $\mathcal{I}_L$ and constraint involving the function dom:

a) $\mathsf{select}(a, x) \neq \bot \vee \mathsf{mem}(x, \mathsf{dom}(a)) \neq \mathsf{true}$;

b) $\mathsf{select}(a, x) = \bot \vee \mathsf{mem}(x, \mathsf{dom}(a)) = \mathsf{true}$;

c) $b = \mathsf{true} \vee b = \mathsf{false}$;

d) $\mathsf{false} \neq \mathsf{true}$;

e) $\mathsf{dom}(a) = s$.

v) non-unit clauses:

a) $\mathsf{select}(a, x) \neq \bot \vee \mathsf{mem}(x, t) \neq \mathsf{true} \vee x = i_1 \vee \cdots \vee x = i_n \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t is either s or $\mathsf{dom}(a')$;

b) $\mathsf{select}(a, x) = \bot \vee \mathsf{mem}(x, t) = \mathsf{true} \vee x = i_1 \vee \cdots \vee x = i_n \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t is either s or $\mathsf{dom}(a')$;

c) $\mathsf{mem}(x, t_1) \neq \mathsf{true} \vee \mathsf{mem}(x, t_2) = \mathsf{true} \vee x = i_1 \vee \cdots \vee x = i_n \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t_i is either s_i or $\mathsf{dom}(a_i)$;

d) $\mathsf{select}(a, x) \neq \bot \vee \mathsf{select}(a', x) = \bot \vee x = i_1 \vee \cdots \vee x = i_n \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

vi) non-unit ground clauses:

a) $t_1 \neq \bot \vee t_2 \neq \mathsf{true} \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t_1 is either e or $\mathsf{select}(a, i)$ and t_2 is b or $\mathsf{mem}(i, s)$ or $\mathsf{mem}(i, \mathsf{dom}(a))$;

b) $t_1 = \bot \vee t_2 = \mathsf{true} \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t_1 is either e or $\mathsf{select}(a, i)$ and t_2 is b or $\mathsf{mem}(i, s)$ or $\mathsf{mem}(i, \mathsf{dom}(a))$;

c) $t_1 \neq \mathsf{true} \vee t_2 = \mathsf{true} \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t_k is b_k or $\mathsf{mem}(i, s_k)$ or $\mathsf{mem}(i, \mathsf{dom}(a_k))$;

d) $t_1 \neq \bot \vee t_2 = \bot \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where t_k is either e_k or $\mathsf{select}(a_k, i)$;

e) $e \neq \bot \vee b = \mathsf{true} \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

f) $e_1 \neq \bot \vee e_2 \neq \bot \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$;

g) $b_1 \bowtie v_1 \vee b_2 \bowtie v_2 \vee \underbrace{\mathsf{false} = \mathsf{true} \vee \cdots \vee \mathsf{false} = \mathsf{true}}_{k \text{ times}} \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ where $v_1, v_2 \in \{\mathsf{true}, \mathsf{false}\}$ and $k \leq 16N^2$ (N is the number of constant symbols of sort BOOL);

h) $\mathsf{dom}(a) = s \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$.

derived by considering the instances of axioms (4.18) and (4.20) in $\mathcal{I}_L$.

In fact, by termination results in Armando et al. (2003), the clauses that can be generated by the exhaustive application of the rules of $\mathcal{SP}$ to $\mathcal{A} \cup L$ and to $\mathcal{S}^{\emptyset}_{=} \cup L$

can only be of type i) - iii) and i') - iii'), respectively.[3] It is clear that no more clauses will be generated by the saturation process between clauses of the kind (i),(ii),(iii) and (i'),(ii'),(iii').

Let us consider the clauses of type iv), i.e. instances of the axioms (4.18) and (4.20) and constraints involving the function dom. The superposition calculus can only generate clauses of types v) and vi) as shown below:

- *Inferences between a clause in iv) and a clause in ii) or iii)*: inferences between iv.a) and iii.a), iii.b), iii.d) give clauses respectively in v.a), vi.a), and v.a); inferences between iv.b) and iii.a), iii.b), iii.d) give clauses respectively in v.b), vi.b), and v.b); inferences between iv.e) and iii.d) give a clause in vi.h).

- *Inferences between a clause in iv) and a clause in ii') or iii')*: inferences between iv.c) and iii'.e) give a clause in vi.g); inferences between iv.d) and iii'.e) give a clause in iii'.f): in fact, because of the ordering, the literal $b \bowtie b'$ in iii'.e) have to be equal to $\mathsf{false} = \mathsf{true}$, thus the inference would give a clause of the kind $\mathsf{true} \neq \mathsf{true} \vee j_1 \bowtie j'_1 \vee \cdots \vee j_m \bowtie j'_m$ which will be immediately simplified by the *Deletion* rule into a clause of the kind iii'.f) because of the chosen strategy.

- *Inferences between a clause in iv) and a clause in iv)*: inferences between iv.a) and iv.b) give a clause in v.c) or v.d) (depending on the chosen ordering on the function symbols); inferences between iv.e) and iv.a), iv.b) give a clause in v.a) and v.b) respectively; inferences between iv.e) and itself give a clause in iii'.d).[4]

- *Inferences between a clause in iv) and a clause in v)*: inferences between iv.a) and v.b), v.c), v.d) give clauses respectively in v.c) or v.d) (again depending on the chosen ordering on the function symbols), v.a), and v.a); inferences between iv.b) and v.a), v.c), v.d) give clauses respectively in v.c) or v.d), v.b), and v.b); inferences between iv.e) and v) give clauses which are still in v).

- *Inferences between a clause in iv) and a clause in vi)*: inferences between iv.a) and vi.b), vi.c), vi.d), iv.f) give clauses respectively in vi.c) or vi.d), vi.a), vi.a), and vi.a); inferences between iv.b) and vi.a), vi.c), vi.d), vi.h)

[3] It is easy to see that the saturation with the clause ii'.c) do not generate any additional clause since $\emptyset$ is minimal in the chosen precedence.

[4] Notice that no inference between iv.c) and iv.d) can occur because, if so, the constant b should be equal to false, thus the clause iv.c) would have already been removed by the *Deletion* rule.

give clauses respectively in vi.c) or vi.d), vi.b), vi.b), and vi.b); inferences between iv.c) and vi.c) give clauses which are in vi.g) (in this case, t_1, t_2 have to be constant symbols); inferences between iv.c) and vi.g) give clauses which are still in vi.g); inferences between iv.d) and vi.g) give clauses in vi.g) or iii'.e): in fact, if an inference between iv.d) and vi.g) occur, then, because of the ordering, vi.g) have to be a clause of the kind $\mathsf{false} = \mathsf{true} \vee \mathsf{false} = \mathsf{true} \vee \cdots \vee j_1 \bowtie j_1' \vee \cdots \vee j_m \bowtie j_m'$, thus a clause with one less occurrence of the literal $\mathsf{false} = \mathsf{true}$ will be produced (see inferences between iv.d) and iii'.e) above); inferences between iv.e) and vi) give clauses which are still in vi).[5]

– *Inferences between a clause in v) and a clause in ii) or iii)*: inferences between v.a) and iii.a), iii.b), iii.d) give clauses respectively in v.a), vi.a), and v.a); inferences between v.b) and iii.a), iii.b), iii.d) give clauses respectively in v.b), vi.b), and v.b); inferences between clauses in v.d) and iii.a), iii.b), iii.d) give clauses respectively in v.d), vi.d), and v.d).

– *Inferences between a clause in v) and a clause in ii') or iii')*: inferences between v.a) and iii'.a), iii'.b), iii'.d) give clauses respectively in v.a), vi.a), and v.a); inferences between v.b) and iii'.a), iii'.b), iii'.d) give clauses respectively in v.b), vi.b), and v.b); inferences between v.c) and iii'.a), iii'.b), iii'.d) give clauses respectively in v.c), vi.c), and v.c).

– *Inferences between a clause in v) and a clause in v)*: all the clauses produced are still in v).

– *Inferences between a clause in v) and a clause in vi)*: inferences between v) and vi.h) give clauses in v); all other inferences still give clauses in vi).

– *Inferences between a clause in vi) and a clause in ii) or iii)*: inferences between vi.a) and iii.a), iii.b), iii.d), iii.e) give clauses in vi.a); inferences between vi.b) and iii.a), iii.b), iii.d) give clauses in vi.b); inferences between vi.b) and iii.e) give clauses in vi.b) or in vi.e) depending of the sign of the literal $e \bowtie e'$ in iii.e) (notice that if $e = \bot$ is maximal in vi.b), then t_2 have to be a constant of sort BOOL); inferences between vi.d) and iii.a), iii.b), iii.d) give clauses in vi.d); inferences between vi.d) and iii.e) give clauses in vi.d) or in vi.f) again depending of the sign of the literal $e_1 \bowtie e_2$ in iii.e); inferences between vi.e)

[5] Notice that no inference can occur between iv.d) and vi.c) because a literal of the kind $\mathsf{false} = \mathsf{true}$ cannot be maximal in vi.c); the same argument apply also to clauses of the kind vi.e).

and iii.e) still remain in vi.e); inferences between vi.f) and iii.e) still remain in vi.f).

– *Inferences between a clause in vi) and a clause in ii') or iii')*: inferences between vi.a) and iii'.a), iii'.b), iii'.d), iii'.e) give clauses in vi.a); inferences between vi.b) and iii'.a), iii'.b), iii'.d), iii'.e) give clauses in vi.b) (notice that no inference can occur between vi.b) and iii'.e) if the sign of $b \bowtie b'$ in iii'.e) is negative because the literal $t_2 = \mathsf{true}$ cannot be maximal in vi.b) if t_2 is a constant of sort BOOL); inferences between vi.c) and iii'.a), iii'.b), iii'.d) give clauses in vi.c); inferences between vi.c) and iii'.e) give clauses in vi.g); inferences between vi.g) and iii'.e) give clauses in vi.g).

– *Inferences between a clause in vi) and a clause in vi)*: all the clauses produced are still in vi) with the exception of the inferences between vi.h) and itself which give clauses of the kind iii'.d).

