Contribution à la génération de séquences pour la conduite de systèmes complexes critiques

Thomas Cochard

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Contribution à la génération de séquences pour la conduite de systèmes complexes critiques

THÈSE

présentée et soutenue publiquement le 13 Décembre 2017

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention génie informatique, automatique et traitement du signal)

par

Thomas COCHARD

Composition du jury

Président :	Benoit IUNG	Professeur à l'Université de Lorraine
Rapporteurs :	Pascal BERRUET	Professeur à l'Université de Bretagne-Sud
	Bernard RIERA	Professeur à l'Université de Reims Champagne-Ardenne
Examinateurs :	Khalid KOUISS	Maître de conférences à SIGMA Clermont
	Catherine DEVIC	EDF R&D
	David GOUYON	Maître de Conférences à l'Université de Lorraine (Co-encadrant de thèse)
Directeur de thèse :	Jean-François PETIN	Professeur à l'Université de Lorraine

Centre de Recherche en Automatique de Nancy - UMR 7039

Mis en page avec la classe thesul.

Remerciements

Les dernières lignes d'écriture, et un sentiment d'achèvement.

Dans les quelques lignes qui suivent, je tiens à remercier des personnes qui, de près ou de loin, ont contribuées à mon développement personnel et / ou professionnel, au cours de la thèse, mais également avant et après celle-ci.

Dans un premier temps, je souhaite remercier l'ensemble du jury de ma soutenance de thèse : Benoit Iung, Catherine Devic, Khalid Kouassi. J'adresse un remerciement tout particulier à Bernard Riera et à Pascal Berruet, pour avoir pris le temps de lire et d'évaluer, en amont de la soutenance, le manuscrit.

Je souhaite remercier également l'ensemble des doctorants et enseignants-chercheurs que j'ai pu cotoyé au cours de ces années. Certains sont rattachés au laboratoire et / ou à la Faculté des Sciences de Nancy, merci à eux pour les bons moments passés qui ont permis de se détendre dans des périodes parfois compliquées, d'autres au département R&T de l'IUT de Nancy-Brabois, ceux-ci m'ont vu grandir pendant ces 9 dernières années, et d'autres encore au département GEII de l'IUT de Nancy-Brabois, pour la confiance qu'ils m'ont accordé et qu'ils renouvellent encore en 2017/2018.

J'adresse pour tout cela un grand merci : à Fabien, Alexandre, Jérémy, Damien, Laëtitia, Dorine, Houssem, Salah, Teju, et tous ceux avec qui j'ai pu partager un café à la BSR ; à Eric, Pascale, Vincent, Patrick, Benoit, André, Jean-Yves, Alexis, Alexandre, William, Jean-Philippe, Mario, Philippe pour leur aide pendant ces trois années et demies passées au laboratoire ; à Thierry, Francis, Brice, Michael, Hugues, David, Vincent, Nicolas, Bénédicte, Hélène, Béné, et tous les autres enseignants de R&T pour la formation de qualité qu'ils dispensent ; à Taha, Caroline, Cédric, Jean-Marie, Dominique, Frédéric, Christophe S., Christophe A., Franck, Patrick, Christian, Olivier, et tous les autres enseignants de GEII pour leur confiance et leur bonne humeur quotidienne.

J'adresserais un remerciement tout particulier encore à trois autres personnes : à Didier, pour m'avoir fait confiance pendant les divers conseils de laboratoire, pour sa prise en compte des remarques de doctorants, et sa confiance pour la visite du comité du HCERES ; à Dominique, pour la même confiance qu'il m'a accordé, et pour sa grande implication auprès des doctorants de l'École Doctorale IAEM ; à Taha, pour m'avoir considéré non pas comme un doctorant mais tout simplement comme un collègue, et pour m'avoir donné l'opportunité d'enseigner et de m'investir dans la vie du département, notamment à la préparation du colloque anuuel des départements GEII.

J'adresse également mes remerciements à Eric, qui m'a accepté dans son équipe chez Adista après mes trois années de thèse. Je pense aussi à tous les collègues que j'ai pu cotoyé depuis que j'ai été embauché, et qui ont facilité mon insertion dans le mon de professionnel : Nicolas bien sûr, mais également les SJ4 Thomas, Adil, Marco, Quentin, Axel, Jérôme, Thomas, Stan, Manu, Guillaume, Yoann, Ludo, Pierre, Jonas, Sebastien, Brice, Bastien, les SJ3 Dylan, Chris, Alex, Nico, Manu, Philip, Guillaume, Fred, Marie, Ivy, Hélène, et tous les autres, d'agences

distantes mais toujours disponibles et toujours dans la bonne humeur : Guillaume, Pierrick, Yohan, Florian, Jérémie, Julien, David, Elisa, Stéphanie, Patrick, Laurent, Cyrille, Souhail, etc ...

Enfin, je tiens à remercier ma famille, et plus particulièrement mes parents, qui ont toujours été là pour supporter mes joies et mes peines. Je remercie également mon parrain pour son soutien infaillible, et tout le reste de ma famille.

Je remercie également mes amis, passés, présents et à venir, pour les expériences vécues ensemble, qui auront fait de moi la personne que je suis aujourd'hui. Un merci tout particulier à Léa et Geoffrey pour leur présence constante et le soutien mutuel que l'on sait s'apporter sans mots.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Le projet CONNEXION (COntrôle-commande Nucléaire Numérique pour l'EXport et la rénovatION) a réuni, depuis 2012, les principaux acteurs académiques et industriels français du domaine du contrôle/commande nucléaire [1]. L'objectif de ce projet, financé dans le cadre de l'appel à projet Briques Génériques du Logiciel Embarqué 2 (BGLE2) des Investissements d'Avenir, est de proposer une démarche de conception et d'exploitation d'architectures innovantes et sûres de contrôle-commande [Devic and Morilhat, 2013, Devic, 2014, Devic and Dourgnon, 2015]. En conception, l'enjeu affiché est d'être en mesure de disposer d'une chaîne intégrée d'outils logiciels facilitant la définition, le développement et la validation de solutions modulaires de contrôle-commande face à la diversité des exigences normatives et de sécurité à l'export. En exploitation, il s'agit de tirer parti des avancées dans le domaine du numérique (IoT, technologies sans fil, simulateurs, ...) pour proposer des assistances à l'exploitation (conduite, maintenance, gestion technique) tout en conservant la maîtrise des données tout au long du cycle de vie des installations.

Parmi les axes de recherche développés dans le projet CONNEXION, celui relatif à la conduite de systèmes critiques concerne :

- la phase d'ingénierie avec pour objectif d'intégrer le point de vue de l'exploitant au plus tôt dans la validation des architectures de contrôle de commande,
- la phase d'exploitation avec pour objectif de fournir une aide à la préparation et à l'exécution des procédures de conduite.

En effet, d'un point de vue ingénierie, la diversité des contraintes normatives selon le pays cible conduit à une variabilité importante des architectures de contrôle-commande. Dans ce contexte, même si les outils logiciels développés, dans le projet CONNEXION, en termes de traçabilité et de vérification des exigences apportent une aide efficace pour les activités de validation des architectures, la prise en compte des contraintes d'exploitation dans ces activités a été jugée comme un complément significatif par les partenaires du projet. D'autre part, en termes d'exploitation, la complexité croissante des architectures et la multiplicité de leurs configurations rendent délicat le choix d'une stratégie de conduite pertinente et sûre. L'objectif est donc de fournir une aide à la préparation des actions prévisionnelles de conduite en proposant un ensemble de choix réalisables et compatibles avec les exigences de sécurité.

1. Académiques : Commissariat à l'Énergie Atomique, Centre de Recherche en Automatique de Nancy, Institut National de Recherche en Informatique et en Automatique, Laboratoire d'Informatique de Grenoble, Laboratoire Universitaire de Recherche en Production Automatisée, Télécom ParisTech
Industriels : Électricité de France, Areva, Alstom, Atos Worlgrid, Rolls-Royce Civil Nuclear, Corys TESS, Esterel Technologies, All4Tec, Predict

Dans ce contexte, la contribution présentée dans ce mémoire porte sur la génération et la vérification de séquences d'actions de conduite répondant à un objectif donné et pouvant être opérées en toute sécurité sur le procédé. Selon la phase concernée, les séquences générées et vérifiées pourront servir :

- en ingénierie, à vérifier l'existence d'une séquence permettant d'amener les installations d'un point de fonctionnement donné à un point de fonctionnement cible en tenant compte de l'éventuelle indisponibilité d'un ensemble d'équipements ou de fonctions de contrôle-commande,
- en exploitation, à définir ou à sélectionner parmi un ensemble de solutions admissibles, une séquence prévisionnelle d'actions de conduite satisfaisant l'ensemble des contraintes de sécurité et les objectifs fonctionnels de conduite.

S'il existe des approches centrées sur l'ingénierie de la conduite, et plus particulièrement sur la modélisation des procédures [Lind, 2011a] [Lind et al., 2011] [Arzen and Johnsson, 1996] [Viswanathan et al., 1998a], peu d'approches permettant la génération automatique de séquences d'actions sont proposées dans la littérature scientifique du domaine. En première approximation, le problème posé peut être considéré comme un problème de recherche de chemin dans un graphe représentant les situations stabilisées du procédé à conduire. En pratique, le problème s'avère plus complexe dans la mesure où la dynamique des installations doit être prise en compte pour autoriser ou interdire certaines évolutions :

- en fonctions de l'évolution des grandeurs physiques, suite à certaines actions de conduite,
- des contraintes de précédence éventuelles entre fonctions ou équipements mis en œuvre dans l'architecture de contrôle-commande,
- des contraintes de sécurité.

En généralisant, le problème se ramène donc à une recherche de chemins dans un espace d'états caractérisant les évolutions d'un ensemble de modèles dynamiques. Dans la mesure où nous faisons l'hypothèse qu'une représentation discrète et simplifiée de l'évolution des grandeurs physiques doit être suffisante pour notre problème et que les contraintes de précédence ou de sécurité peuvent s'exprimer de manière logique, l'approche retenue sera basée sur l'exploration de l'espace d'état de modèles de Systèmes à Événements Discrets. Cette technique a déjà montré son efficacité dans le domaine de la conduite des procédés chimiques [Yeh and Chang, 2012] mais présente des limites auxquelles s'attaque notre contribution :

- d'une part, l'approche proposée doit s'insérer dans un contexte d'ingénierie industriel existant ; elle doit notamment s'efforcer à réduire la consommation de ressources (en hommes et en temps) requise pour la modélisation de multiples solutions d'architectures de contrôle-commande, en privilégiant la modularité et la réutilisation de modèles,
- d'autre part, l'approche proposée doit permettre un passage à l'échelle pour des installations industrielles, ce qui s'avère souvent délicat pour les approches basées sur la synthèse ou l'exploration de l'espace d'état.

Le premier chapitre de ce manuscrit a pour objectif de présenter le contexte industriel des travaux et leur positionnement scientifique. Pour cela, après avoir introduit la conduite de systèmes complexes critiques et les besoins industriels en conduite, il donne une formalisation du problème industriel adressé. Pour répondre à ce problème, le chapitre fait ensuite un état

de l'art des approches permettant la génération de séquences d'actions de conduite, en se focalisant tout d'abord sur la modélisation des procédures de conduite, sur leur génération, puis sur les formalismes de modélisation du système à conduire permettant cette génération. La problématique scientifique des travaux conclut ce chapitre en justifiant le choix proposé pour y répondre à l'aide d'une recherche d'atteignabilité sur un réseaux d'automates à états temporisés.