Actually, a clause of the form $C' := \bot \neq \bot \vee C$ (resp. $C' := \mathsf{true} \neq \mathsf{true} \vee C$) will be produced in many of the cases considered above. However, notice that an application of the *Reflection* rule produces the clause C which immediately subsumes C' (because of the strategy gives higher priority to the simplification rules). Moreover, since the precedence of the constant $\bot$ (resp. true) is less than all other constant of sort ELEM (resp. BOOL), it is clear that any clause derived from C' is subsumed by the clause obtained applying the same derivation from C. □

According to the rewriting approach of Armando et al. (2003), we can immediately conclude that $\mathcal{SP}$ behaves as a satisfiability procedure for $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L$, because of the refutation completeness of $\mathcal{SP}$.

Let us call $\mathcal{ASD}$ the theory axiomatized by $\mathcal{A}_e \cup \mathcal{S}^{\emptyset}_{e} \cup \{(4.20)\}$. The following lemma is needed to prove the correctness of the decision procedure for $\mathcal{ADP}_{\mathsf{dom}}$.

Lemma 4.4.3. *Let L be an $\mathcal{E}_{\mathsf{set}}$-instantiation closed set of $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-literals. Then, L is $\mathcal{ASD}$-satisfiable if and only if L is $(\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L)$-satisfiable.*

Proof. The 'only if' case is trivial, since the theory $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-}$ is a subtheory of $\mathcal{ASD}$, and the sentences in $\mathcal{I}_L$ are obtained by instantiating the axioms (4.18), (4.20).

For the 'if' case, suppose that L is satisfied in a model $\mathcal{M}'$ of $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L$. We define the following binary relation $\sim$

– to hold over $\text{ARRAY}^{\mathcal{M}'}$ whenever $\mathsf{select}^{\mathcal{M}'}(a_1, i) = \mathsf{select}^{\mathcal{M}'}(a_2, i)$ for all $i \in \text{INDEX}^{\mathcal{M}'}$;

- to hold over $\text{SET}^{\mathcal{M}'}$ to hold whenever, for all $i \in \text{INDEX}^{\mathcal{M}'}$, $\mathsf{mem}^{\mathcal{M}'}(s_1, i) = true^{\mathcal{M}'}$ if and only if $\mathsf{mem}^{\mathcal{M}'}(s_2, i) = true^{\mathcal{M}'}$;
- to coincide over the sort $\text{BOOL}^{\mathcal{M}'}$ with the equivalence relation induced by the partition into the two subset $\{true^{\mathcal{M}'}\}, \text{BOOL}^{\mathcal{M}'} \setminus \{true^{\mathcal{M}'}\}$;
- to coincide with the identity relation over each domain of $\mathcal{M}'$ different from $\text{ARRAY}^{\mathcal{M}'}$, $\text{SET}^{\mathcal{M}'}$ and $\text{BOOL}^{\mathcal{M}'}$.

The relation $\sim$ is clearly an equivalence relation; we now show that it is a $\Sigma_{\mathcal{A} \cup \mathcal{S}^{\emptyset}}$-congruence. It is straightforward to verify that $\sim$ is faithful with respect to $\mathsf{select}^{\mathcal{M}'}$, $\mathsf{store}^{\mathcal{M}'}$, and $\mathsf{mem}^{\mathcal{M}'}$. Given $s_1 \sim s_2$, we need to show that, for all $j \in \text{INDEX}^{\mathcal{M}'}$, $\mathsf{ins}^{\mathcal{M}'}(j, s_1) \sim \mathsf{ins}^{\mathcal{M}'}(j, s_2)$. This condition is easily verified, recalling the definition of $\sim$ and the fact that $\mathcal{M}'$ is a model for the axioms (4.16) and (4.17).

We set $\mathcal{M} := \mathcal{M}'/_{\sim}$. It is simple to check that $\mathcal{M} \models \mathcal{A}_e \cup \mathcal{S}_e^{\emptyset}$. We fix the interpretation in $\mathcal{M}$ of the function dom as follows: for each element $a \in \text{ARRAY}^{\mathcal{M}}$, $\mathsf{dom}^{\mathcal{M}}(a)$ is the set such that, for every $i \in \text{INDEX}^{\mathcal{M}}$, $\mathsf{mem}^{\mathcal{M}}(i, \mathsf{dom}^{\mathcal{M}}(a)) = true^{\mathcal{M}}$ if and only if $\mathsf{select}^{\mathcal{M}}(a, i) = \bot^{\mathcal{M}}$. The definition of $\sim$ is sufficient to verify that $\mathsf{dom}^{\mathcal{M}}$ is well-defined; moreover, the definition of $\mathsf{dom}^{\mathcal{M}}$ clearly implies that $\mathcal{M} \models$ (4.20), thus $\mathcal{M}$ is a model of $\mathcal{ADP}_{\mathsf{dom}}$.

To conclude the proof, it is sufficient to check that $\mathcal{M} \models L$. By construction, all the equalities between constants over the sort INDEX, ELEM, ARRAY, BOOL, and SET hold in $\mathcal{M}$; moreover, all the literals of the kind $\mathsf{select}(a, i) = e$, $\mathsf{store}(a, i, e) = b$, $\mathsf{mem}(i, s) = e$, and $\mathsf{ins}(i, s_1) = s_2$ are satisfied for the same reason. Inequalities between constant of sort ARRAY are verified because L is $\mathcal{E}$-instantiation closed, while inequalities between contents of sort ELEM and INDEX trivially hold.

If a literal of the kind $b_1 \neq b_2$ is in L (b_1, b_2 constants of sort BOOL), then, since $\mathcal{I}_L$ contains the clauses $b_1 = \mathsf{true} \vee b_1 = \mathsf{false}$ and $b_2 = \mathsf{true} \vee b_2 = \mathsf{false}$, we can freely suppose that $\mathcal{M} \models b_1 = \mathsf{true}$ and $\mathcal{M} \models b_2 = \mathsf{false}$, thus $\mathcal{M} \models b_1 \neq b_2$. If a literal of the kind $s_1 \neq s_2$ is in L (s_1, s_2 constants of sort SET), then, since L is an $\mathcal{E}_{\mathsf{set}}$-instantiation closed set of literals, $L' := \{\mathsf{mem}(s_1, i) = b_1, \mathsf{mem}(s_2, i) = b_2, b_1 \neq b_2\} \subseteq L$ for some constants $b_1, b_2 :$ BOOL and $i :$ INDEX, thus clearly $\mathcal{M} \models L'$ (see above) and so $\mathcal{M} \models s_1 \neq s_2$.

Finally, if the literal $\mathsf{dom}(a) = s$ is in L, $\mathcal{I}_L$ contain the clauses $\{\mathsf{select}(a, x) \neq \bot \vee \mathsf{mem}(x, \mathsf{dom}(a)) \neq \mathsf{true}, \mathsf{select}(a, x) = \bot \vee \mathsf{mem}(x, \mathsf{dom}(a)) = \mathsf{true}\}$. Since $\mathcal{M}' \models \mathcal{I}_L \cup L$ then, for each $i \in \text{INDEX}^{\mathcal{M}'}$, $\mathsf{select}^{\mathcal{M}'}(a^{\mathcal{M}'}, i) \neq \bot^{\mathcal{M}'}$ if and only if $\mathsf{mem}^{\mathcal{M}'}(i, s^{\mathcal{M}'}) = \mathsf{true}^{\mathcal{M}'}$. By construction we have that, for each $i \in \text{INDEX}^{\mathcal{M}}$,

$\mathsf{select}^{\mathcal{M}}(a^{\mathcal{M}}, i) \neq \bot^{\mathcal{M}}$ if and only if $\mathsf{mem}^{\mathcal{M}}(i, s^{\mathcal{M}}) = \mathsf{true}^{\mathcal{M}}$, which precisely means that $\mathsf{dom}^{\mathcal{M}}(a^{\mathcal{M}}) = s^{\mathcal{M}}$. □

Below, we denote with $\mathsf{DP}_{\mathcal{SP}}$ the function taking a set L of ground $\Sigma_{\mathcal{A}\cup\mathcal{S}^{\emptyset}}$-literals, computing $\mathcal{I}_L$ and then invoking $\mathcal{SP}$ on the clauses $\mathcal{A} \cup \mathcal{S}^{\emptyset}_{-} \cup \mathcal{I}_L \cup L$. If the empty clause is derived by $\mathcal{SP}$, then $\mathsf{DP}_{\mathcal{SP}}$ returns *"unsatisfiable"*; otherwise, it returns *"satisfiable"*. Hence, the new variant of the decision procedure $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{dom}}}$ for the theory $\mathcal{ADP}_{\mathsf{dom}}$ can be obtained from $\mathsf{DP}_{\mathcal{ADP}}$ by replacing the module for $\mathcal{E}$-instantiation in Figure 4.1 with a module for $\mathcal{E}_{\mathsf{set}}$-instantiation (cf. Definition 4.3.4) and by invoking $\mathsf{DP}_{\mathcal{SP}}$ and $\mathsf{DP}_{\mathcal{P}}$ in Algorithm 3, i.e. by setting T to $\{\mathcal{SP}, \mathcal{P}\}$.

Now, we can state and prove the correctness of the new version of $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{dom}}}$.