Le second chapitre est consacré à la définition d'un cadre formel de modélisation des architectures pour la génération de séquences d'actions de conduite, en vue de répondre aux exigences de modularité et de réutilisation exprimées précédemment. Après une description des principes généraux de la génération de séquences, le chapitre illustre sur un cas d'étude simple ces principes. Il formalise une première démarche de modélisation modulaire, en définissant des mécanismes de synchronisation entre modèles ainsi que des patrons de modélisation définissant une structure de modèle générique et réutilisable qui pourra être instanciée pour produire les modèles complets des architectures. L'approche de génération de séquences d'actions de conduite, à partir de ces modèles, repose sur un algorithme d'interprétation d'une trace d'exécution depuis un état cible jusqu'à un état désiré. Sa pertinence est discutée à la fin de ce chapitre, montrant son intérêt, mais aussi ses limites sur des modèles de grande taille.

Le troisième chapitre aborde le problème du passage à l'échelle à l'aide de techniques d'abstraction de modèles et d'un processus de recherche d'atteignabilité itératif exploitant la hiérarchisation intrinsèque des architectures. La génération de séquences d'actions de conduite peut alors être réalisée par raffinements successifs sur des paires de niveaux de modèles, du plus haut niveau d'abstraction jusqu'au plus bas, permettant ainsi de réduire la taille de l'espace d'état exploré. A partir des traces obtenues pour chacune des itérations, des algorithmes sont proposés pour reconstruire une trace puis une séquence d'action globale, une séquence d'actions optimisée ou plus généralement un ensemble de séquences globales classées selon un ou plusieurs critères (longueur, durée, robustesse, ...).

La mise en œuvre de l'approche de génération de séquences est présentée au chapitre quatre. Celui-ci débute par la présentation d'un prototype d'outil pour la modélisation et la génération de séquences reposant sur le logiciel UppAal, des *templates* et des exécutables spécifiques. Le premier cas d'étude est basé sur la plateforme CISPI du CRAN. Son objectif est d'éprouver l'applicabilité de la méthodologie de génération de séquences d'actions de conduite, sur un cas de taille plus importante que le cas simple des chapitres 2 et 3. Le second cas d'étude, issu du projet CONNEXION, a pour objectif d'évaluer le passage à l'échelle industrielle de l'approche. La pertinence de l'approche est ensuite discutée en fin de chapitre, notamment sur les apports en termes de modélisation, de temps de calcul et de taille de l'espace d'état exploré.

Le mémoire se conclue en rappelant les principales contributions de ce travail et leur apport dans le cadre de l'ingénierie de la conduite de systèmes critiques, notamment pour les partenaires du projet CONNEXION. Enfin, des perspectives de recherche sont proposées pour enrichir l'approche présentée dans ce mémoire par la prise en compte de manière plus fine de la dynamique des procédés et d'un contexte incertain, notamment en ce qui concerne la disponibilité ou la performance des équipements.

Chapitre 1

Contexte industriel et positionnement scientifique

Sommaire

Introduction

L'objectif de ce mémoire est la génération de séquences d'actions de conduite pour des systèmes soumis à de fortes contraintes de sûreté de fonctionnement, notamment de sécurité. Ce chapitre débute par une présentation des objectifs et des contraintes de la conduite de systèmes critiques, en particulier des Centrales Nucléaires de Production d'Électricité (CNPE) dans le contexte du projet CONNEXION. Ce chapitre présente ensuite une formalisation du problème de génération de séquences d'actions de conduite tel que formulé par les acteurs industriels du projet CONNEXION et dresse un état de l'art sur les approches pouvant contribuer à la résolution de ce problème. Enfin, le chapitre se conclut par une synthèse argumentée qui justifie le choix d'une approche basée sur la recherche d'atteignabilité sur un réseau d'automates temporisés et présente les principaux verrous scientifiques qui feront l'objet des contributions proposées dans ce mémoire.

1.1 Contexte des travaux

1.1.1 Conduite de systèmes complexes critiques

1.1.1.1 Systèmes complexes critiques

Les systèmes qui font l'objet de cette thèse sont des systèmes industriels qualifiés de complexes et de critiques selon les définitions suivantes.

Pour définir la notion de système, on retiendra la définition donnée par [Meinadier, 2009] :

> *Par définition, tout système est constitué d'un ensemble d'éléments dont la synergie est organisée pour répondre à une finalité dans un environnement donné.*

Cette définition peut être complétée par celle de la complexité fournie plus loin dans la même référence :

> *La complexité des systèmes est souvent caractérisée, au-delà de la complexité intrinsèque des composants et de leur variété, par la complexité du réseau d'interaction, d'où proviennent des comportements émergeants tant intentionnels (les synergies recherchées) que non intentionnels, ces derniers pouvant être néfastes et difficiles à prévoir et maîtriser (résonances, interférences, interblocages, ...)*

La notion de criticité fait référence à des systèmes soumis à de fortes contraintes de sûreté de fonctionnement. Les composantes de la sûreté de fonctionnement sont la Fiabilité, la Maintenabilité, la Disponibilité et la Sécurité. Parmi des quatre composantes, nous considérerons ici principalement la Sécurité (dans le sens de « sécurité-innocuité », en anglais *safety*), telle que définie par [Machin, 2015] :

> *La sécurité-innocuité est l'attribut de la sûreté de fonctionnement défini comme « la non-occurrence de conséquences catastrophiques pour l'environnement » dues au système. Les conséquences catastrophiques comprennent, bien évidemment, le décès et les blessures des utilisateurs et des opérateurs, mais aussi des dommages matériels et financiers, s'ils sont sans commune mesure avec le service fourni par le système.*

> *Beaucoup de systèmes ne sont pas concernés par cette propriété car leurs limites physiques ne leur permettent pas de causer de tels dommages. Les autres systèmes sont appelés systèmes critiques.*

Ces systèmes sont critiques lorsque les comportements imprévus mènent à des situations catastrophiques [Machin, 2015] telles que celles liées à l'accident de Three Miles Island [Walker and Stine, 2004].

Dans le cadre du projet CONNEXION, les systèmes complexes critiques étudiés sont les CNPE. Leur principe de fonctionnement est de produire de la chaleur par une réaction nucléaire, contrôlée par introduction (pour diminuer la puissance) ou retrait (pour l'augmenter) d'éléments absorbant les neutrons. La chaleur générée par la réaction nucléaire permet de porter de l'eau à ébullition, générant ainsi de la vapeur. Cette vapeur permet à son tour de faire tourner une turbine, produisant ainsi de l'électricité par le biais d'un alternateur.

Pour assurer ce processus, une centrale nucléaire se compose essentiellement de trois circuits FIGURE 1.1 :

- le circuit primaire dont la mission principale est de produire de l'eau chaude pressurisée (la température de l'eau est aux environs de 320 °C, sa pressurisation permettant de la maintenir à l'état liquide) ; le composant principal de ce circuit est le réacteur ;
- le circuit secondaire dont l'objectif est de produire de l'électricité à l'aide d'une turbine alimentée en vapeur d'eau pressurisée ;
- le circuit de refroidissement dont l'objectif est de transformer la vapeur du circuit secondaire en eau avant sa réinjection dans les générateurs de vapeurs.

Des équipements à l'interface entre ces circuits assurent les échanges thermiques :

- entre les circuits primaires et secondaires : la vaporisation de l'eau est réalisée par les générateurs de vapeurs ;
- entre les circuits secondaire et de refroidissement : le changement de phase vapeur vers eau est réalisé dans le condenseur à l'aide d'une source froide (une rivière par exemple).

1.1.1.2 Architecture d'un système complexe critique

L'architecture d'un système complexe critique décrit un ensemble de services permettant de garantir à la fois que :

1. Le système rempli sa mission, ce pour quoi il a été conçu ;
2. Le système fourni un niveau de sécurité suffisant pour qu'il ne cause aucun dommage, à lui-même ou à son environnement.

D'un point fonctionnel, ces services sont classiquement structurés autour de différents niveaux de fonctionnalités, des fonctions relatives au procédé, au contrôle/commande, à la conduite, la gestion ... Que ces services soient automatisés ou non, il existe également une structuration hiérarchique sur le matériel : on retrouve, du niveau le plus bas au niveau le plus haut respectivement : les équipements (tels que vannes, pompes, ...), les Automates Programmables Industriels (API), et les systèmes de type Supervisory Control And Data Acquisition (SCADA).

Dans un soucis d'unification et de clarification, des représentations mixtes fonctionnelles / organiques existent. En ce qui concerne les procédés de type batch, la méthode orientée objet ASTRID, s'appuyant sur la norme ISA/S88 [ISA, 1998] (publiée comme norme internationale

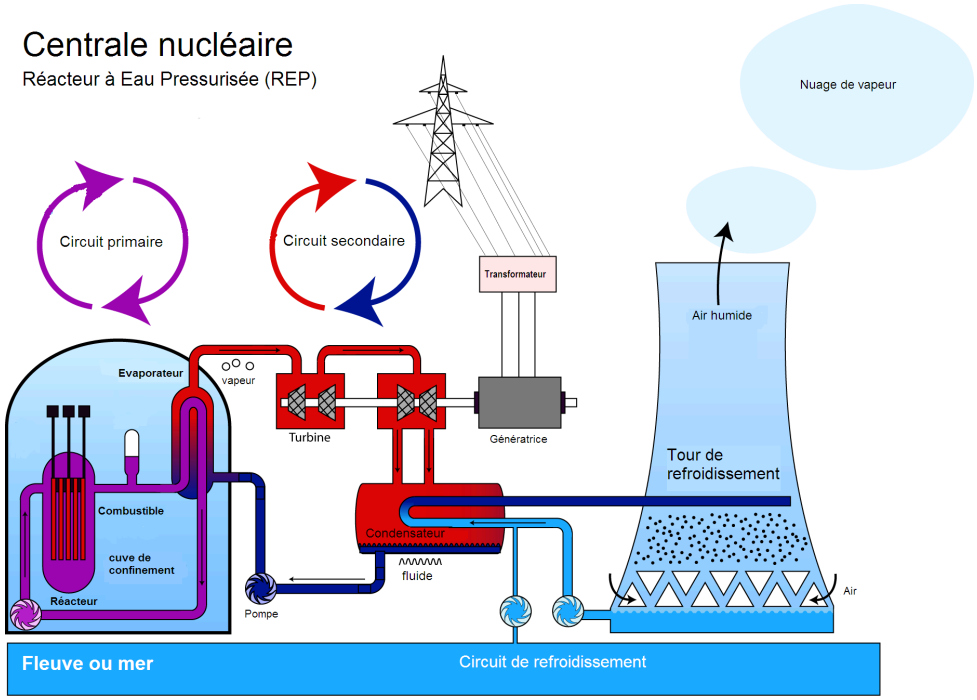


FIGURE 1.1 – Schéma de principe d'une centrale nucléaire de production d'électricité (source : Wikipedia)

IEC/ISO61512 [IEC, 1997]), permet de structurer ces éléments, selon un principe de description hiérarchique de l'installation et des modes opératoires, en sous-ensembles matériels et fonctionnels (FIGURE 1.2) : organes, ressources, fonctions et recettes. Dans le cadre d'ASTRID, les organes (tels que les capteurs et les actionneurs) sont les objets matériels de premier niveau de la hiérarchie, manipulables par des opérateurs ou par le contrôle-commande. Leur rôle est d'observer et d'agir sur le procédé. D'un point de vue comportemental, tandis que les capteurs n'ont que deux états « Normal » ou « Défaut », les actionneurs peuvent être volontairement bloqués par un cadenas. Cette action permet d'empêcher une ressource autre que celle à l'origine du cadennassage de contrôler l'organe, habituellement pour des raisons de sécurité (telles que l'isolation d'un circuit). Les ressources sont des interfaces entre les fonctions et les actionneurs, responsable de la garantie du respect de contraintes de sécurité. La gestion de l'occupation des ressources est nécessaire pour l'ordonnancement des tâches, et sa maîtrise est assurée au niveau recette.

La norme ISA/S95 [ISA, 2010] (publiée comme norme internationale IEC/ISO62264 [IEC, 2013]) généralise, en s'appuyant sur le modèle de référence de l'Université de Purdue [Williams, 1994] cette structuration à tous les types de procédés : continus, batch et discrets (FIGURE 1.3).

Dans le cas particulier des CNPE, à l'image de ce qui est présenté dans [Pétin et al., 1998], on retient classiquement pour la conduite 3 niveaux parmi ceux identifiés précédemment :

– les équipements de terrain : ils assurent la mesure et l'actionnement en interaction avec le procédé (lecture, filtrage et traitement d'un signal, conversion analogique – numérique, gestion de puissance, ...),

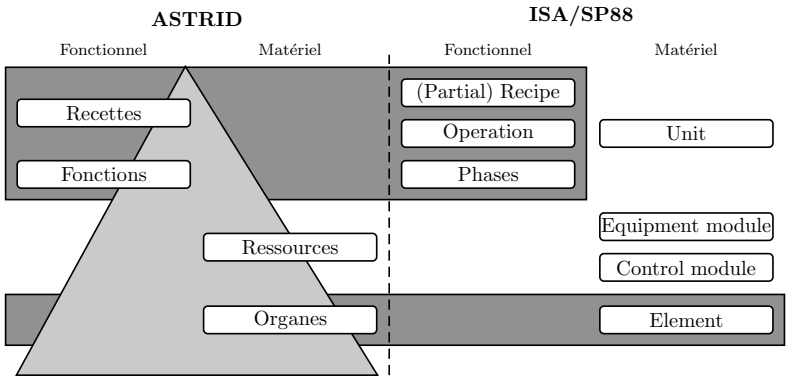


FIGURE 1.2 – Niveaux utilisés dans la norme ISA/S88 [ISA, 1998] et la méthode ASTRID

- les automatismes réflexes : ils assurent la commande du procédé pour maintenir l'état de celui-ci dans une plage de fonctionnement donnée (notamment via l'émission de requêtes à destination des actionneurs),
- la conduite et la surveillance du procédé (conduite locale des équipements manuels et conduite à distance des équipements automatisés, surveillance des grandeurs caractéristiques du procédé, gestion des alarmes, visualisation de l'état du procédé et des chaînes d'actionnement de la transmission des requêtes au système de commande).

Il faut ici noter une caractéristique importante de ces systèmes qui est qu'ils sont majoritairement placés sous le contrôle d'opérateurs humains (FIGURE 1.4). Cela provient du fait qu'un humain est résilient, c'est-à-dire capable de s'adapter aux situations auxquelles il fait face et de réagir en conséquence [Galara and Hennebicq, 1999, Galara, 2006]. Toutefois, cela rajoute de la complexité, du fait de l'imprévisibilité des comportements humains.

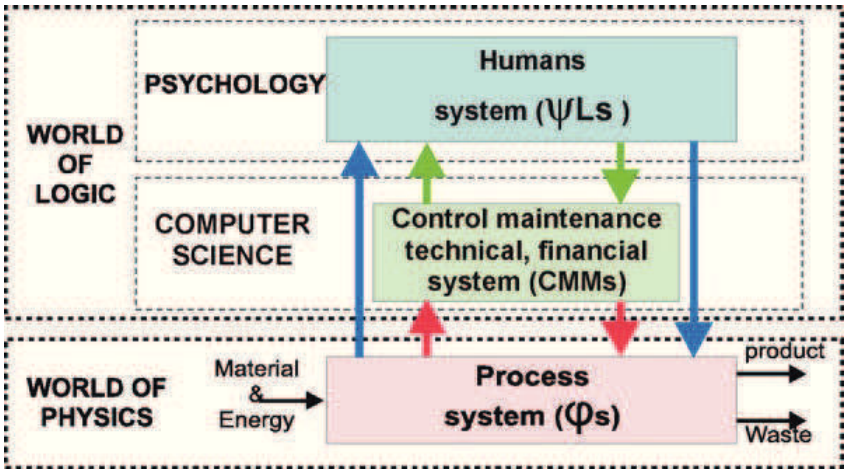


FIGURE 1.4 – Composition du système de conduite d'après [Galara, 2006]

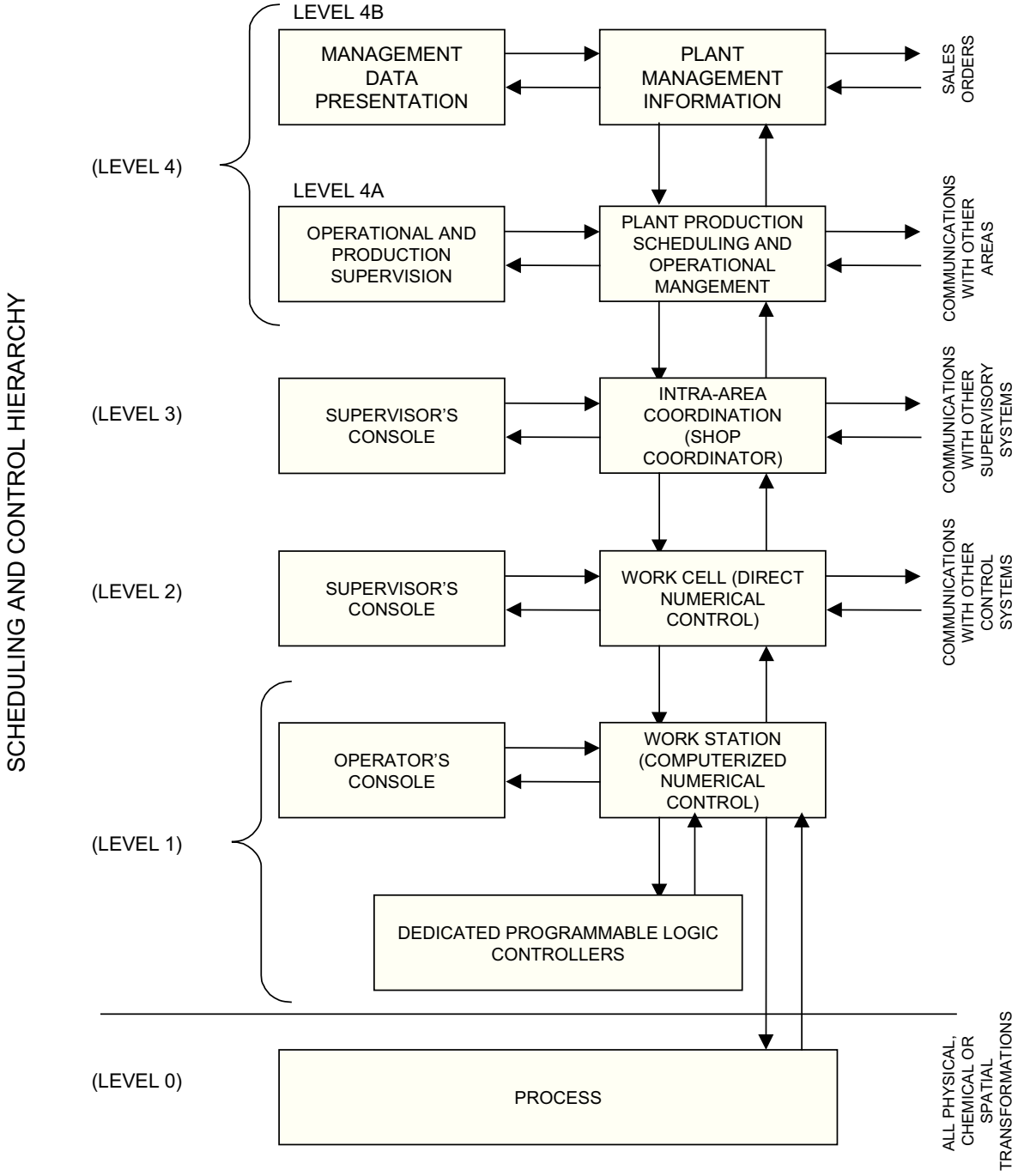


FIGURE 1.3 – Niveaux utilisés dans la norme ISA/S95 [ISA, 2010]

1.1.1.3 Conduite des systèmes complexes critiques

L'objectif de la conduite d'un système industriel est de maîtriser le comportement du procédé, en vue de lui permettre d'accomplir sa mission, en toute sécurité. Pour cela, la conduite nécessite l'observation et la surveillance du procédé, l'analyse des situations observées, et la mise en œuvre d'actions de conduite. Classiquement, des systèmes de type SCADA (Supervisory Control and Data Acquisition) sont utilisés. Ces systèmes permettent de faire :

- l'acquisition des données du procédé,
- le traitement des données,
- l'historisation des données,
- la visualisation de l'état courant du procédé,
- la commande des éléments permettant de faire évoluer cet état.

Les principaux éléments à commander et à contrôler sont les organes de bas niveau, les **équipements**, tels que les vannes et les pompes. Pour les systèmes de grande taille, la gestion de milliers d'équipements repose sur une organisation hiérarchisée telle que présentée à la section précédente. Un ensemble d'équipements peut accomplir, selon une certaine configuration, une **fonction**.

Exemple :
Exemples de fonctions : alimenter en eau par la voie A, filtrer la bâche ASG011BA, ...

L'ensemble des fonctions permet de faire évoluer les **variables physiques** du procédé, permettant ainsi le contrôle du **système** global. Le système, en fonctionnement normal, est alors géré en fonction de situations stabilisées.

Exemple :
Exemples de situations [Alengry, 1989b] : (1) arrêt à froid pour rechargement ; (2) arrêt à froid pour intervention ; (3) arrêt à froid normal ; (4) arrêt intermédiaire monophasique ; (5) arrêt intermédiaire aux conditions du RRA ; (6) arrêt intermédiaire normal (ou biphasique) ; (7) arrêt à chaud ; (8) attente à chaud ; (9) fonctionnement en puissance.

Le rôle de la conduite dans les CNPE

Le pilotage d'un système de production d'électricité consiste à contrôler une réaction complexe dans le but de produire de la chaleur, puis de l'électricité. Maintenir la stabilité de cette réaction est le but des métiers de la conduite, atteignable par le biais des nombreux systèmes de contrôle à disposition.

La planification de la conduite, aussi bien lorsque la tranche est en marche ou lors d'un arrêt, est très complexe, du fait des nombreux matériels à gérer et dont la disponibilité est parfois obligatoire pour des raisons de sécurité [Alengry, 1989a, Alengry, 1989b]. Étant donné la taille du système considéré (à savoir, de 15 000 à 20 000 vannes), une organisation en sous-systèmes hiérarchisés selon trois niveaux (équipements, fonctions, système), est utilisée pour structurer la conduite.

On admet alors, sur la base de connaissances d'experts, qui doivent à la fois connaître l'état réel de l'installation, ainsi que toutes les actions qui seront en cours à l'instant t, que les actions

prévues, non seulement sont réalisables, mais également laissent un degré de liberté suffisant pour assurer les fonctions de sécurité.

C'est entre autre ce manque de connaissance de l'état réel de l'installation qui a mené à l'accident de Three Miles Island [Walker and Stine, 2004]. À l'origine, le problème provient de l'arrêt inopiné d'une pompe. Cet évènement aurait dû être suivi par l'ouverture automatique d'une vanne, qui aurait permis un flux continu d'eau. Cependant, un mauvais fonctionnement, menant à la non ouverture de cette vanne, n'a pas été remarqué par les opérateurs, qui ont poursuivi leurs opérations en se basant sur cette information erronée.

> *Jusqu'à l'accident de Three Mile Island, le 29 mars 1979, aux États-Unis, l'exploitant devait mettre en œuvre une approche dite "évènementielle" pour corriger les anomalies de fonctionnement. Elle consistait, pour chaque type d'incident ou accident, à réagir en menant à bien une suite d'actions prédéterminées. Mais cette approche ne permet pas de gérer les situations complexes où l'évènement à l'origine du problème se complique de défaillances matérielles ou humaines.*

Quelle que soit l'approche utilisée, la conduite a toujours reposé sur des documents papiers prescrivant un ensemble d'opérations et de règles : les procédures.