Theorem 4.4.4. $\mathsf{DP}_{\mathcal{ADP}_{\mathsf{dom}}}$ *is a decision procedure for the* $\mathcal{ADP}_{\mathsf{dom}}$*-satisfiability problem.*

Proof. According to the result in Theorem 4.3.6, an $\mathcal{E}_{\mathsf{set}}$- and a $\mathcal{G}_{\mathsf{dom}}$-instantiation closed finite set of literals

$$L = L_{\mathcal{ADP}} \cup L_{\mathcal{S}^{\emptyset}} \cup L_{\mathsf{dom}} \tag{4.25}$$

is $\mathcal{ADP}_{\mathsf{dom}}$-satisfiable whenever $L_{\mathcal{A}}$, $L_{\mathcal{P}}$ and $L_{\mathcal{S}^{\emptyset}}$ are $\mathcal{A}$-, $\mathcal{P}$- and $\mathcal{S}^{\emptyset}$-satisfiable, respectively (here L_{dom} is the set of literals from L involving the function dom). From now on, we assume that the set of literals (4.25) is only $\mathcal{E}_{\mathsf{set}}$- and a $\mathcal{G}$-instantiation closed. We still assume that $L_{\mathcal{P}}$ is $\mathcal{P}$-satisfiable and (this is the new fact due to Lemma 4.4.3) that $L_{\mathcal{A}} \cup L_{\mathcal{S}^{\emptyset}} \cup L_{\mathsf{dom}}$ is $\mathcal{ASD}$-satisfiable. Now, consider a model $\mathcal{M}$ of $\mathcal{ASD} \cup L_{\mathcal{A}} \cup L_{\mathcal{S}^{\emptyset}} \cup L_{\mathsf{dom}}$: looking at this model, we can add to $L_{\mathcal{A}} \cup L_{\mathcal{S}^{\emptyset}}$ more literals true in $\mathcal{M}$ (let them be $\hat{L}_{\mathcal{A}} \cup \hat{L}_{\mathcal{S}^{\emptyset}}$), in such a way that

$$\hat{L} = (L_{\mathcal{ADP}} \cup \hat{L}_{\mathcal{A}}) \cup (L_{\mathcal{S}^{\emptyset}} \cup \hat{L}_{\mathcal{S}^{\emptyset}}) \cup L_{\mathsf{dom}}$$

is $\mathcal{E}_{\mathsf{set}}$- and $\mathcal{G}_{\mathsf{dom}}$-instantiation closed (notice in fact that the newly introduced literals do not contain new constants of sort INDEX); now, $\hat{L} \supseteq L$ satisfies the requirements of Theorem 4.3.6 and is $\mathcal{ADP}_{\mathsf{dom}}$-satisfiable. □

4.4.2 An SMT-based algorithm

We present an algorithm which integrates our instantiation-based schema into an SMT solver by adapting the ideas described in Bozzano et al. (2006), where a

Algorithm 4 An SMT solver for $\mathcal{ADP}$-satisfiability

Require: φ quantifier-free $\Sigma^a_{\mathcal{ADP}}$-formula

procedure SMT(φ)
  $\varphi \leftarrow \mathit{flatten}(\varphi)$
  $\varphi \leftarrow \varphi \wedge e\text{-}\mathit{inst}(\varphi) \wedge g_2\text{-}\mathit{inst}(\varphi)$
  $\varphi \leftarrow \varphi \wedge g_{3,4}\text{-}\mathit{inst}(\varphi)$
  $\varphi^p \leftarrow \mathit{fol2prop}(\varphi)$
  while Bool-$\mathit{satisfiable}(\varphi^p)$ **do**
    $\beta^p \leftarrow \mathit{pick\_total\_assign}(\varphi^p)$
    $L^{\mathcal{G}} \leftarrow g_5\text{-}\mathit{inst}(\mathit{prop2fol}(\beta^p))$
    $(\rho_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathsf{DP}_{\mathcal{A}}(L^{\mathcal{G}}_{\mathcal{A}})$
    $(\rho_{\mathcal{P}}, \pi_{\mathcal{P}}) \leftarrow \mathsf{DP}_{\mathcal{P}}(L^{\mathcal{G}}_{\mathcal{P}})$
    if ($\rho_{\mathcal{A}} =$ *"satisfiable"* $\wedge\ \rho_{\mathcal{P}} =$ *"satisfiable"*) **then**
      return *"satisfiable"*
    else
      if $\rho_{\mathcal{A}} =$ *"unsatisfiable"* **then** $\varphi^p \leftarrow \varphi^p \wedge \neg \mathit{fol2prop}(\pi_{\mathcal{A}})$
      if $\rho_{\mathcal{P}} =$ *"unsatisfiable"* **then** $\varphi^p \leftarrow \varphi^p \wedge \neg \mathit{fol2prop}(\pi_{\mathcal{P}})$
    end if
  end while
  return *"unsatisfiable"*
end procedure

Boolean solver is used in order to efficiently handle the guessing phase of non-deterministic procedures.

The decision procedure described in Algorithm 4 relies on two simple functions. The former is a propositional abstraction function, i.e. a bijective function *fol2prop* which maps a ground first-order formula φ into a Boolean formula φ^p as follows: *fol2prop* maps Boolean atoms into themselves, ground atoms into fresh Boolean atoms, and is homomorphic with respect to Boolean operators. The second function, *prop2fol* (called, the refinement) is the inverse of *fol2prop*. In the following, the procedure DP_T is a decision procedure for the constraint satisfiability problem for T, where T is $\mathcal{A}$ or $\mathcal{P}$. If a constraint L is T-satisfiable, DP_T returns (*"satisfiable"*, $\emptyset$), otherwise it returns (unsat, π) where $\pi \subseteq L$ and π is a T-unsatisfiable set, called the (theory) conflict set. The associated (theory) conflict clause is $\neg\pi$.

The algorithm runs as follows. First of all, the function *flatten* transforms, by introducing sufficiently many fresh constants to name subterms, the input formula φ into an equisatisfiable formula of the kind $\varphi_u \wedge \varphi_s$, where φ_u is a constraint containing just positive flat equalities (including the literal $\mathsf{dim}(\varepsilon) = 0$) and φ_s is a Boolean

combination of equalities between constants. Then, according to Definitions 4.2.2 and 4.2.4, we add to the input formula φ some of its logical consequences with respect to $\mathcal{ADP}$. More in detail, we have

$$\varphi \longleftarrow \varphi \wedge e\text{-}inst(\varphi) \wedge g_2\text{-}inst(\varphi)$$
$$\varphi \longleftarrow \varphi \wedge g_{3,4}\text{-}inst(\varphi)$$

where the functions e-$inst$, g_2-$inst$ and $g_{3,4}$-$inst$ are such that

- e-$inst(\varphi)$ is the conjunction of the formulae $a \neq b \rightarrow (\mathsf{select}(a,i) = e_1 \wedge \mathsf{select}(b,i) = e_2 \wedge e_1 \neq e_2)$ for each constants $a, b : \text{ARRAY}$ such that $a = b$ occurs in φ (cf. Definition 4.2.2);

- g_2-$inst(\varphi)$ is the conjunction of the formulae $i = 0 \vee (e \neq \bot \wedge \mathsf{select}(a,j) = e \wedge \mathsf{s}(j) = i)$ for each constants $a : \text{ARRAY}, i : \text{INDEX}$ such that $\mathsf{dim}(a) = i$ occurs in φ (cf. Definition 4.2.4(2)); and

- $g_{3,4}$-$inst(\varphi)$ is the conjunction of the clauses of the kind $i < j \vee i = j \vee j < i$ for each constant $i, j : \text{INDEX}$ occurring in φ (cf. Definition 4.2.4(3) and (4)).

At this point, φ contains almost all the atoms needed to eventually obtain $\mathcal{E}$- and $\mathcal{G}$-instantiation closed sets of literals; the only missing closure is w.r.t. condition (5) of Definition 4.2.4. This will be done by the function g_5-$inst$ in the loop, as it will be clear in a moment.

The **while** loop is iterated until there exists a propositional assignment β^p which satisfies the propositional abstraction φ^p of φ. The propositional assignment β^p is refined, thus obtaining a constraint which is (deterministically) closed under condition (5) of Definition 4.2.4 by the function g_5-$inst$, and then passed to the decision procedures for Presburger Arithmetic $\mathsf{DP}_{\mathcal{P}}$ and for the constraint satisfiability problem for the theory of arrays $\mathsf{DP}_{\mathcal{A}}$. If both procedures return (*"satisfiable"*, $\emptyset$), then the algorithm stops returning satisfiability; otherwise, as it is customary in lazy SMT solvers (see, e.g., Bozzano et al., 2006), the corresponding conflict clause is used to prune the search space in order to avoid enumerating useless guesses, i.e. all those sharing the same conflict set.

The correctness of the procedure can be obtained along the lines of the Delayed Theory Combination algorithm in Bozzano et al. (2006). The main differences lie in showing that the pre-processing steps preserve the $\mathcal{ADP}$-equisatisfiability and that

$L^{\mathcal{G}}$ is an $\mathcal{E}$- and $\mathcal{G}$-instantiation closed set of literals so that Lemma 4.2.7 above can be re-used.

Finally, we notice that all the extensions considered in Subsection 4.3.3 can be easily integrated in Algorithm 4 by adapting the g_*'s functions in order to mirror the extensions in the definition of $\mathcal{G}$-instantiation closed sets.

4.5 Conclusions and Related Work

In this chapter we have considered extensions of the theory of arrays which are relevant for many important applications such as program verification. These extensions are such that the indexes of arrays have the algebraic structure of Presburger Arithmetic and the theory of arrays is augmented with axioms characterizing additional symbols such as dimension, injectivity, or the domain of definition of arrays. We have obtained the decidability of the constraint satisfiability problem for all the considered extensions by a combination of decision procedures for the theories of arrays and Presburger Arithmetic with various instantiation strategies based both on model-theoretic and rewriting-based methods. We have also described techniques for the efficient implementation of the non-deterministic decision procedures by adapting techniques developed in the SMT community.

The work most closely related to the topic of this chapter is Bradley et al. (2006), where a syntactic characterization of a class of full first-order formulae is considered, which turns out to be expressive enough to specify many properties of interest about arrays. The main difference is that we have a semantic approach to extending $\mathcal{A}$ by considering a well-chosen class of first-order structures. This allows us to get a more refined characterization of some properties of arrays, yielding, for example, the decidability of the constraint satisfiability problem for the extension of $\mathcal{A}$ with the injectivity axiom (cf. Subsection 4.3.1). The decidability of a similar problem was firstly left open in Bradley et al. (2006) and finally proved undecidable in Bradley (2007): this is so because the class of models (associated to a set of axioms) is larger than the one considered in this work. Moreover, in Bradley (2007), a decidability result for a guarded fragment of *Partial Arrays* (i.e., arrays in which elements may be undefined) is given; this fragment is expressive enough to encode some of the properties covered by our combination framework (such as, for example, sortedness).

Our instantiation strategy based on Superposition Calculus (cf. Subsection 4.4.1)

has a similar spirit of the work Ganzinger and Korovin (2004), where equational reasoning is integrated in instantiation-based theorem proving. The main difference with Ganzinger and Korovin (2004) is that we solve the state-explosion problem, due to the recombination of formulae caused by the use of standard superposition rules (see, e.g., Nieuwenhuis and Rubio, 2001), by deriving a new termination result for an extension of $\mathcal{A}$ as recommended by the rewriting approach to satisfiability procedures of Armando et al. (2003). This allows us to re-use efficient state-of-the-art theorem provers without the need to implement a new inference system as required by Ganzinger and Korovin (2004).