Conduire en respectant des procédures

Une procédure est une suite d'étapes exprimées sous une forme précise, à savoir un verbe (une action à mener) et un complément d'objet (un organe à contrôler). Elles prescrivent deux types d'actions :

- des opérations d'ouvertures / fermetures de vannes, de consignations d'équipements, à effectuer sur le procédé ;
- des observations de valeurs de variables physiques, à relever sur le procédé et à reporter.

Qualifiées avant leur mise en application, elles sont habituellement sous format papier. Cependant, des procédures informatisés sont introduites progressivement ; il s'agit là de représentations des procédures pour leur utilisation en simulateurs [O'Hara et al., 2000].

1.1.2 Besoins industriels

Un des constats issus du cluster CONNEXION est que l'un des domaines sur lesquels les contraintes sont les plus fortes est celui des métiers de la conduite.

> *La salle de commande, siège de nombreuses interactions Hommes-Systèmes, est le véritable centre nerveux de la centrale. Par les moyens qui y sont disposés, l'équipe de conduite doit être à même de conserver la maîtrise de l'installation et ce, en toute situation. [Devic and Dourgnon, 2015]*

De ce constat particulier, diverses problématiques industrielles peuvent alors être identifiées. On s'intéressera en premier lieu à l'organisation actuelle des métiers de la conduite, afin de mettre en évidence le problème industriel particulier que l'on adresse. On en déduira ensuite la problématique scientifique adressée qui sera développée tout au long de ce manuscrit.

1.1.2.1 Aide à la préparation des actions de conduite

La complexité de la planification et la réalisation des travaux Tranche en Marche (TeM) et en Arrêt de Tranche (AT) est importante du fait de nombreux matériels requis en permanence, que ce soit pour des raisons de sécurité, de production ou d'environnement. Actuellement, les séquences de conduite en fonctionnement normal sont préparées par des opérateurs confirmés appartenant à l'équipe hors quart. Ces plannings de conduite peuvent faire appel à un séquencement de tâches réparties entre des équipements automatisés et manuels. Face à la complexité d'une centrale nucléaire, il semble cependant impossible que des humains soient capables d'envisager tous les scénarios potentiellement possibles dans des délais compatibles avec l'exploitation

> *Maîtriser l'évolution de cette infrastructure socio-technologique qu'est le Contrôle-Commande de centrale nucléaire constitue un défi en soit du fait de la complexité liée [...] au volume de matériel à gérer : 840 moteurs, 300 vannes motorisées électriques, 950 vannes pneumatiques, 100 vannes réglantes, 3 000 capteurs logiques, 2 500 capteurs analogiques ; et du volume de données manipulées : 13 000 alarmes, 5 000 procédures. Une centrale nucléaire, c'est aussi 115 kilomètres de tuyaux et de 15 à 20 000 robinets et vannes. [Devic and Morilhat, 2013]*

Parallèlement, il ne semble pas exister aujourd'hui de méthode pour dénombrer tous ces cas de figure du fait du caractère continu du procédé à considérer, de la diversité fonctionnelle et technologique inhérente aux installations et du volume d'équipements à considérer. Seule une démarche empirique basée sur l'expérience et la connaissance des installations est utilisée. Dans la conduite normale des installations, la complexité combinatoire est masquée par les choix de stratégies de conduite encodées dans les procédures et les consignes, qui résultent de l'expérience et des bonnes pratiques.

Entre la vue informationnelle envisagée à la date de la première préparation des plannings de conduite et celle présente à l'approche de la date de réalisation, plusieurs facteurs peuvent également survenir :

- l'état de l'équipement peut avoir évolué (dégradation, manipulation intempestive, remplacement de composants, etc.) ;
- l'état du système « global » peut avoir évolué ;
- des procédures prioritaires peuvent avoir été mises en œuvre.

Ces modifications peuvent impacter le respect des exigences de sécurité du système.

L'objectif est donc de fournir à l'équipe de conduite hors-quart des outils d'aide à la décision leur permettant de préparer les plannings d'intervention, et de les réactualiser, en tenant compte des enjeux de sécurité et de disponibilité prévisionnelle des matériels.

Ces outils pourront revêtir plusieurs formes telles que :

- la génération d'une ou plusieurs séquences de conduite permettant d'atteindre une situation d'exploitation souhaitée depuis une situation d'exploitation courante en tenant compte des indisponibilités des matériels consignées dans les plannings de maintenance ;
- la vérification d'une séquence de conduite au regard des contraintes de sécurité, en particulier consignées dans les Spécifications Techniques d'exploitation ;

– la fourniture d'indicateurs permettant d'évaluer la pertinence d'une séquence de conduite au regard de l'état de l'installation ou de son bilan de santé.

Un pré-requis à la fourniture de ces services d'aide à la décision est de disposer d'un modèle de fonctionnement de la tranche sur lequel les analyses ou la génération des séquences de conduite pourront être réalisées.

Deux phases d'exploitation bien différentes sont à distinguer :

– une phase de réalisation des activités avec suivi temps réel des écarts vis-à-vis des Spécifications Techniques d'Exploitation (STE) et aide au changement d'état. Dans cette phase, il est important de bien distinguer les activités TeM versus AT, au sein de la cellule Arrêt de Tranche ou non ;
– une phase de préparation modulaire des chantiers.

Les travaux à réaliser TeM résultent : des demandes d'interventions, des activités planifiées automatiquement et des activités reprogrammées parce qu'elles n'ont pas pu être effectuées en raisons d'aléas. La liste des activités à réaliser par quart est concrètement élaborée par une équipe de planificateurs, encadrée par l'Ingénieur Tranche En Fonctionnement (ITEF), qui évalue la pertinence et la priorité des travaux à réaliser en concertation avec le Chef d'Exploitation. La FIGURE 1.5 présente l'organisation globale des différents services concernés par cette planification sur un site de production EDF.

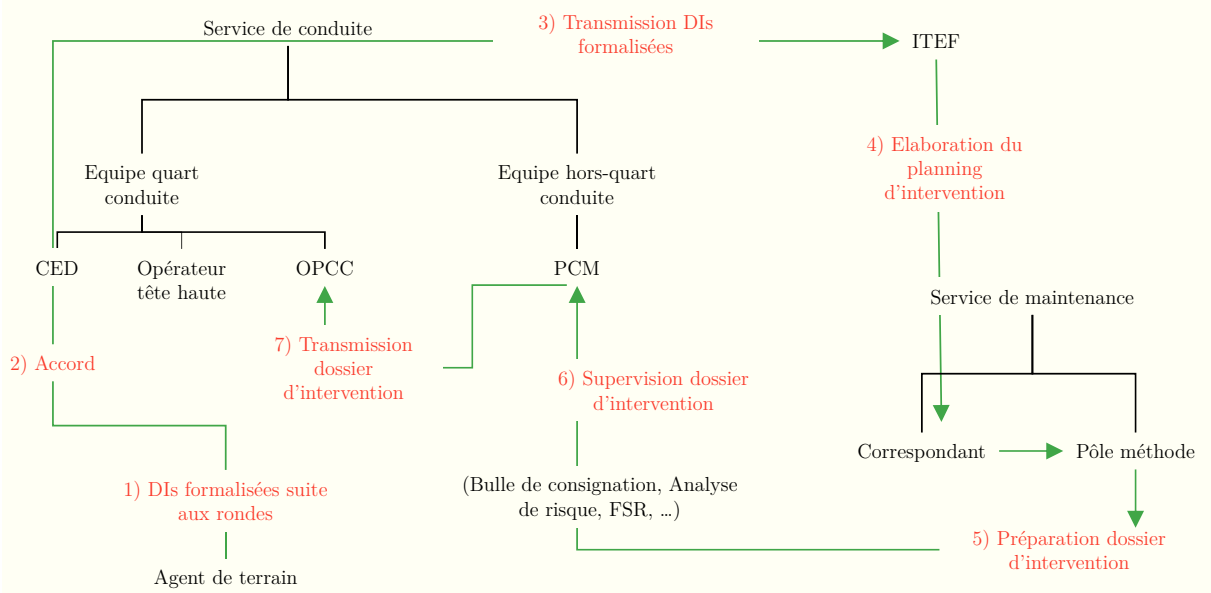


FIGURE 1.5 – Exemple d'organisation du service de conduite (Source EDF)

Les Demandes d'Interventions (DI) de maintenance sont pour la plupart formalisées par les équipes de quart de la conduite à l'issue des rondes des agents de terrain, puis affinées et validées par les Chefs d'Exploitation (CE) et Chefs d'Exploitation Délégués (CED). Les DIs validées sont ensuite transmises à l'Ingénieur Tranche En Fonctionnement (ITEF) qui les intègre à une planification dans laquelle sont également intégrées les activités prévisibles que sont les essais périodiques et les opérations de maintenance préventive. Ces DIs sont agrégées aux activités de maintenance prévisibles et à celles qui ont été reprogrammées. Acceptées et

priorisées, ces DIs parviennent au pôle préparation du métier de maintenance, qui élabore les documents (Analyse de Risques, Régimes, Fiche de Simple Requalification - FSR, gammes...) nécessaires à l'intervention. Cette structure fait ensuite transiter le dossier à l'équipe de Conduite Hors quart qui vérifie la compatibilité des activités entre elles, contrôle et complète le dossier avant de le faire parvenir à l'équipe de Conduite Temps réel.

1.1.2.2 Aide à la vérification de la sécurité d'architectures en ingénierie

En phase d'ingénierie d'une centrale, la conception de l'architecture doit tenir compte d'exigences de sûreté de fonctionnement, notamment de sécurité. Cela se traduit au niveau de l'architecture par des redondances fonctionnelles, et par la mise en place de systèmes de secours. Le respect des exigences de disponibilité nécessite, au niveau de l'ingénierie de la conduite, de s'assurer que le système sera bien en mesure d'être conduit dans diverses situations, comme dans le cas de l'indisponibilité de certains équipements suite à une panne ou à une condamnation. Également, il faut s'assurer que les opérateurs seront en mesure de conduire le procédé. Pour cela, il est nécessaire de prendre en compte les facteurs socio-organisationnels et humains au plus tôt dans la conception [Devic and Morilhat, 2013]. Il est également possible d'intégrer les opérateurs dès les phases amont d'ingénierie [Bouffaron et al., 2014] de manière à ce qu'ils puissent valider, d'un point de vue opérationnel, les propositions qui sont faites, par exemple en utilisant des simulateurs [Blanchon and Bruneau, 2016].

> *Pour maîtriser ces risques et mettre au point l'organisation de l'équipe de conduite, un programme d'ingénierie « facteur humain intégré » est mis en place dès les phases amont d'un projet. L'équipe pluridisciplinaire (opérateurs, ingénieur de fonctionnement, ingénieurs sûreté, facteurs humains) qui en a la charge est la première à mettre en place les moyens de simulation du procédé et des scénarios dimensionnant et permettant de définir et d'évaluer les principes de conduite. [Devic, 2016]*

De manière complémentaire, au cours de la vie de la centrale, des modifications fonctionnelles peuvent intervenir pour permettre, par exemple, d'augmenter son niveau d'intégrité. Il peut s'agir d'ajout de conditions d'ouverture ou de fermeture de vannes, de conditions de mise en service de pompes ... Ces modifications peuvent dans une certaine mesure impacter la conduite du système, mais ne doivent en aucun cas réduire le nombre de situations atteignables en conduite normale avant la modification.

Il convient donc de s'assurer, à la fois en ingénierie lors de la conception initiale de la centrale, et à la fois avant toute modification de l'installation, que les situations devant être atteintes lors de la conduite sont effectivement atteignables.

Pour cela, le Cluster CONNEXION a envisagé de mettre en avant de nouveaux usages de la simulation, couplée à de la génération de séquences d'actions de conduite. L'idée est de générer une séquence d'actions de conduite sur un modèle de l'installation, incluant les choix de conception et les états de disponibilité des équipements, afin de vérifier dans un premier temps qu'une situation donnée est atteignable :

– S'il n'est pas possible de générer de séquence, autrement dit que la situation n'est pas atteignable, cela signifie que la conception ou la modification n'est pas valide, et qu'il faut

en envisager une autre ;
- Dans le cas où il existe une séquence d'actions de conduite possible pour atteindre la situation, cette séquence est exécutée sur un modèle simulable de la centrale. Le résultat de cette exécution est observé de manière à vérifier que l'état du système respecte les conditions de sécurité tout au long de l'exécution de la séquence d'actions :
  - Si ce n'est pas le cas, alors la séquence générée n'est pas valide d'un point de vue de la sécurité, il faut alors en générer une autre ;
  - Si c'est le cas, il est alors possible de conclure que la situation est atteignable, et que la conception ou la modification est valide d'un point de vue de la sécurité.

1.1.2.3 Contraintes en ingénierie

Dans le cadre de l'ingénierie de la conduite des systèmes de grande taille, plusieurs difficultés sont à envisager. Tout d'abord, ces systèmes possèdent de nombreux points de fonctionnement, auxquels il est possible de parvenir par différents scénarios, dépendants de l'état et de la disponibilité des composants du système. La définition des procédures de conduite est donc une opération qui va se répéter de nombreuses fois en phase de préparation, et pour de multiples architectures en phase d'ingénierie. De plus, la taille de ces systèmes fait que les modèles qui les représentent sont également de grande taille, et donc que les temps nécessaires à la modélisation sont longs.

Cependant, les ressources humaines et temporelles disponibles pour la préparation et l'ingénierie étant limités, il est nécessaire de faciliter le travail de modélisation. Partant du constat que ces systèmes sont souvent construits hiérarchiquement selon des niveaux utilisant des familles de composants similaires, une manière de procéder est de structurer les modèles de manière modulaire afin de permettre leur réutilisation. En effet, les travaux dans le domaine de la modélisation de partie opérative montrent qu'une certaine similitude se dégage entre les comportements des équipements d'une même famille. Il semble donc envisageable d'obtenir des modèles génériques et paramétrables pour couvrir certaines différences comportementales mineures. Ceci induit que les formalismes utilisés pour la génération de séquences d'actions de conduite doivent :
- supporter une modélisation hiérarchique permettant de structurer les modèles selon le niveau d'architecture ;
- supporter une modélisation modulaire, réutilisable et paramétrable pour tirer parti des similitudes comportementales entre certains composants du système.

1.2 Formalisation du problème industriel

1.2.1 Caractérisation du contexte industriel adressé

Les travaux présentés dans ce manuscrit sont basés sur une problématique industrielle d'aide à la conduite de systèmes complexes critiques. Comme cela a été dit précédemment, ces systèmes sont organisés hiérarchiquement. Les trois niveaux hiérarchiques principaux retenus dans le cadre de ces travaux, et caractérisés dans la suite de ce paragraphe, sont :
- Equipements de terrain ;
- Fonctions ;

– Système.

1.2.1.1 Caractérisation des équipements manipulés

On note EQ l'ensemble des équipements composant un système tel que $EQ = \{eq_1, eq_2, \ldots, eq_n\}$, où $n \in \mathbb{N}$ est ici le nombre d'équipements composant le système. Par équipement, on entend tous les éléments manipulables (par des opérateurs ou par le contrôle-commande, sans distinction), c'est-à-dire des pompes et des vannes.

Un équipement eq_i est caractérisé par un couple $(state, status)$, où $state$ correspond à l'état (ouvert, fermé, en marche, ...) de l'équipement et $status$ correspond à sa disponibilité opérationnelle (condamné, consigné, ...).

Exemple :
Exemples de « state » : ouvert, fermé, enclenché, ...
Exemples de « status » : disponible, condamné, ...

L'ensemble EQ peut être décomposé en deux sous-ensembles distincts :
– L'ensemble EQ^P correspondant aux pompes
– L'ensemble EQ^V correspondant aux vannes

Chaque évolution du $state$ ou du $status$ d'un équipement eqt_i est provoquée par une action de conduite. On note A^{eq_i} l'ensemble des actions réalisables sur un équipement eq_i tel que $A^{eq_i} = \{a_1^{eq_i}, a_2^{eq_i}, \ldots, a_n^{eq_i}\}$, où $n \in \mathbb{N}$ est ici le nombre d'actions différentes qui peuvent être opérées sur un équipement. La réalisation de (certaines de) ces actions peut être soumise à certaines conditions de sécurité, dues par exemple à la condamnation administrative d'un équipement ou à des contraintes physiques.

Exemple :
Une condition de sécurité pour l'enclenchement d'une pompe est par exemple l'ouverture de l'ensemble des vannes amont afin de permettre l'arrivée d'un fluide.

1.2.1.2 Caractérisation des fonctions

On note F l'ensemble des fonctions constituant un système, tel que $F = \{f_1, f_2, \ldots, f_n\}$, où $n \in \mathbb{N}$ est le nombre de fonctions constituant le système global. On note EQ_{f_i} l'ensemble des équipements à configurer pour la fonction f_i.

Une fonction f_i peut être en exécution ou non, selon la configuration C^{F_i} de différents équipements de EQ_{f_i}. On appelle lignage d'une fonction l'ensemble des actions nécessaires pour obtenir sa configuration. Parmi les équipements à configurer, on distingue quatre sous-ensembles d'équipements [2] :
– Un ensemble $P^{f_i} \subseteq EQ^P$ de pompes ;
– Un ensemble $V^{f_i} \subseteq EQ^V$ de vannes découpé en trois sous-ensembles :

2. On admet que $V_{f_i}^{amont} \cup V_{f_i}^{aval} \cup V_{f_i}^{ferme} = V_{f_i}$, que ces trois ensembles sont disjoints, et que $P_{f_i} \cup V_{f_i} = EQ_{f_i}$

- L'ensemble $V^{f_i}_{amont} \subseteq V^{f_i}$ de vannes à ouvrir pour qu'un fluide puisse arriver depuis une source jusqu'à une pompe ;
- L'ensemble $V^{f_i}_{aval} \subseteq V^{f_i}$ de vannes à ouvrir pour qu'un fluide puisse arriver depuis une pompe jusqu'à une destination ;
- L'ensemble $V^{f_i}_{ferme} \subseteq V^{f_i}$ de vannes à fermer pour que le fluide soit contenu.

On note ϕ l'ensemble des variables physiques du procédé considéré. Une fonction f_i, une fois configurée, influe sur une ou plusieurs variables physiques de l'ensemble $\phi^{f_i} \subseteq \phi$, où $\phi^{f_i} = \{\varphi^{f_i}_1, \varphi^{f_i}_2, \ldots, \varphi^{f_i}_n\}$, avec $n \in \mathbb{N}$ le nombre de valeurs physiques sur lesquelles l'exécution de la fonction provoque un changement de la valuation.

1.2.1.3 Caractérisation du système global

Le système considéré est caractérisé par sa recette R, qui correspond à l'ensemble SIT des situations stabilisées tel que $SIT = \{sit_1, sit_2, \ldots, sit_n\}$, où $n \in \mathbb{N}$ est ici le nombre de situations stabilisées définies, et des transitions possibles entre situations stabilisées. Chacune de ces situations est elle-même caractérisée par un ensemble de contraintes sur les valuations d'un ensemble $\phi^{sit_i} \subseteq \phi$ de variables physiques, qui doivent appartenir à des intervalles définis $[\varphi_{min}, \varphi_{max}]$.

Ainsi, à tout instant, pour que le système soit considéré dans une situation i, toutes les variables physiques telles que $\varphi \in \phi^{sit_i}$ doivent respecter $\varphi^{sit_i}_{min} \leq \varphi \leq \varphi^{sit_i}_{max}$.

1.2.2 Formalisation du problème

L'objectif des opérateurs est de conduire le procédé, en fonction d'objectifs de conduite, tout en respectant des conditions de sécurité. Pour cela, ils suivent des scénarios de conduite. Pour les guider dans ces scénarios, les procédures de conduite contiennent à la fois des actions à opérer et des observations à effectuer sur le procédé. Avant leur mise en place opérationnelle, les procédures de conduite sont qualifiées, c'est-à-dire qu'elles ont été approuvées par un expert d'un point de vue de la sécurité.

On propose ici de faire une abstraction de ces procédures de conduite en **séquences d'actions** dans lesquelles on ne considère que les actions à effectuer, sans faire apparaitre les observations. Une séquence d'actions Seq correspond donc à une suite ordonnée d'actions $a^j_{eqt_i}$.

A partir des éléments caractérisés dans cette section 1.2, le problème industriel à résoudre est donc celui de la génération de séquences d'actions de conduite, avec trois objectifs complémentaires :

- génération d'une séquence admissible : pour montrer la faisabilité de la conduite d'une situation à une autre, ou pour valider une modification fonctionnelle du système à conduire ;
- génération d'une séquence optimisée : pour conduire le système de manière optimale ;
- génération d'un ensemble de séquences : pour permettre un choix entre plusieurs séquences possibles.

Ce problème de génération de séquences nous a menés à nous poser principalement deux questions, pour lesquelles la section suivante présente un état de l'art :

- comment modéliser et générer des procédures de conduite ?

– à partir de quels modèles du système à conduire ?

1.3 Approches pour la génération de séquences d'actions de conduite

1.3.1 Modélisation des procédures de conduite

1.3.1.1 Multilevel Flow Modeling

De nombreux travaux ont déjà porté sur la conduite et la modélisation des procédés industriels critiques, entre autres dans le domaine de la chimie et de l'énergie. Dans un objectif de modélisation fonctionnelle des installations, on peut noter en particulier les travaux de M. Lind sur le formalisme Multilevel Flow Modeling [Lind, 2011a].

> *Multilevel Flow Modeling (MFM) is a methodology for modeling of industrial processes on several interconnected levels of means and part-whole abstractions [Lind, 2011a]*

> *Multilevel Flow Modeling (MFM) is an approach to modeling goals and functions of complex industrial processes involving interactions between flows of mass, energy and information [Saleem and Lind, 2009]*

> *The level of details and abstraction of the model must comply with the needs of the task to be solved [Saleem and Lind, 2009]*

Dans un objectif de modélisation fonctionnelle des installations, les travaux de M. Lind sur le formalisme Multilevel Flow Modeling [Lind, 1990, Lind, 2011a], utilisés par exemple dans le cadre d'une modélisation de la centrale nucléaire de Monju [Lind et al., 2011], s'intéressent à la décomposition des fonctions principales en fonctions élémentaires (transport, stockage, ...), ainsi qu'aux flux (matière, énergie et information) échangés. Ils ont principalement été utilisés dans le cadre d'analyses causes-effets [Lind, 2011b] sur la base de modèles réalisés à l'aide des différents éléments de modélisation présentés en FIGURE 1.6. Un lien peut être fait avec les différents organes modélisés dans [Lind, 1994], mais sans représentation évidente de l'organe considéré.

1.3.1.2 Méthode ASTRID

Un lien entre vues fonctionnelle et organique est fait au travers la norme ISA/S88 (évoquée précédemment [ISA, 1998]), qui est à la base de différents travaux sur les procédés batch. Parmi eux, la méthode orientée objet ASTRID (Analyse STRucturée pour l'Industrialisation des ateliers Discontinus), qui résulte d'une analyse critique de cette norme, consiste à décrire l'installation et les modes opératoires en sous-ensembles matériels et fonctionnels. Elle est utilisée pour l'analyse des procédés industriels en vue de leur automatisation. Parmi les objets décrits dans cette approches, les procédures présentent le séquencement des fonctions à mettre en œuvre, chaque fonction étant associée à des ressources et chaque ressource à des organes de plus bas niveau (FIGURE 1.2).