Conclusions and Future Works

In this thesis we have investigated how combination methods for the constraint satisfiability problem for non-disjoint theories can be successfully applied to give decidability results for: (i) (fragments of) theories of interest for the field of software verification (such as extensions of the theory of arrays with dimension), (ii) the satisfiability problem for "temporalized" fragments of first-order theories, and (iii) the model checking problem for infinite-state systems.

First of all, as far as the satisfiability problem for "temporalized" fragments of first-order theories is concerned, we have dealt with the satisfiability problem for data-flow theories, i.e. for fragments of first-order theories endowed with a temporal dimension (in a sense similar to that introduced in Finger and Gabbay, 1992). Even if we focused on the quantifier-free fragment of theories whose constraint satisfiability problem is decidable, we have shown the undecidability of the (ground) satisfiability problem for (totally flexible) data-flow theories. This limitative result is obtained through a reduction to the constraint satisfiability problem for unions of (signature disjoint) theories in a first-order framework. The analysis of the causes that lead to undecidability suggested a strong relationship between the ground satisfiability problem for data-flow theories and the constraint satisfiability problem for first-order theories over non-disjoint signature; this relationship allowed us to derive sufficient conditions in order to guarantee decidability results.

Secondly, we have enriched the framework for dealing with the satisfiability problem for data-flow theories in such a way that it becomes possible to encode transition relations. The framework so obtained allows a declarative approach to the model checking problem for (possibly) infinite-state systems. By a straightforward reduction to the reachability problem for Minksy machines, we have shown that the Noetherianity hypothesis, sufficient to guarantee the decidability of the ground satisfiability problem for data-flow theories, is not anymore sufficient to avoid undecidability of the ground model checking problem. However, under the stronger

local finiteness hypothesis of the theory over the time-independent signature, we have been able to decide the ground model checking problem first when considering safety properties, and then the entire class of temporal properties.

The last contribution of this thesis concerns decidability results for the constraint satisfiability problem for interesting extensions of the theory of arrays with dimension. To this aim, we have developed a uniform framework based on instantiation strategies and on combination of decision procedures for Linear Arithmetic over indexes and (the universal fragment of) arrays; this two ingredients are used to eliminate the extensions in favor of the available decision procedures. We have also addressed some of the problems arising in the implementation of the procedures presented, both by using the rewriting-approach to build satisfiability procedures and relying on the use of SMT solvers.

As far as future works are concerned, on the one hand we plan to generalize the conditions guaranteeing the decidability of the satisfiability problem for data-flow theories following the argument used to show that the validity problem for LTL($\Sigma^{\underline{a}}$)-sentences is recursive enumerable; this should allow us to enlarge the scope of our techniques and to include in our framework, among others, some of the results of Sofronie-Stokkermans (2006). On the other hand, there are two main lines of future work concerning the model checking problem. First, we intend to investigate how to exploit SMT solvers to solve model checking problems; the design of suitable heuristics to efficiently explore the safety and LTL($\Sigma^{\underline{a},\underline{c}}$)-graphs should be the key to show the viability of our approach. Second, we intend to find termination results for model checking modulo richer background theories (e.g., Presburger Arithmetic). We believe that this can be achieved by considering transition relations satisfying certain requirements as it is done in, e.g., Demri et al. (2006). Finally, as far as the extensions of the theory of arrays with dimension are regarded, we plan to implement the SMT-based algorithm that we have developed and perform some experimental evaluations. In particular, this should allow us to compare the relative benefits of the two variants of the decision procedure for the theory of arrays with dimensions on some significant problems.

Appendix

Superposition Calculus: an Overview

From now on, we consider only universal, finitely axiomatized theories, whose signatures are finite. Without loss of generality, we assume that signatures contain only function symbols, because any atom $P(t_1, \ldots, t_n)$ with predicate symbol P other than equality can be written as an equation $p(t_1, \ldots, t_n) = true$, where p is a fresh function symbol and $true$ a fresh constant symbol, and this transformation preserves satisfiability (see, e.g., Nieuwenhuis and Rubio, 2001).

In the following, $=$ denotes equality, $\equiv$ denotes identity, l, r, u, t are terms, v, w, x, y, z are variables, all other lower case letters are constant or function symbols. A fundamental feature of the Superposition Calculus (from now on, $\mathcal{SP}$) is the usage of a *term reduction ordering (TRO)* $\prec$ (see, e.g., Baader and Nipkow, 1998) which is total on ground terms. The ordering $\prec$ is extended to literals in such a way that only maximal sides of maximal instances of literals are considered when applying the expansion rules of Figure 2. The most commonly used orderings are the *Knuth-Bendix ordering (KBO)* and the *lexicographic path ordering (LPO)*.

A clause C is *redundant* with respect to a set S of clauses if either $C \in S$ or S can be obtained from $S \cup \{C\}$ by a sequence of application of the contraction rules of Figure 3. An inference is *redundant* with respect to a set S of clauses if its conclusion is redundant with respect to S. A set S of clauses is *saturated* with respect to $\mathcal{SP}$ if every inference of $\mathcal{SP}$ with a premise in S is redundant with respect to S. A *derivation* is a sequence $S_0, S_1, \ldots, S_i, \ldots$ of sets of clauses where at each step an inference of $\mathcal{SP}$ is applied to generate and add a clause (see expansion rules in Figure 2) or to delete or reduce a clause (see contraction rules in Figure 3). A derivation is characterized by its *limit*, defined as the set of persistent clauses $S_\infty = \bigcup_{j \geq 0} \bigcap_{i > j} S_i$. A derivation $S_0, S_1, \ldots, S_i, \ldots$ with limit S_∞ is *fair* with respect to $\mathcal{SP}$ if for every inference in $\mathcal{SP}$ with premises in S_∞, there is some $j \geq 0$ such that the inference is redundant in S_j.

Theorem (Nieuwenhuis and Rubio, 2001)**.** *If $S_0, S_1, \ldots$ is a fair derivation of $\mathcal{SP}$, then* (i) *its limit S_∞ is saturated with respect to $\mathcal{SP}$,* (ii) *S_0 is unsatisfiable iff the empty clause is in S_j for some j, and* (iii) *if such a fair derivation is finite, i.e. it is of the form $S_0, \ldots, S_n$, then S_n is saturated and logically equivalent to S_0.*

Superposition	$\dfrac{\Gamma \Rightarrow \Delta, l[u'] = r \quad \Pi \Rightarrow \Sigma, u = t}{(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)\sigma}$	$(i), (ii), (iii), (iv)$
Paramodulation	$\dfrac{\Gamma, l[u'] = r \Rightarrow \Delta \quad \Pi \Rightarrow \Sigma, u = t}{(l[t] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma)\sigma}$	$(i), (ii), (iii), (iv)$
Reflection	$\dfrac{\Gamma, u' = u \Rightarrow \Delta}{(\Gamma \Rightarrow \Delta)\sigma}$	(v)
Eq. Factoring	$\dfrac{\Gamma \Rightarrow \Delta, u = t, u' = t'}{(\Gamma, t = t' \Rightarrow \Delta, u = t')\sigma}$	$(i), (vi)$

Legenda: a clause $\neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_n$ is written in sequent style as $\{A_1, \ldots, A_n\} \Rightarrow \{B_1, \ldots, B_m\}$ (where the A_i's and B_j's are literals), equality is the only predicate symbol, σ is the most general unifier of u and u', u' is not a variable in *Superposition* and *Paramodulation*, L is a literal, and the following hold:

(i) $u\sigma \not\preceq t\sigma$, *(ii)* $\forall L \in \Pi \cup \Sigma : (u = t)\sigma \not\preceq L\sigma$, *(iii)* $l[u']\sigma \not\preceq r\sigma$, *(iv)* $\forall L \in \Gamma \cup \Delta : (l[u'] = r)\sigma \not\preceq L\sigma$, *(v)* for all $L \in \Gamma \cup \Delta : (u' = u)\sigma \not\prec L\sigma$, and *(vi)* for all $L \in \Gamma : u\sigma \not\preceq L\sigma$, and for all $L \in \{u' = t'\} \cup \Delta : (u = t)\sigma \not\prec L\sigma$.

Figure 2: Expansion Inference Rules of $\mathcal{SP}$.

Subsumption	$\dfrac{S \cup \{C, C'\}}{S \cup \{C\}}$	if $C\vartheta \subseteq C'$ for some substitution ϑ
Simplification	$\dfrac{S \cup \{C[l'], l = r\}}{S \cup \{C[r\vartheta], l = r\}}$	if $l' \equiv l\vartheta$, $r\vartheta \prec l\vartheta$, and $\forall L \in C[l\vartheta] : (l\vartheta = r\vartheta) \prec L$
Deletion	$\dfrac{S \cup \{\Gamma \Rightarrow \Delta, t = t\}}{S}$	

where C and C' are clauses and S is a set of clauses.

Figure 3: Contraction Inference Rules of $\mathcal{SP}$.

We say that $\mathcal{SP}$ is *refutationally complete* since it is possible to derive the empty clause with a finite derivation from an unsatisfiable set of clauses (see *(ii)* of theorem above). The proof of this theorem (see Nieuwenhuis and Rubio, 2001, but also Bachmair and Ganzinger, 1990, 1994) relies on the creation of a convergent rewriting system from the set of all the ground instances of a saturated set of clauses. If the empty clause does not belong to the saturation, a model for S_0 can be built from the set of all the ground terms identified by the equivalence relation deriving from the rewriting rules.

The Water Level Controller Example

This section presents more details about Example 3.4.1. We recall that we are considering a data-flow theory $\mathcal{T} = \langle \Sigma, T, \Sigma_r, \underline{a}, \underline{c} \rangle$ such that

- $\Sigma = \{in, out, l_{\text{alarm}}, l_{\text{overflow}}, <\}$ where in, out are two unary function symbols, $l_{\text{alarm}}, l_{\text{overflow}}$ are two constant symbols, $<$ is a binary predicate symbol;
- $\Sigma_r = \{l_{\text{alarm}}, l_{\text{overflow}}, <\}$;
- $T = T_r \cup \left\{ \begin{array}{l} \forall x \, (x < l_{\text{alarm}} \rightarrow in(x) < l_{\text{overflow}}), \\ \forall x \, (x < l_{\text{overflow}} \rightarrow out(x) < l_{\text{alarm}}) \end{array} \right\}$
  where T_r is the theory of dense linear orders without endpoints endowed with the further axiom $l_{\text{alarm}} < l_{\text{overflow}}$.
- l is the only system variable and there are no system parameters (that is, $\underline{a} := \{l\}$ and $\underline{c} := \emptyset$).