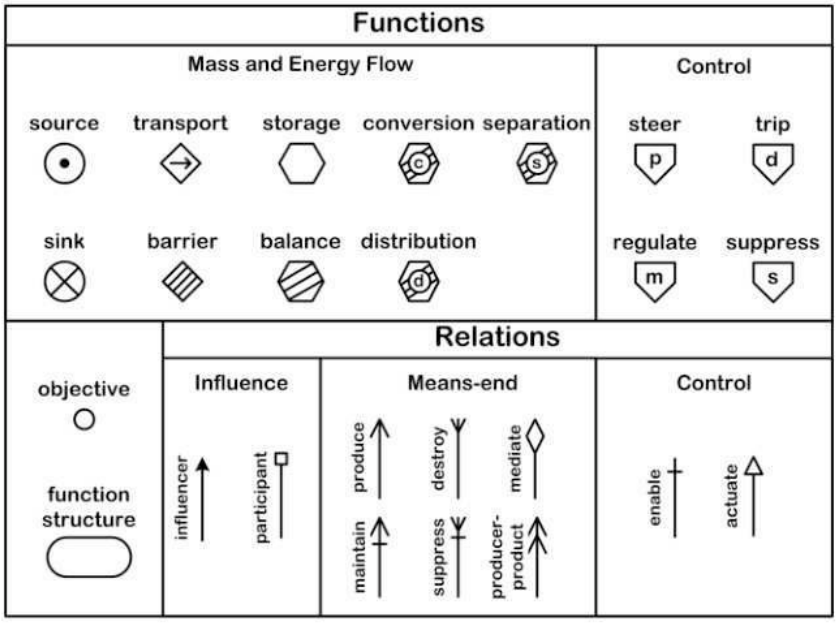


FIGURE 1.6 – Artefacts de modélisation du formalisme MFM

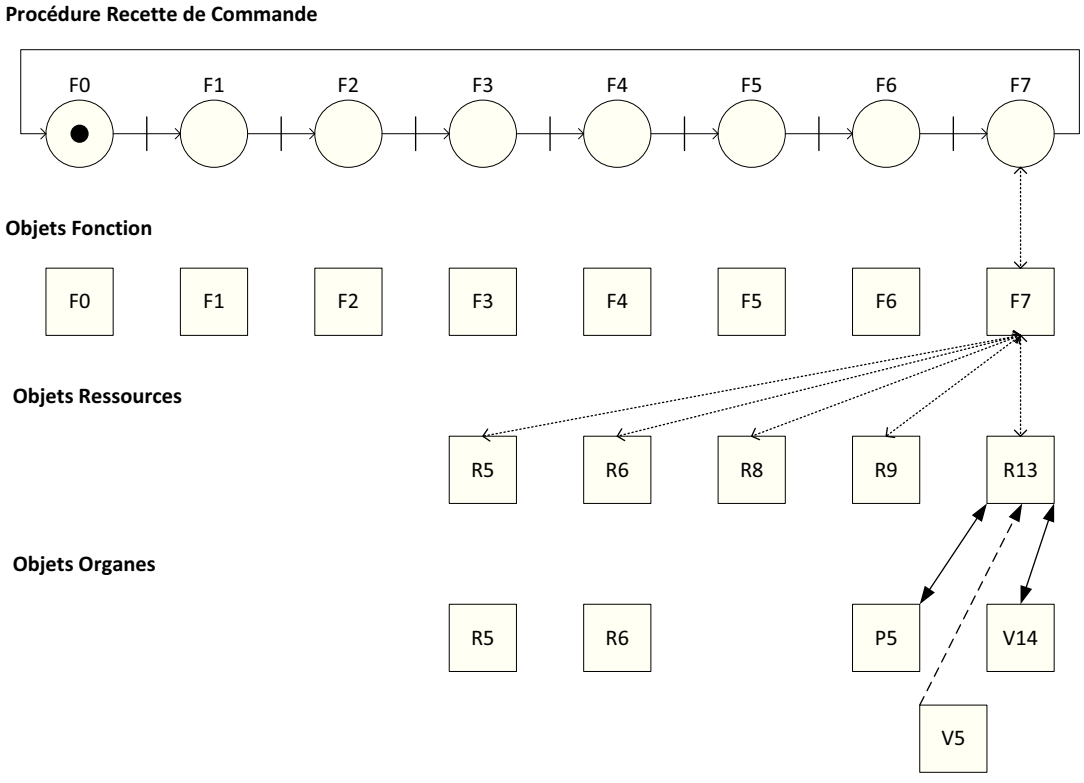


FIGURE 1.7 – Exemple de procédure ASTRID montrant le lien avec les fonctions, ressources et organes à utiliser

1.3.1.3 Grafcharts

De manière complémentaire à la méthode ASTRID, les procédures de conduite des procédés batch ont également fait l'objet d'autres travaux de représentation graphique, comme ceux de [Castelnuovo et al., 2007] à l'aide des réseaux de Petri, et comme ceux de [Arzen and Johnsson, 1996, Viswanathan et al., 1998a, Viswanathan et al., 1998b] qui introduisent le formalisme Grafchart. Les Grafcharts sont une extension des Grafcets, adaptée à la modélisation séquentielle et hiérarchique de procédures.

Un fort intérêt de ce formalisme, contrairement aux différentes représentations citées précédemment, est l'intégration d'un mécanisme d'abstraction appelé « macro step ». Une macro step est proche d'une macro étape d'un grafcet, qui permet de décomposer graduellement la modélisation du système, ajoutant une structuration intéressante au modèle (voir FIGURE 1.8). L'exécution des modèles peut être utilisée apporter une aide à la formation des opérateurs, en augmentant progressivement le niveau de détail des opérations de conduite.

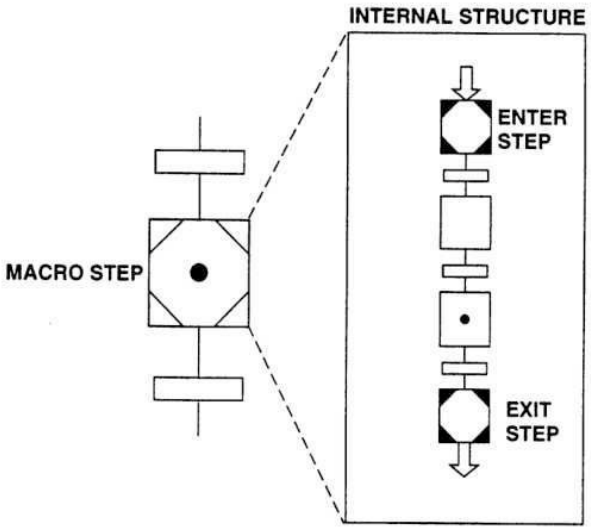


FIGURE 1.8 – Exemple de macro step appliqué au formalisme Gracharts

Ce formalisme n'est toutefois prévu que pour la représentation de séquences existantes, et non pour la génération ou la vérification.

Conclusion

Si les travaux précédents ont l'avantage de permettre la représentation d'un procédé et de séquences d'actions de conduite, le fait que ces langages soient semi-formels ne permet pas, d'une part, de modéliser les aspects dynamiques, et, d'autre part, d'utiliser des approches de génération automatique. Pour ces raisons, on s'intéresse dans la section suivante à des modèles et méthodes formels avec lesquels il est possible d'envisager de générer automatiquement des séquences d'actions de conduite.

1.3.2 Génération des procédures de conduite par apprentissage

Une première possibilité d'obtenir des procédures de conduite est de se baser sur le savoir-faire d'experts en conduite, par exemple des opérateurs ayant une longue expérience opérationnelle. Ce savoir-faire doit être capitalisé sous une forme réutilisable par d'autres personnes, avec pour contrainte de limiter les possibilités d'interprétations et d'incompréhension qui peuvent être dues aux différences de culture technique entre différents intervenants (opérateurs, concepteurs ...).

Pour éviter cela, et pour faciliter l'écriture de spécifications qui peuvent être fastidieuses dans le cas de systèmes de grande taille, les travaux de [Goubali, 2017] proposent, plutôt que de les intégrer tardivement en phases avales de validation, de mettre au plus tôt dans la boucle d'ingénierie les utilisateurs finaux. Ils montrent pour cela la faisabilité d'une approche d'obtention de spécifications fonctionnelles validées par des experts métiers conception qui utilise les techniques du End User Development.

Ces techniques de End User Development ont pour objectif de permettre aux utilisateurs finaux de développer et adapter eux-même les systèmes qu'ils utilisent, sans nécessiter de connaissances et compétences particulières en conception / programmation.

> *End-User Development can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact.[Lieberman et al., 2006]*

Partant de ce principe, [Goubali, 2017] utilise de manière complémentaire les techniques de l'ingénierie dirigée par les modèles pour proposer, à partir des manipulations effectuées par les opérateurs, une approche de génération automatique d'une interface de spécification (intégrant un Enregistreur, un Généralisateur, un Rejoueur, et un Correcteur), l'interface de supervision du système à piloter et son programme de commande.

1.3.3 Génération des procédures de conduite à l'aide de modèles

1.3.3.1 Approche par résolution de contraintes

Les premiers travaux portant sur l'obtention des procédures furent notamment ceux de [Rivas and Rudd, 1974], qui furent parmi les premiers à proposer une approche de génération d'actions sur des vannes.

> *Methods are developed for the computer-aided synthesis of sequences of valve operations to reach complex operation goals with safety. Given dangerous events which must not occur and operation goals to be reached, sequences of valve openings and closings are formed rapidly for industrially significant problems. [Rivas and Rudd, 1974]*

Ces travaux ont par la suite été prolongés entre autres par [Foulkes et al., 1988] qui ont proposé une méthode de génération automatique de trajectoires dans des systèmes plus complexes, en se basant sur les changements d'états calculés à partir des situations initiales et attendues d'une installation, en tenant compte de règles de sécurité, sans toutefois tenir compte d'éléments structurels au niveau de l'installation ni de grandeurs physiques.

D'autres approches, reposant sur l'atteinte de différents objectifs intermédiaires avant d'atteindre un objectif final, ont été proposées par [Fusillo and Powers, 1987] et [Fusillo and Powers, 1988] en vue de la planification sous contraintes de procédures [Li et al., 1997]. Il s'agit principalement d'approches basées sur des heuristiques ou des techniques de recherche opérationnelle.

Cependant, ces approches ne couvrent que difficilement les aspects modulaires de la modélisation (niveaux hiérarchiques, équipements, fonctions ...) et les comportements dynamiques des architectures de conduite (consignation des matériels par exemple).

1.3.3.2 Approches à base de modèles de Systèmes à Evènements Discrets

Les approches basées sur la théorie et les formalismes des Systèmes à Evènements Discrets [Cassandras and Lafortune, 2008] apportent des réponses aux limites des approches citées ci-dessus. En effet, elles permettent de représenter de manière formelle la structure et le comportement d'une installation. De par leur syntaxe et leur sémantique, les langages formels de modélisation des Systèmes à Evènements Discrets permettent de simuler et d'analyser formellement les modèles et en particulier de vérifier des propriétés (sécurité, atteignabilité ...). Le respect de ces propriétés peut être assurée *a priori* par construction, à l'aide par exemple des approches de synthèse de la commande, ou vérifié *a posteriori* par model-checking.

Les techniques de synthèse automatique de la commande [Ramadge and Wonham, 1987, Kumar, 1992, Zaytoon and Riera, 2017] permettent d'obtenir *a priori* l'ensemble des trajectoires possibles dans un modèle de commande [Yeh and Chang, 2012], tout en respectant un ensemble de spécifications comportementales, relatives à des aspects fonctionnels et de sécurité. Elles ont montré leur intérêt dans les travaux de [Qiu, 2005] et [Pétin et al., 2007] dans le cadre de la reconfiguration de systèmes manufacturiers. Pour rechercher l'atteignabilité d'une situation donnée, ces techniques de synthèse ont également été employées pour la génération de séquences d'actions. Il s'agit par exemple de l'utilisation de réseaux de Petri dans les travaux de [Wang et al., 2005], qui proposent une technique de génération de procédures par synthèse, appliquée au cas d'étude utilisé par [Foulkes et al., 1988]. Aucun mécanisme d'agrégation n'est cependant traité, malgré un nombre important d'éléments à considérer. Une autre approche, utilisant les automates à états pour la génération de séquences d'actions de conduite, est proposée par [Yeh and Chang, 2012]. Elle permet la génération de l'ensemble des chemins menant à un objectif. Ce résultat n'est pas directement utilisable pour la conduite dans le sens où il est nécessaire en conduite de connaitre de manière déterministe, à partir d'une situation, la prochaine action à exécuter. Ainsi, ces approches par synthèse posent le problème d'un critère de choix pour déterminer le chemin à suivre parmi l'ensemble des chemins possibles.