The LTL-system specification is $(\mathcal{T}, \delta, \iota)$ where δ is

$$\begin{array}{rcl} \delta(l^0, l^1) & :\equiv & \big(l_{\text{alarm}} \leq l^0 \rightarrow l^1 = in^0(out^0(l^0))\big) \quad \wedge \\ & \wedge & \big(l^0 < l_{\text{alarm}} \rightarrow l^1 = in^0(l^0)\big) \end{array}$$

and the initial state condition is

$$\iota(l) \quad :\equiv \quad l < l_{\text{alarm}}.$$

Finally, the unsafe states are represented by the formula $\nu(l) :\equiv l_{\text{overflow}} \leq l$.

Figures 4 and 5 present the set of T-consistent $\hat{\delta}$ assignment and transition Σ_r guessings. Since the transition relation δ is a purely left formula, from now on we consider only T-consistent nodes (i.e, nodes (V, G) such that $V \wedge G$ is T-consistent); indeed, recalling Definition 3.3.3 of safety graph, T-inconsistent nodes (i) cannot be initial nodes and (ii) cannot be reached by any path in the safety graph (it is easy to see that such nodes cannot have any incoming edge).

(a)	$l^0 < l_{\text{alarm}}$	$l^0 \neq l_{\text{alarm}}$	$l_{\text{alarm}} \not< l^0$	$l^1 = in^0(out^0(l^0))$	$l^1 = in^0(l^0)$
(b)	$l^0 < l_{\text{alarm}}$	$l^0 \neq l_{\text{alarm}}$	$l_{\text{alarm}} \not< l^0$	$l^1 \neq in^0(out^0(l^0))$	$l^1 = in^0(l^0)$
(c)	$l^0 \not< l_{\text{alarm}}$	$l^0 = l_{\text{alarm}}$	$l_{\text{alarm}} \not< l^0$	$l^1 = in^0(out^0(l^0))$	$l^1 = in^0(l^0)$
(d)	$l^0 \not< l_{\text{alarm}}$	$l^0 = l_{\text{alarm}}$	$l_{\text{alarm}} \not< l^0$	$l^1 = in^0(out^0(l^0))$	$l^1 \neq in^0(l^0)$
(e)	$l^0 \not< l_{\text{alarm}}$	$l^0 \neq l_{\text{alarm}}$	$l_{\text{alarm}} < l^0$	$l^1 = in^0(out^0(l^0))$	$l^1 = in^0(l^0)$
(f)	$l^0 \not< l_{\text{alarm}}$	$l^0 \neq l_{\text{alarm}}$	$l_{\text{alarm}} < l^0$	$l^1 = in^0(out^0(l^0))$	$l^1 \neq in^0(l^0)$

Figure 4: T-consistent $\hat{\delta}$ assignments

(1)	$l^1 < l^0 < l_{\text{alarm}} < l_{\text{overflow}}$	(13)	$l^1 < l_{\text{alarm}} < l^0 < l_{\text{overflow}}$
(2)	$l^1 = l^0 < l_{\text{alarm}} < l_{\text{overflow}}$	(14)	$l^1 = l_{\text{alarm}} < l^0 < l_{\text{overflow}}$
(3)	$l^0 < l^1 < l_{\text{alarm}} < l_{\text{overflow}}$	(15)	$l_{\text{alarm}} < l^1 < l^0 < l_{\text{overflow}}$
(4)	$l^0 < l^1 = l_{\text{alarm}} < l_{\text{overflow}}$	(16)	$l_{\text{alarm}} < l^0 = l^1 < l_{\text{overflow}}$
(5)	$l^0 < l_{\text{alarm}} < l^1 < l_{\text{overflow}}$	(17)	$l_{\text{alarm}} < l^0 < l^1 < l_{\text{overflow}}$
(6)	$l^0 < l_{\text{alarm}} < l^1 = l_{\text{overflow}}$	(18)	$l_{\text{alarm}} < l^0 < l^1 = l_{\text{overflow}}$
(7)	$l^0 < l_{\text{alarm}} < l_{\text{overflow}} < l^1$	(19)	$l_{\text{alarm}} < l^0 < l_{\text{overflow}} < l^1$
(8)	$l^1 < l^0 = l_{\text{alarm}} < l_{\text{overflow}}$	(20)	$l^1 < l_{\text{alarm}} < l^0 = l_{\text{overflow}}$
(9)	$l^0 = l^1 = l_{\text{alarm}} < l_{\text{overflow}}$	(21)	$l^1 = l_{\text{alarm}} < l^0 = l_{\text{overflow}}$
(10)	$l^0 = l_{\text{alarm}} < l^1 < l_{\text{overflow}}$	(22)	$l_{\text{alarm}} < l^1 < l^0 = l_{\text{overflow}}$
(11)	$l^0 = l_{\text{alarm}} < l^1 = l_{\text{overflow}}$	(23)	$l_{\text{alarm}} < l^0 = l^1 = l_{\text{overflow}}$
(12)	$l^0 = l_{\text{alarm}} < l_{\text{overflow}} < l^1$	(24)	$l_{\text{alarm}} < l^0 = l_{\text{overflow}} < l^1$

(25)	$l^1 < l_{\text{alarm}} < l_{\text{overflow}} < l^0$
(26)	$l^1 = l_{\text{alarm}} < l_{\text{overflow}} < l^0$
(27)	$l_{\text{alarm}} < l^1 < l_{\text{overflow}} < l^0$
(28)	$l_{\text{alarm}} < l^1 = l_{\text{overflow}} < l^0$
(29)	$l_{\text{alarm}} < l_{\text{overflow}} < l^1 < l^0$
(30)	$l_{\text{alarm}} < l_{\text{overflow}} < l^0 = l^1$
(31)	$l_{\text{alarm}} < l_{\text{overflow}} < l^0 < l^1$

Figure 5: T-consistent transition Σ_r guessings

The first column of Figure 6 presents the initial nodes, i.e. the nodes satisfying the initial condition $\iota(l^0) :\equiv l^0 < l_{\text{alarm}}$, whereas the second column presents the final nodes, i.e. the nodes satisfying $\nu(l^1) :\equiv l_{\text{overflow}} \leq l^1$.

		(F)	$(V_{(\text{e})}, G_{(23)})$
(I)	$(V_{(\text{a})}, G_{(1)})$	(F)	$(V_{(\text{e})}, G_{(23)})$
(I)	$(V_{(\text{b})}, G_{(1)})$	(F)	$(V_{(\text{e})}, G_{(24)})$
(I)	$(V_{(\text{a})}, G_{(2)})$	(F)	$(V_{(\text{f})}, G_{(24)})$
(I)	$(V_{(\text{b})}, G_{(2)})$	(F)	$(V_{(\text{e})}, G_{(28)})$
(I)	$(V_{(\text{a})}, G_{(3)})$	(F)	$(V_{(\text{f})}, G_{(28)})$
(I)	$(V_{(\text{b})}, G_{(3)})$	(F)	$(V_{(\text{e})}, G_{(29)})$
(I)	$(V_{(\text{a})}, G_{(4)})$	(F)	$(V_{(\text{f})}, G_{(29)})$
(I)	$(V_{(\text{b})}, G_{(4)})$	(F)	$(V_{(\text{e})}, G_{(30)})$
(I)	$(V_{(\text{a})}, G_{(5)})$	(F)	$(V_{(\text{f})}, G_{(30)})$
(I)	$(V_{(\text{b})}, G_{(5)})$	(F)	$(V_{(\text{e})}, G_{(31)})$
		(F)	$(V_{(\text{f})}, G_{(31)})$

Figure 6: T-consistent initial (I) and final (F) nodes

Let us now mimic a forward-search strategy for the exploration of the safety graph. Applying Definition 3.3.3 and recalling that the transition relation is a purely left formula, there is an edge between two nodes (V, G) and (W, H) iff

$$G(l^0, l^1) \wedge W(l^1, l^2) \wedge H(l^1, l^2)$$

is T-satisfiable. Table 1 presents the nodes that can be reached in one transition step from an initial node. Each row of the table should be intended as follows: any node on the left-side can reach in one transition step any node on the right-side. We discover that, besides the initial nodes, also the nodes marked with an asterisk sign (*) can be reached.

$(V_{(a)}, G_{(1)})$, $(V_{(b)}, G_{(1)})$, $(V_{(a)}, G_{(2)})$, $(V_{(b)}, G_{(2)})$, $(V_{(a)}, G_{(3)})$, $(V_{(b)}, G_{(3)})$, $(V_{(a)}, G_{(5)})$, $(V_{(b)}, G_{(5)})$	$\Longrightarrow$	$(V_{(a)}, G_{(1)})$, $(V_{(b)}, G_{(1)})$, $(V_{(a)}, G_{(2)})$, $(V_{(b)}, G_{(2)})$, $(V_{(a)}, G_{(3)})$, $(V_{(b)}, G_{(3)})$, $(V_{(a)}, G_{(4)})$, $(V_{(b)}, G_{(4)})$, $(V_{(a)}, G_{(5)})$, $(V_{(b)}, G_{(5)})$
$(V_{(a)}, G_{(4)})$, $(V_{(b)}, G_{(4)})$	$\Longrightarrow$	$(V_{(c)}, G_{(8)})^\star$, $(V_{(d)}, G_{(8)})^\star$, $(V_{(c)}, G_{(9)})^\star$, $(V_{(d)}, G_{(9)})^\star$, $(V_{(c)}, G_{(10)})^\star$, $(V_{(d)}, G_{(10)})^\star$

Table 1: T-consistent nodes reachable in one step from the initial nodes

We iterate the same procedure over the nodes discovered in the previous step to collect the nodes that are reachable in two transition steps. Again, newly discovered nodes are marked with an asterisk sign (*).