Une autre technique permettant de vérifier *a posteriori* l'atteignabilité d'une situation souhaitée est le model checking. Le model checking [Clarke and Wing, 1996, Clarke et al., 1999] est une technique automatique de parcours de l'espace d'état d'un modèle formel qui permet la vérification de propriétés exprimées dans une logique formelle dans le but de s'assurer qu'une spécification est respectée par un modèle ou un programme [Niang et al., 2017].

L'utilisation du model checking requiert trois éléments : un modèle formel, la spécification d'une propriété à vérifier, et une méthode de vérification permettant l'exploration de toutes les

situations possibles et pour laquelle la propriété doit être vérifiée (FIGURE 1.9).

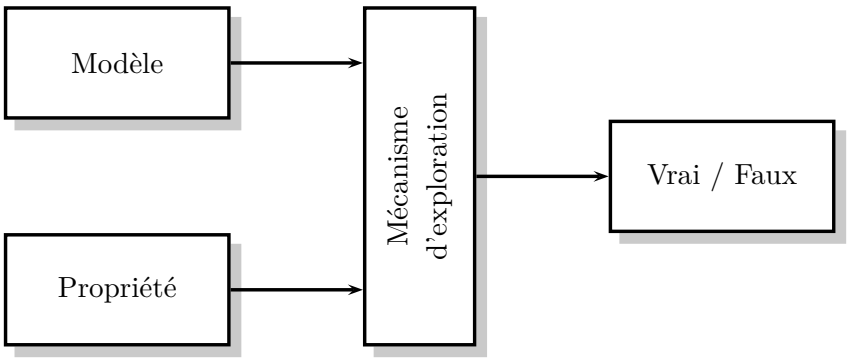
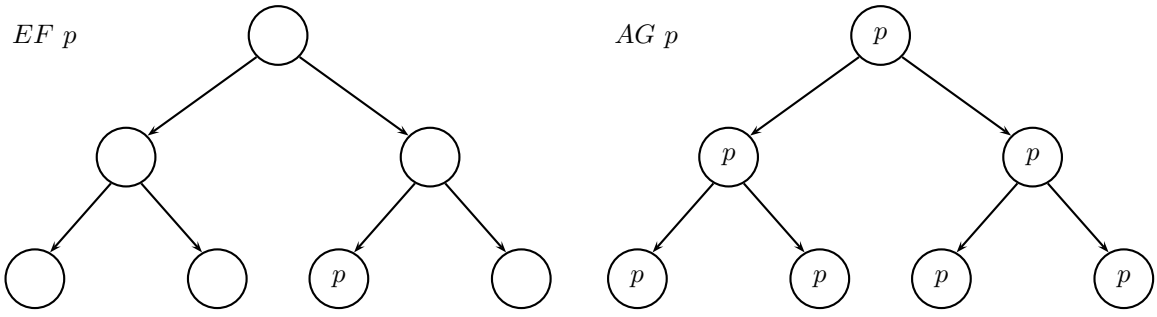


FIGURE 1.9 – Méthodologie de vérification de propriété sur un modèle

La première tâche est de construire un modèle formel du système à vérifier. Pour cela, les formalismes des Systèmes à Evènements Discrets sont utilisables. Ensuite, il est nécessaire de spécifier les propriétés à vérifier. Cette spécification peut être donnée suivant diverses logiques [McMillan, 1993] : Computation Tree Logic (CTL), Linear Temporal Logic (LTL), ... On manipule principalement deux types de propriétés à l'aide de ces logiques :

- les propriétés d'invariance, c'est-à-dire s'assurer qu'une propriété p est toujours vraie, quelque que soit l'état du système, que l'on exprime $AG\ p$ en logique CTL ; il s'agit souvent de propriétés liées à la sécurité ;
- les propriétés d'atteignabilité, c'est-à-dire vérifier que la propriété p peut être vraie à un moment donné, que l'on exprime $EF\ p$ en logique CTL.

Ces deux types de propriétés sont illustrés par la FIGURE 1.10.



(a) Propriété d'atteignabilité (b) Propriété d'invariance

FIGURE 1.10 – Illustration de propriétés vérifiées par model checking

Enfin, on vérifie que le modèle respecte la propriété exprimée. Lors de l'exploration de l'espace d'état, le model checking s'assure que la propriété exprimée reste vraie. Un résultat négatif dans le cas d'une vérification de propriété d'invariance amène généralement à la génération d'une trace d'exécution qui a mené à violer la spécification ; il s'agit d'un contre-exemple qu'il faut interpréter de manière à comprendre d'où vient l'erreur de modélisation [Mesli Kesraoui, 2017]. Un résultat positif dans le cas d'une recherche d'atteignabilité amène à la génération d'une trace d'exécution qui, dans le cas qui nous intéresse de génération de séquences d'actions de conduite,

permet de déterminer un exemple de séquence admissible. La spécification des règles de conduite de façon formelle à partir de modèles automates à états intégrant les règles de sécurité permet d'assurer le respect complet de ces règles lors de l'évolution de l'état du procédé.

Les vérifications peuvent s'effectuer de manière automatique à l'aide d'outils logiciels appelés model checkers. Ces outils informatiques permettent l'exploration de l'espace d'état défini par la composition des différents modèles manipulés. Selon les cas, différents outils de vérification sont disponibles, les plus populaires étant SPIN [Holzmann, 1997], CPN Tools [Jensen et al., 2007], NuSMV [Cimatti et al., 2002], ou Uppaal [Behrmann et al., 2001, Behrmann et al., 2006a].

Par rapport aux techniques de synthèse formelle, les approches par model-checking ont l'avantage de proposer une seule séquence parmi l'ensemble des possibles, et donc de ne pas poser le problème du choix. Cet intérêt est renforcé dans la mesure où, comme l'ont montré les travaux de [Marangé et al., 2011], la séquence générée peut être très proche de l'optimum. L'approche peut également être utilisée de manière itérative pour générer un ensemble de séquences d'actions de conduite possibles, conformément au problème formalisé plus haut. De manière complémentaire, [Li et al., 2014] montre l'intérêt de ce type d'approche pour la génération de procédures cycliques d'actions de conduite. D'autres travaux, issus du Commissariat à l'Énergie Atomique, ont également montré l'intérêt des méthodes de model-checking pour la génération de scénarios de tests [Bigot et al., 2003].

Conclusion

L'utilisation des techniques de model checking pour générer des séquences d'actions de conduite par recherche d'atteignabilité est un choix qui a montré son potentiel dans différents travaux ayant des objectifs industriels proches de ceux présentés ici [Li et al., 2014]. Nous proposons donc d'utiliser cette approche par recherche d'atteignabilité, et présentons dans la section suivante des formalismes de Systèmes à Evènements Discrets potentiellement utilisables pour la modélisation du système à conduire.

1.3.4 Formalisme de modélisation du système à conduire

En vue de modéliser le système à conduire, plusieurs aspects sont à prendre en compte. Tout d'abord, les systèmes étant de grande taille, leur structure est souvent organisés hiérarchiquement, et il arrive que des maillages interviennent. Ensuite, comme l'on cherche à générer des séquences d'actions de conduite, il est nécessaire d'en modéliser le comportement dynamique, notamment de manière discrète, tout en incluant la représentation du temps.

1.3.4.1 Statecharts

Pour faciliter la représentation de l'organisation hiérarchique des systèmes considérés, les Statecharts, formalisme de description « visuelle » de systèmes complexes réactifs [Harel, 1987b, Harel, 2007], peuvent être envisagés. Les statecharts sont une extension des diagrammes états-transitions utilisant des mécanismes supplémentaires : hiérarchie, concurrence et communication. Il est également possible de représenter le temps à l'aide de timers explicites (par exemple : *timeout (event,number)*). La notion de hiérarchie étant prépondérante dans les systèmes considérés, il s'agit là d'un point fort appuyant le choix d'un formalisme de

modélisation, facilitant l'agrégation des états. Cependant, le maillage inter-niveaux, c'est-à-dire le fait qu'un même équipement puisse appartenir à plus d'une fonction (FIGURE 1.11), ne peux être représenté en statecharts. Comme indiqué dans [Harel and Kahana, 1992], cette composition apporte trop de problèmes de sémantique formelle pour être envisagée.

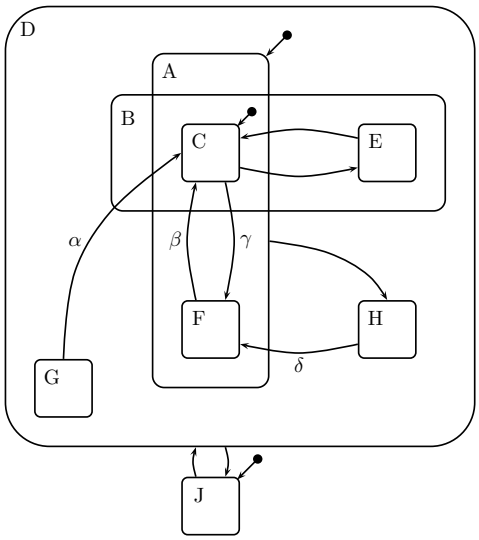


FIGURE 1.11 – Overlaps [Harel and Kahana, 1992] : l'état C fait partie de A et de B

En complément de cette description « visuelle », il est nécessaire de considérer un algorithme d'interprétation de ces modèles [Lüttgen and Mendler, 2002] en vue de rendre le comportement modélisé exécutable. Pour cela, plusieurs sémantiques ont été définies [Maggiolo-Schettini et al., 2003, Eshuis, 2009], parmi lesquelles :

- la vision externe [Harel, 1987a], qui correspond à une interprétation sans recherche de stabilité ;
- la vision micro-pas [Pnueli and Shalev, 1991], qui correspond à une interprétation avec recherche de stabilité ;
- l'hypothèse synchrone [Harel et al., 1990].

1.3.4.2 Réseaux de Petri

Les Réseaux de Petri (RdP) sont un formalisme introduit par Carl Adam Petri [Petri, 1966]. Ils permettent la représentation et la simulation de différents aspects de la structure et du comportement de modèles de Systèmes à Evènements Discrets [David and Alla, 1992, Rene and Hassane, 2005].

Un réseau de Petri [Cassandras and Lafortune, 2008] est défini formellement par $PN = (P, T, A, \omega)$, où :

- P est un ensemble de places ;
- T est un ensemble de transitions ;
- $A \subseteq (P \times T) \cup (T \times P)$ est l'ensemble des arcs allant d'une place à une transition et d'une transition à une place ;
- $\omega : A \to \{1, 2, 3, ...\}$ est une fonction de pondération des arcs.

L'utilisation des RdP est très répandue en recherche [Murata, 1989, Jensen et al., 2007]. Sa base formelle mathématique en fait un outil puissant pour la vérification de propriétés sur un modèle d'un système complexe. Dans ce formalisme, le comportement d'une vanne peut être représenté comme le propose la FIGURE 1.12 [Wang et al., 2005]. Dans cet exemple, les places *PV(O)* et *PV(C)* représentent respectivement les états ouverts et fermés de la vanne, atteints par franchissement des transitions *TV(O)* et *TV(C)* qui représentent respectivement les actions d'ouverture/fermeture de la vanne. Les places *PA(O)* et *PA(C)* permettent de retenir le nombre de franchissements des transitions précédentes. Les places *PC(O)* et *PC(C)* modélisent des éventuelles conditions sur l'ouverture/fermeture d'une vanne.