$(V_{(c)}, G_{(8)})$, $(V_{(d)}, G_{(8)})$	$\Longrightarrow$	$(V_{(a)}, G_{(1)})$, $(V_{(b)}, G_{(1)})$, $(V_{(a)}, G_{(2)})$, $(V_{(b)}, G_{(2)})$, $(V_{(a)}, G_{(3)})$, $(V_{(b)}, G_{(3)})$, $(V_{(a)}, G_{(4)})$, $(V_{(b)}, G_{(4)})$, $(V_{(a)}, G_{(5)})$, $(V_{(b)}, G_{(5)})$
$(V_{(c)}, G_{(9)})$, $(V_{(d)}, G_{(9)})$	$\Longrightarrow$	$(V_{(c)}, G_{(8)})$, $(V_{(d)}, G_{(8)})$, $(V_{(c)}, G_{(9)})$, $(V_{(d)}, G_{(9)})$, $(V_{(c)}, G_{(10)})$, $(V_{(d)}, G_{(10)})$
$(V_{(c)}, G_{(10)})$, $(V_{(d)}, G_{(10)})$	$\Longrightarrow$	$(V_{(e)}, G_{(13)})^\star$, $(V_{(f)}, G_{(13)})^\star$, $(V_{(e)}, G_{(14)})^\star$, $(V_{(f)}, G_{(14)})^\star$, $(V_{(e)}, G_{(15)})^\star$, $(V_{(f)}, G_{(15)})^\star$, $(V_{(e)}, G_{(16)})^\star$, $(V_{(f)}, G_{(16)})^\star$, $(V_{(e)}, G_{(17)})^\star$, $(V_{(f)}, G_{(17)})^\star$

Table 2: Further T-consistent nodes reachable in two steps from the initial nodes

The third iteration of the process does not reach new nodes, hence the set of nodes collected till now is exactly the set of nodes that are reachable from an initial node. Since this set does not contain any final node, we can conclude that the system is safe.

$(V_{(c)}, G_{(13)})$, $(V_{(d)}, G_{(13)})$	$\Longrightarrow$	$(V_{(a)}, G_{(1)})$, $(V_{(b)}, G_{(1)})$, $(V_{(a)}, G_{(2)})$, $(V_{(b)}, G_{(2)})$, $(V_{(a)}, G_{(3)})$, $(V_{(b)}, G_{(3)})$, $(V_{(a)}, G_{(4)})$, $(V_{(b)}, G_{(4)})$, $(V_{(a)}, G_{(5)})$, $(V_{(b)}, G_{(5)})$
$(V_{(c)}, G_{(14)})$, $(V_{(d)}, G_{(14)})$	$\Longrightarrow$	$(V_{(c)}, G_{(8)})$, $(V_{(d)}, G_{(8)})$, $(V_{(c)}, G_{(9)})$, $(V_{(d)}, G_{(9)})$, $(V_{(c)}, G_{(10)})$, $(V_{(d)}, G_{(10)})$
$(V_{(c)}, G_{(15)})$, $(V_{(d)}, G_{(15)})$, $(V_{(c)}, G_{(16)})$, $(V_{(d)}, G_{(16)})$, $(V_{(c)}, G_{(17)})$, $(V_{(d)}, G_{(17)})$	$\Longrightarrow$	$(V_{(e)}, G_{(13)})$, $(V_{(f)}, G_{(13)})$, $(V_{(e)}, G_{(14)})$, $(V_{(f)}, G_{(14)})$, $(V_{(e)}, G_{(15)})$, $(V_{(f)}, G_{(15)})$, $(V_{(e)}, G_{(16)})$, $(V_{(f)}, G_{(16)})$, $(V_{(e)}, G_{(17)})$, $(V_{(f)}, G_{(17)})$

Table 3: Further T-consistent nodes reachable in three or more steps from the initial nodes

Finally, Table 4 mimic the result of a backward-search strategy, i.e. collects all the nodes that can reach a final node of the safety graph. Since from the result presented in table it turns out that only final nodes can reach final nodes, the exploration stops immediately. The fact that no final node is initial allows to conclude again the safety of the system.

$(V_{(e)}, G_{(23)})$, $(V_{(f)}, G_{(23)})$, $(V_{(e)}, G_{(28)})$, $(V_{(f)}, G_{(28)})$	$\Longrightarrow$	$(V_{(e)}, G_{(23)})$, $(V_{(f)}, G_{(23)})$
$(V_{(e)}, G_{(23)})$, $(V_{(f)}, G_{(23)})$, $(V_{(e)}, G_{(28)})$, $(V_{(f)}, G_{(28)})$	$\Longrightarrow$	$(V_{(e)}, G_{(24)})$, $(V_{(f)}, G_{(24)})$
$(V_{(e)}, G_{(24)})$, $(V_{(f)}, G_{(24)})$, $(V_{(e)}, G_{(31)})$, $(V_{(f)}, G_{(31)})$	$\Longrightarrow$	$(V_{(e)}, G_{(28)})$, $(V_{(f)}, G_{(28)})$
$(V_{(e)}, G_{(24)})$, $(V_{(f)}, G_{(24)})$, $(V_{(e)}, G_{(31)})$, $(V_{(f)}, G_{(31)})$	$\Longrightarrow$	$(V_{(e)}, G_{(29)})$, $(V_{(f)}, G_{(29)})$
$(V_{(e)}, G_{(24)})$, $(V_{(f)}, G_{(24)})$, $(V_{(e)}, G_{(31)})$, $(V_{(f)}, G_{(31)})$	$\Longrightarrow$	$(V_{(e)}, G_{(30)})$, $(V_{(f)}, G_{(30)})$
$(V_{(e)}, G_{(24)})$, $(V_{(f)}, G_{(24)})$, $(V_{(e)}, G_{(31)})$, $(V_{(f)}, G_{(31)})$	$\Longrightarrow$	$(V_{(e)}, G_{(31)})$, $(V_{(f)}, G_{(31)})$

Table 4: T-consistent nodes that can reach final nodes

Index

Bibliography

Parosh A. Abdulla and Bengt Jonsson. Verifying networks of timed processes. In Bernhard Steffen, editor, *Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1998)*, volume 1384 of *Lecture Notes in Computer Science*, pages 298–312, Lisbon (Portugal), 1998. Springer-Verlag.

Parosh A. Abdulla, Ahmed Bouajjani, Bengt Jonsson, and Marcus Nilsson. Handling global conditions in parameterized system verification. In N. Halbwachs and D. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification (CAV 1999)*, volume 1633 of *Lecture Notes in Computer Science*, pages 134–145, Trento (Italy), 1999. Springer-Verlag.

Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings of the 11th IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 313–321, New Brunswick (NJ, USA), 1996. IEEE Computer Society.

Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.

Alessandro Armando, Maria P. Bonacina, Silvio Ranise, and Stephan Schulz. New results on rewrite-based satisfiability procedures. *ACM Transactions on Computational Logic*, 2007. (To appear).

Franz Baader and Silvio Ghilardi. Connecting many-sorted theories. *Journal of Symbolic Logic*, 2006. (To appear).

Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, United Kingdom, 1998.

Franz Baader and Cesare Tinelli. Deciding the word problem in the union of equational theories. *Information and Computation*, 178(2):346–390, 2002.

Franz Baader, Carsten Lutz, Holger Sturm, and Frank Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research*, 16:1–58, 2002.

Franz Baader, Silvio Ghilardi, and Cesare Tinelli. A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. *Information and Computation*, 204(10):1413–1452, 2006.

Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In M. E. Stickel, editor, *Proceedings of 10th International Conference on Automated Deduction (CADE 1990)*, volume 449 of *Lecture Notes in Computer Science*, pages 427–441, Kaiserslautern (Germany), 1990. Springer-Verlag.

Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

Thomas Ball and Sriram K. Rajamani. The slam toolkit. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102 of *Lecture Notes in Computer Science*, pages 260–264, Paris (France), 2001. Springer-Verlag.

Clark W. Barrett, David L. Dill, and Aaron Stump. A generalization of Shostak's method for combining decision procedures. In A. Armando, editor, *Proceedings of the 4th International Workshop on Frontiers of Combining Systems (FroCoS 2002)*, volume 2309 of *Lecture Notes in Computer Science*, pages 132–147, Santa Margherita Ligure (Italy), 2002. Springer-Verlag.

Peter Baumgartner, Ulrich Furbach, and Uwe Petermann. A unified approach to theory reasoning. Research Report 15–92, Universität Koblenz-Landau, Koblenz (Germany), 1992. Fachberichte Informatik.

Saddek Bensalem, Ahmed Bouajjani, Claire Loiseaux, and Joseph Sifakis. Property preserving simulations. In G. von Bochmann and D. K. Probst, editors, *Proceedings of the 4th International Workshop on Computer Aided Verification (CAV 1992)*, volume 663 of *Lecture Notes in Computer Science*, pages 260–273, Montreal (Canada), 1992. Springer-Verlag.

Orna Bernholtz, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Proceedings of 6th International Conference on Computer Aided Verification (CAV 1994)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, Stanford (CA, USA), 1994. Springer-Verlag.

Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Masahiro Fujita, and Yunshan Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings of the 36th Conference on Design Automation (DAC 1999)*, pages 317–320, New Orleans (LA, USA), 1999. ACM Press.

Nikolaj Bjørner, Mark E. Stickel, and Tomás E. Uribe. A practical integration of first-order reasoning and decision procedures. In W. McCune, editor, *Proceedings of the 14th International Conference on Automated Deduction (CADE 1997)*, volume 1249 of *Lecture Notes in Computer Science*, pages 101–115, Townsville (Australia), 1997. Springer-Verlag.

Nikolaj S. Bjørner. *Integrating Decision Procedures for Temporal Verification*. PhD thesis, Department of Computer Science, Stanford University, Stanford (CA, USA), 1998.

Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2002.

BLAST. BLAST: Berkeley Lazy Abstraction Software Verification Tool. http://mtc.epfl.ch/software-tools/blast/.

Maria P. Bonacina, Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Decidability and undecidability results for Nelson-Oppen and rewrite-based decision procedures. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Computer Science*, pages 513–527, Seattle (WA, USA), 2006. Springer-Verlag.

Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In A. W. Mazurkiewicz and J. Winkowski, editors, *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR 1997)*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150, Warsaw (Poland), 1997. Springer-Verlag.

Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular model checking. In A. E. Emerson and A. P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*, volume 1855 of *Lecture Notes in Computer Science*, pages 403–418, Chicago (IL, USA), 2000. Springer-Verlag.

Ahmed Bouajjani, Peter Habermehl, and Tomás Vojnar. Abstract regular model checking. In R. Alur and D. A. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004)*, volume 3114 of *Lecture Notes in Computer Science*, pages 372–386, Boston (MA, USA), 2004. Springer-Verlag.

Ahmed Bouajjani, Peter Habermehl, Pierre Moro, and Tomás Vojnar. Verifying programs with dynamic 1-selector-linked structures in regular model checking. In N. Halbwachs and L. D. Zuck, editors, *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*

(TACAS 2005), volume 3440 of *Lecture Notes in Computer Science*, pages 13–29, Edinburgh (UK), 2005. Springer-Verlag.

Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomás Vojnar. Programs with lists are counter automata. In T. Ball and R. B. Jones, editors, *Proceedings of the 18th International Conference on Computer Aided Verification (CAV 2006)*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531, Seattle (WA, USA), 2006. Springer-Verlag.

Robert S. Boyer and J. Strother Moore. *A Computational Logic Handbook*. Academic Press, second edition, 1997.

Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi A. Junttila, Silvio Ranise, Peter van Rossum, and Roberto Sebastiani. Efficient theory combination via boolean search. *Journal of Information and Computation*, 204(10): 1493–1525, 2006.

Aaron R. Bradley. *Safety Analysis of Systems*. PhD thesis, Department of Computer Science, Stanford University, Stanford (CA, USA), 2007.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. Termination of polynomial programs. In R. Cousot, editor, *Proceedings of the 6th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2005)*, volume 3385 of *Lecture Notes in Computer Science*, pages 113–129, Paris (France), 2005. Springer-Verlag.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. What's decidable about arrays? In A. E. Emerson and K. S. Namjoshi, editors, *Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2006)*, volume 3855 of *Lecture Notes in Computer Science*, pages 427–442, Charleston (SC, USA), 2006. Springer-Verlag.

Torben Bräuner and Silvio Ghilardi. First-order modal logic. In J. van Benthem, P. Blackburn, and F. Wolter, editors, *Handbook of Modal Logic*, pages 549–620. Elsevier, Amsterdam, 2007.

Andrej Brodnik, Svante Carlsson, Erik D. Demaine, J. Ian Munro, and Robert Sedgewick. Resizable arrays in optimal time and space. In F. K. H. A. Dehne, A. Gupta, J.-R. Sack, and R. Tamassia, editors, *Proceedings of the 6th International Workshop on Algorithms and Data Structures (WADS 1999)*, volume 1663 of *Lecture Notes in Computer Science*, pages 37–48, Vancouver (Canada), 1999. Springer-Verlag.

Stephen D. Brookes. A semantics for concurrent separation logic. In P. Gardner and N. Yoshida, editors, *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR 2004)*, volume 3170 of *Lecture Notes in Computer Science*, pages 16–34, London (UK), 2004. Springer-Verlag.

Randal E. Bryant, Shuvendu K. Lahiri, and Sanjit A. Seshia. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 78–92, Copenhagen (Denmark), 2002. Springer-Verlag.

Richard J. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the International Congress on Logic, Math, and Philosophy of Science*. Standford University Press, 1962.

Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2), 1992.

Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification of infinite state structures. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebras*, pages 545–623. Elsevier, Amsterdam, 2001.

Alexander Chagrov and Michael Zakharyaschev. *Modal Logic*. Clarendon Press, Oxford, 1997.

Cheng-Chung Chang and Jerome H. Keisler. *Model Theory*. North-Holland, Amsterdam-London, third edition, 1990.

Philippe Le Chenadec. *Canonical Forms in Finitely Presented Algebras*. Research Notes in Theoretical Computer Science. Pitman-Wiley, 1986.

Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364, Copenhagen (Denmark), 2002. Springer-Verlag.

Edmund M. Clarke, Allen E. Emerson, and Prasad A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Language and Systems*, 8(2):244–263, 1986.

Edmund M. Clarke, Thomas Filkorn, and Somesh Jha. Exploiting symmetry in temporal logic model checking. In C. Courcoubetis, editor, *Proceedings of the 5th International Conference on Computer Aided Verification (CAV 1993)*, volume 697 of *Lecture Notes in Computer Science*, pages 450–462, Elounda (Greece), 1993. Springer-Verlag.

Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5): 1512–1542, 1994.

Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In A. E. Emerson and A. P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000) July 15-19, 2000,*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169, Chicago (IL, USA), 2000. Springer-Verlag.

Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Abstraction refinement for termination. In C. Hankin and I. Siveroni, editors, *Proceedings of the 12th International Symposium on Static Analysis (SAS 2005)*, volume 3672 of *Lecture Notes in Computer Science*, pages 87–101, London (UK), 2005. Springer-Verlag.

Coq. The Coq proof assistant. Reference manual, 2006. Available at http://coq.inria.fr/.

Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages (POPL 1977)*, pages 238–252, Los Angeles (CA, USA), 1977. ACM Press.

David Cyrluk and Paliath Narendran. Ground temporal logic: A logic for hardware verification. In D. L. Dill, editor, *Proceedings of the 6th International Conference on Computer Aided Verification (CAV 1994)*, volume 818 of *Lecture Notes in Computer Science*, pages 247–259, Stanford (CA, USA), 1994. Springer-Verlag.

Leonardo M. de Moura, Harald Rueß, and Maria Sorea. Lazy theorem proving for bounded model checking over infinite domains. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction (CADE 2002)*, volume 2392 of *Lecture Notes in Computer Science*, pages 438–455, Copenhagen (Denmark), 2002. Springer-Verlag.

Anatoli Degtyarev, Michael Fisher, and Boris Konev. Monodic temporal resolution. *ACM Transaction on Computational Logic*, 7(1):108–150, 2006.

Giorgio Delzanno, Javier Esparza, and Andreas Podelski. Constraint-based analysis of broadcast protocols. In J. Flum and M. Rodríguez-Artalejo, editors, *Proceedings of the 13th International Workshop on Computer Science Logic (CSL 1999)*, volume 1683 of *Lecture Notes in Computer Science*, pages 50–66, Madrid (Spain), 1999. Springer-Verlag.

Stéphane Demri. Linear-time temporal logics with Presburger constraints: An overview. *Journal of Applied Non-Classical Logics*, 16(3–4):311–347, 2006.

Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Towards a model-checker for counter systems. In S. Graf and W. Zhang, editors, *Proceedings of the 4th International Symposium on Automated Technology for Verification and Analysis (ATVA 2006)*, volume 4218 of *Lecture Notes in Computer Science*, pages 493–507, Beijing (ROC), 2006. Springer-Verlag.

Peter J. Downey and Ravi Sethi. Assignment commands with array references. *Journal of the ACM*, 25(4):652–666, 1978.

Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical logic*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1994.

Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York-London, 1972.

Javier Esparza and Stefan Schwoon. A BDD-based model checker for recursive programs. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102 of *Lecture Notes in Computer Science*, pages 324–336, Paris (France), 2001. Springer-Verlag.

Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS 1999)*, pages 352–359, Trento (Italy), 1999. IEEE Computer Society.

Marcelo Finger and Dov M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language, and Information*, 1(3):203–233, 1992.

Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, and Raymie Stata. Extended static checking for Java. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2002)*, volume 37 of *ACM SIGPLAN Notices*, pages 234–245, Berlin (Germany), 2002. ACM Press.

Dov M. Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam-London, 2003.

Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Harper & Row, 1986.

Harald Ganzinger. Shostak light. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction (CADE 2002)*, volume 2392 of *Lecture Notes in Computer Science*, pages 332–346, Copenhagen (Denmark), 2002. Springer-Verlag.

Harald Ganzinger and Konstantin Korovin. Integrating equational reasoning in instantiation-based theorem proving. In J. Marcinkowski and A. Tarlecki, editors, *Proceedings of the 18th International Workshop on Computer Science Logic (CSL 2004)*, volume 3210 of *Lecture Notes in Computer Science*, pages 71–84, Karpacz (Poland), 2004. Springer-Verlag.

Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

Silvio Ghilardi. Reasoners' cooperation and quantifiers elimination. Technical Report 288-03, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Milano (Italy), 2003. Available at http://homes.dsi.unimi.it/~ghilardi.

Silvio Ghilardi. Model theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4):221–249, 2004.

Silvio Ghilardi and Luigi Santocanale. Algebraic and model theoretic techniques for fusion decidability in modal logics. In M. Vardi and A. Voronkov, editors, *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2003)*, volume 2850 of *Lecture Notes in Computer Science*, pages 152–166, Almaty (Kazakhstan), 2003. Springer-Verlag.

Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Deciding extensions of the theory of arrays by integrating decision procedures and instantiation strategies. In M. Fischer, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Proceedings of the 10th European Conference on Logic in Artificial Intelligence (JELIA 2006)*, volume 4160 of *Lecture Notes in Computer Science*, pages 177–189, Liverpool (UK), 2006a. Springer-Verlag.

Silvio Ghilardi, Enrica Nicolini, and Daniele Zucchelli. A comprehensive framework for combined decision procedures. *ACM Transactions on Computational Logic*, 2006b. (To appear). Technical Report available at http://homes.dsi.unimi.it/~zucchell/publications/techreport/GhiNiZu-RI304-05.pdf.

Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Decision procedures for extensions of the theory of arrays. *Annals of Mathematics and Artificial Intelligence*, 50(3-4):231–254, 2007a.

Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Combination methods for satisfiability and model-checking of infinite-state systems. In F. Pfenning, editor, *Proceedings of the 21st Conference on Automated Deduction (CADE*

2007), volume 4603 of *Lecture Notes in Computer Science*, pages 362–378, Bremen (Germany), 2007b. Springer-Verlag.

Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Noetherianity and combination problems. In B. Konev and F. Wolter, editors, *Proceedings of the 6th International Workshop on Frontiers of Combining Systems (FroCoS 2007)*, volume 4720 of *Lecture Notes in Computer Science*, pages 206–220, Liverpool (UK), 2007c. Springer-Verlag.

Susanne Graf and Hassen Saïdi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification (CAV 1997)*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83, Haifa (Israel), 1997. Springer-Verlag.

Reiko Heckel. *Open Graph Transformation Systems: A New Approach to the Compositional Modelling of Concurrent and Reactive Systems.* PhD thesis, Fachberichte Informatik, Technischen Universität Berlin, Berlin (Germany), 1998.

Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Lazy abstraction. In *Proceedings of the 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2002)*, pages 58–70, Portland (OR, USA), 2002. ACM Press.

Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Software verification with blast. In T. Ball and S. K. Rajamani, editors, *Proceedings of the 10th International SPIN Workshop on Model Checking Software*, volume 2648 of *Lecture Notes in Computer Science*, pages 235–239, Portland (OR, USA), 2003. Springer-Verlag.

Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Kenneth L. McMillan. Abstractions from proofs. In N. D. Jones and X. Leroy, editors, *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2004)*, pages 232–244, Venice (Italy), 2004. ACM Press.

Wilfrid Hodges. *Model Theory.* Number 42 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993.

Ian M. Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragment of first-order temporal logics. *Annals of Pure and Applied Logic*, 106(1–3):85–134, 2000.

Norris C. Ip and David L. Dill. Better verification through symmetry. In D. Agnew, L. J. M. Claesen, and R. Camposano, editors, *Proceedings of the 11th IFIP WG10.2 International Conference on Computer Hardware Description Languages and their Applications (CHDL 1993)*, volume A-32 of *IFIP Transactions*, pages 97–111, Ottawa (Canada), 1993. North-Holland.

Daniel Jackson and Mandana Vaziri. Finding bugs with a constraint solver. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 2000)*, ACM SIGSOFT Software Engineering Notes, pages 14–25, Portland (OR, USA), 2000. ACM Press.

Joxan Jaffar. Presburger arithmetic with array segments. *Information Processing Letters*, 12(2):79–82, 1981.

Shmuel Katz and Doron A. Peled. An efficient verification method for parallel and distributed programs. In J. W. de Bakker, Willem P. de Roever, and G. Rozenberg, editors, *Proceedings of the Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (REX Workshop)*, volume 354 of *Lecture Notes in Computer Science*, pages 489–507, Noordwijkerhout (The Netherlands), 1988. Springer-Verlag.

Hélène Kirchner, Silvio Ranise, Christophe Ringeissen, and Duc-Khanh Tran. On superposition-based satisfiability procedures and their combination. In D. Van Hung and M. Wirsing, editors, *Proceedings of the 2nd International Colloquium on Theoretical Aspects of Computing (ICTAC 2005)*, volume 3722 of *Lecture Notes in Computer Science*, pages 594–608, Hanoi (Vietnam), 2005. Springer-Verlag.

Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

Jean-Louis Lassez and Michael J. Maher. On Fourier's algorithm for linear arithmetic constraints. *Journal of Automated Reasoning*, 9(3):373–379, 1992.

Jean-Louis Lassez and Ken McAloon. A canonical form for generalized linear constraints. *Journal of Symbolic Computation*, 13(1):1–24, 1992.

Claire Loiseaux, Susanne Graf, Joseph Sifakis, Ahmed Bouajjani, and Saddek Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1), 1995.

David E. Long. *Model checking, Abstraction and Compositional Verification*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh (PA, USA), 1993.

Saunders MacLane and Garrett Birkhoff. *Algebra*. Chelsea Publishing Co., New York (USA), third edition, 1988.

Monika Maidl. A unifying model checking approach for safety properties of parameterized systems. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102 of *Lecture Notes in Computer Science*, pages 311–323, Paris (France), 2001. Springer-Verlag.

Michael Makkai and Gonzalo E. Reyes. *First Order Categorical Logic: Model-Theoretical Methods in the Theory of Topoi and Related Categories*. Number 611 in Lecture Notes in Mathematics. Springer-Verlag, 1977.

Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.

Zohar Manna, Anuchit Anuchitanukul, Nicolaj Bjørner, Anca Browne, Edward Chang, Michael Colón, Luca de Alfaro, Harish Devarajan, Henny Sipma, and Tomás Uribe. STeP: The Stanford Temporal Prover. Technical Report STAN-CS-TR-94-1518, Department of Computer Science, Stanford University, Stanford (CA, USA), 1994. Available at http://theory.stanford.edu/~zm/papers/step.ps.Z.

Prabhaker Mateti. A decision procedure for the correctness of a class of programs. *Journal of the ACM*, 28(2):215–232, 1981.

Richard Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*. PhD thesis, Institut für Informatik, Technischen Universität München, Munich (Germany), 1998.

John McCarthy. Towards a mathematical theory of computation. In *Proceedings of IFIP Congress*, pages 21–28, 1962.

Kenneth L. McMillan. Applications of Craig interpolants in model checking. In N. Halbwachs and L. D. Zuck, editors, *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005)*, volume 3440 of *Lecture Notes in Computer Science*, pages 1–12, Edinburgh (UK), 2005. Springer-Verlag.

Kenneth L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In G. von Bochmann and D. K. Probst, editors, *Proceedings of the 4th International Workshop on Computer Aided Verification (CAV 1992)*, volume 663 of *Lecture Notes in Computer Science*, pages 164–177, Montreal (Canada), 1992a. Springer-Verlag.

Kenneth L. McMillan. *Symbolic model checking: An approach to the state explosion problem*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh (PA, USA), 1992b.

Scott McPeak and George C. Necula. Data structures specification via local equality axioms. In K. Etessami and S. K. Rajamani, editors, *Proceedings of the 17th International Conference on Computer Aided Verification (CAV 2005)*, volume 3576 of *Lecture Notes in Computer Science*, pages 476–490, Edinburgh (UK), 2005. Springer-Verlag.

Stephan Merz. Model checking: A tutorial overview. In F. Cassez, C. Jard, B. Rozoy, and M. Dermot Ryan, editors, *4th Summer School on Modeling and Verification of Parallel Processes (MOVEP 2000)*, volume 2067 of *Lecture Notes in Computer Science*, pages 3–38. Springer-Verlag, Nantes (France), 2001.

Stephan Merz. On the logic of TLA. *Computing and Informatics*, 22:351–379, 2003.

Stephan Merz. Isabelle/TLA, 1999. Available at http://isabelle.in.tum.de/library/HOL/TLA.

Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in the theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

NIST – National Institute of Standards and Technology. The economic impacts of inadequate infrastructure for software testing. Planning Report 02-3, U.S. Department of Commerce, 2002. Available at http://www.nist.gov/director/prog-ofc/report02-3.pdf.

Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Transaction on Programming Languages and Systems*, 1(2):245–257, 1979.

Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

Enrica Nicolini. *Combined decision procedures for constraint satisfiability*. PhD thesis, Dipartimento di Matematica, Università degli Studi di Milano, Milano (Italy), 2007.

R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier Science, 2001.

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

Peter W. O'Hearn. Resources, concurrency, and local reasoning. *Theoretical Computer Science*, 375(1-3):271–307, 2007.

Derek C. Oppen. Complexity, convexity and combinations of theories. *Theoretical Computer Science*, 12:291–302, 1980.

William T. Overman. *Verification of concurrent systems: function and timing*. PhD thesis, University of California at Los Angeles, 1981.

Sam Owre, John M. Rushby, and Natarajan Shankar. PVS: A Prototype Verification System. In *Proceedings of the 11th International Conference on Automated Deduction (CADE 1992)*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752, Saratoga Springs (NY, USA), 1992. Springer-Verlag.

David A. Plaisted. A decision procedure for combination of propositional temporal logic and other specialized theories. *Journal of Automated Reasoning*, 2(2):171–190, 1986.

Amir Pnueli, Sitvanit Ruath, and Lenore D. Zuck. Automatic deductive verification with invisible invariants. In T. Margaria and W. Yi, editors, *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, volume 2031 of *Lecture Notes in Computer Science*, pages 82–97, Genova (Italy), 2001. Springer-Verlag.

Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

Silvio Ranise and Cesare Tinelli. Satisfiability modulo theories. *IEEE Magazine on Intelligent Systems*, 21(6):71–81, 2006.

John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS 2002)*, pages 55–74, Copenhagen (Denmark), 2002. IEEE Computer Society.

John C. Reynolds. Reasoning about arrays. *Communications of the ACM*, 22(5): 290–299, 1979.

Michaël Rusinowitch. *Démonstration Automatique (Techniques de réécriture)*. InterEditions, 1989.

Hassen Saïdi and Natarajan Shankar. Abstract and model check while you prove. In N. Halbwachs and D. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification (CAV 1999)*, volume 1633 of *Lecture Notes in Computer Science*, pages 443–454, Trento (Italy), 1999. Springer-Verlag.

Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

Henny B. Sipma, Tomás E. Uribe, and Zohar Manna. Deductive model checking. *Formal Methods in System Design*, 15(1):49–74, 1999.

A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

SLAM. The Slam Project. http://research.microsoft.com/slam/.

Viorica Sofronie-Stokkermans. Interpolation in local theory extensions. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Computer Science*, pages 235–250, Seattle (WA, USA), 2006. Springer-Verlag.

Aaron Stump, Clark W. Barrett, David L. Dill, and Jeremy R. Levitt. A decision procedure for an extensional theory of arrays. In *Proceedings of the 16th IEEE Symposium on Logic in Computer Science (LICS 2001)*, pages 29–37, Boston (MA, USA), 2001. IEEE Computer Society.

Norihisa Suzuki and David R. Jefferson. Verification decidability of Presburger array programs. *Journal of the ACM*, 27(1):191–205, 1980.

Cesare Tinelli. Cooperation of background reasoners in theory reasoning by residue sharing. *Journal of Automated Reasoning*, 3(1):1–31, 2003.

Cesare Tinelli and Mehdi T. Harandi. A new correctness proof of the Nelson-Oppen combination procedure. In F. Baader and K.U. Schulz, editors, *Proceedings of the 1st International Workshop on Frontiers of Combining Systems (FroCoS 1996)*, Applied Logic, pages 103–120, Munich (Germany), 1996. Kluwer Academic Publishers.

Cesare Tinelli and Calogero G. Zarba. Combining non-stably infinite theories. *Journal of Automated Reasoning*, 34(3):209–238, 2005.

Ashish Tiwari. Termination of linear programs. In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004)*, volume 3114 of *Lecture Notes in Computer Science*, pages 70–82, Boston (MA, USA), 2004. Springer-Verlag.

Dirk van Dalen. *Logic and Structure*. Springer-Verlag, 1989. Second edition.

Moshe Y. Vardi. Verification of concurrent programs: the automata-theoretic framework. *Annals of Pure and Applied Logic*, 51(1–2):79–98, 1991.

Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of Symposium on Logic in Computer Science (LICS 1986)*, pages 332–344, Cambridge (MA, USA), 1986. IEEE Computer Society.

William H. Wheeler. Model-companions and definability in existentially complete structures. *Israel Journal of Mathematics*, 25:305–330, 1976.

Frank Wolter. Fusions of modal logics revisited. In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic*. CSLI, Stanford (CA, USA), 1998.