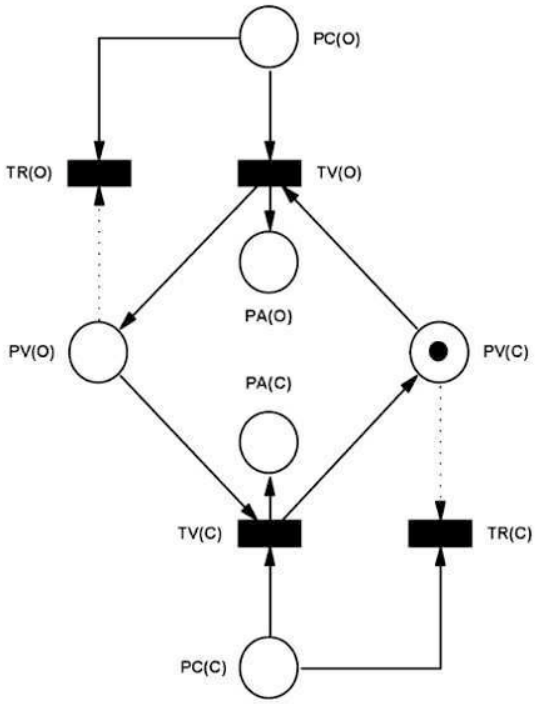


FIGURE 1.12 – Modèle RdP d'une vanne [Wang et al., 2005]

Il existe différentes classes de RdP, qui permettent la représentation d'aspects qui nous intéressent pour la génération de séquences d'actions de conduite. On peut notamment citer :

- les RdP temporisés : ils permettent de décrire les systèmes dont le fonctionnement dépend du temps : les RdP P-temporisés associent une durée de séjour aux places ; les RdP T-temporisés associent une durée de franchissement aux transitions ;
- les RdP synchronisés : ils utilisent la notion d'évènement permettant de modéliser des synchronisations ;
- les RdP interprétés : ils sont à la fois synchronisés et P-Temporisés ;
- les RdP colorés hiérarchiques [Jensen et al., 2007] : ils proposent une structuration modulaire dans laquelle des éléments du réseau sont eux-même composés d'un RdP.

Les modèles utilisés pour la génération de séquences nécessitant la prise en compte de la représentation du temps et des évènements, les RdP Interprétés et Colorés semblent des pistes intéressantes.

1.3.4.3 Automates à états finis

Les automates à états finis sont un formalisme états-transitions, défini par un langage (ensemble de mots) sur un ensemble d'évènements E. La définition formelle d'un automate à états finis déterministe est donnée [Cassandras and Lafortune, 2008] par $A = (X, E, f, \tau, x_0, X_m)$, où :

- X est un ensemble d'états,
- E est l'ensemble d'évènements associé à A,
- $f : X \times E \to E$ est une fonction de transition,
- $\tau : X \to 2^E$ est l'ensemble des évènements pour lesquels f(x,e) est défini,
- x_0 est l'état initial,
- X_m est un ensemble des états marqués.

Les automates à états temporisés [Alur and Dill, 1994] ont ensuite été définis comme une extension des automates à états finis, permettant la représentation du temps et la synchronisation entre modèles. Un automate temporisé $A = (S, V, X, L, I, T, S_m, s_0, v_0)$, est défini par :

- S est un ensemble fini de localités,
- V est un ensemble fini de variables entières,
- X est un ensemble fini d'horloges,
- I est un mapping qui étiquette chaque localité $s \in S$ avec un invariant d'horloges,
- L est un ensemble de labels (ou canaux) de synchronisation décomposé en deux ensembles disjoints de labels d'émission L_e (noté *label*!) et de réception L_r (noté *label*?),
- T est un ensemble de transitions $(s, l, g, m, s') \in S \times L \times G \times M \times S$ où G est un ensemble de gardes (conditions de franchissement sur les variables et les horloges) et M l'ensemble des mises à jour des valuations des variables et des horloges [3] ; l, g and m sont optionnelles,
- $S_m \subseteq S$ est l'ensemble des localités marquées,
- $s_0 \in S$ est la localité initiale,
- $v_0 \in V$ est la valuation initiale des variables.

Dans ce formalisme, il est possible de modéliser une vanne par ses deux états (une vanne peut être ouverte ou fermée), et des transitions montrant les évolutions possibles entre ces états (Figure 1.13). Les évènements *v1O ?* et *v1C ?* doivent être occurrents pour que la vanne change d'état. La variable *v1* permet de renseigner l'état courant de la vanne.

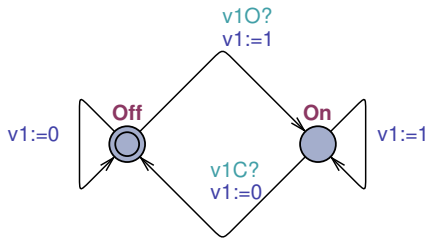


Figure 1.13 – Modèle automates à états temporisés d'une vanne [Li et al., 2014]

Le problème de maillage évoqué à la section précédente semble pouvoir être résolu à l'aide des automates à états temporisés, qui intègrent les notions de concurrence et de synchronisation.

3. Conventions de notation : les labels de synchronisation sont suivis de ! ou ?, les gardes sont entre crochets et les mises à jour de variable sont de la forme $v = valeur$

Diverses méthodes s'appuient sur les automates à états pour, par exemple, déterminer des allocations de fonctions à des organes de commande [Lemattre, 2013], générer automatiquement un ordonnancement [Marangé et al., 2011], ou vérifier des procédures [Li et al., 2014].

1.3.4.4 Choix d'un formalisme de modélisation

Les statecharts, les réseaux de Petri et les automates à états ont chacun des avantages et des inconvénients qui les rendent utilisables pour la modélisation du système et la démonstration des principes qui seront développés au cours de cette thèse. Afin d'en choisir un en particulier, la TABLE 1.1 synthétise leurs aptitudes à satisfaire les besoins identifiés précédemment.

Besoin	Statecharts	RdP	Automates temporisés
Réutilisation Modularité	+ Hiérarchiques	+ Instanciables, modulaires via les transitions de substitution	++ Utilisation de canaux de synchronisation, modèles modulaires instanciables
Exploration de l'espace d'état	- Il est nécessaire de choisir un algorithme d'interprétation, selon la sémantique choisie	+ Sémantique formelle des RdP, mais les RdPI nécessitent un algorithme d'interprétation, RdPC peuvent convenir	++ Sémantique formelle permettant construction et parcours de l'espace d'état
Exploitation de la trace	++ Séquences d'évènements	+ Séquences de tir (RdPC associés à événements, actions, variables ...)	++ Séquences d'évènements
Outillage pour la recherche d'atteignabilité	+ Statemate, Scade	+ CPNTools	++ UppAal, SMV, Prism

TABLE 1.1 – Comparaison des différents formalismes présentés

Conclusion

Les trois formalismes présentés ici présentent des intérêts à la fois pour la modélisation du système mais également pour une détermination de traces d'exécution. Notre choix se porte sur les automates temporisés, car :

- ils sont formellement définis ;
- ils permettent une représentation simple et modulaire du système à conduire ;
- des premiers travaux les utilisent pour la génération de procédures ;
- l'outillage est plus simple d'utilisation que celui des RdP.

Conclusion : problématique scientifique

La thèse présentée dans ce manuscrit s'intéresse à l'ingénierie et la conduite des systèmes complexes critiques. Afin d'apporter, d'une part, une aide à la vérification de la sécurité d'architectures en ingénierie et, d'autre part, une aide à la préparation des actions de conduite, les travaux présentés ici ont pour objectif, selon les cas, de :

- générer une séquence admissible pour montrer la faisabilité de la conduite d'une situation à une autre, ou pour valider une modification fonctionnelle du système à conduire ;
- générer une séquence optimisée pour conduire le système de manière optimale ;
- générer un ensemble de séquences pour permettre un choix entre plusieurs séquences possibles.

Pour cela, nous proposons d'utiliser la recherche d'atteignabilité sur des modèles de type automates à états temporisés.

Une des principales contraintes identifiées est la capacité des modélisateurs à réaliser et à modifier facilement le modèle de l'installation à conduire (par le biais par exemple d'ajouts et de suppressions de composants). Cela nécessite que les modèles utilisés pour représenter l'installation soient homogènes, standardisés et réutilisables. Ce problème est adressé dans le Chapitre 2, qui présente un cadre de modélisation structuré.

Une seconde contrainte à prendre en compte, cette fois-ci liée à une limite classique des approches utilisant le model-checking, est le risque d'explosion combinatoire. Cela peut en effet poser problème lors du passage à l'échelle qui est nécessaire dans le cas de systèmes tels que ceux adressés par le projet CONNEXION, qui comportent plusieurs milliers d'équipements. En ce sens, le Chapitre 3 propose de réduire ce risque d'explosion combinatoire par une approche modulaire prenant en compte de manière itérative des parties du modèle à explorer pour la recherche d'atteignabilité.

Chapitre 2

Cadre de modélisation

Sommaire

Introduction

L'objectif de ce chapitre est de proposer un cadre formel de modélisation, reposant sur une représentation structurée hiérarchiquement, et utilisant un ensemble de modèles génériques, formalisés par des automates temporisés. Ce cadre permet la génération de séquences d'actions de conduite à l'aide de mécanismes de recherche d'atteignabilité. Un premier cas d'étude est utilisé dans ce chapitre afin d'illustrer l'approche proposée, et de l'évaluer.

2.1 Principes généraux pour la génération de séquences

L'approche de génération de séquences proposée repose sur une modélisation du procédé sous la forme d'un réseau d'automates temporisés et sur des séquences de conduite recherchées sous la forme d'une trace d'exécution. La génération des séquences s'obtient alors par la vérification formelle d'une propriété d'atteignabilité à l'aide de techniques de model-checking. Une typologie des contraintes prises en compte dans les modèles de procédé est proposée. Enfin, la section se termine par la description d'un exemple simple qui servira d'illustration aux différentes contributions de ce chapitre.

2.1.1 Modélisation

2.1.1.1 Modélisation du procédé

Le modèle d'un élément de procédé (un équipement, une fonction, une recette, ...) est formalisé sous la forme d'un automate temporisé, tel que défini par [Alur and Dill, 1994] comme un 9-uplet $\mathcal{A} = (S, V, X, L, I, T, S_m, s_0, v_0)$ où :

- S est un ensemble fini de localités,
- V est un ensemble fini de variables entières,
- X est un ensemble fini d'horloges,
- I est un mapping qui étiquette chaque localité $s \in S$ avec un invariant d'horloges,
- L est un ensemble de labels (ou canaux) de synchronisation décomposé en deux ensembles disjoints de labels d'émission L_e (noté *label*!) et de réception L_r (noté *label*?),
- T est un ensemble de transitions $(s, l, g, m, s') \in S \times L \times G \times M \times S$ où G est un ensemble de gardes (conditions de franchissement sur les variables et les horloges) et M l'ensemble des mises à jour des valuations des variables et des horloges [4] ; l, g and m sont optionnelles,
- $S_m \subseteq S$ est l'ensemble des localités marquées,
- $s_0 \in S$ est la localité initiale,
- $v_0 \in V$ est la valuation initiale des variables.

Les états de cet élément de procédé sont constitués d'un triplet $state_{\mathcal{A}} = (s, v, x)$, où :

- s est la localité active ;
- v est l'ensemble des valuations des variables ;
- x est l'ensemble des valuations des horloges.

4. Conventions de notation : les labels de synchronisation sont suivis de ! ou ?, les gardes sont entre crochets et les mises à jour de variable sont de la forme $v = valeur$

L'état (s, v, x) peut évoluer dans deux cas :
- suite à un écoulement du temps (des horloges) sous réserve que l'invariant $I(s)$ soit préservé,
- suite au franchissement d'une transition.

Une trace d'exécution $\gamma = (s_1, x_1, v_1) \xrightarrow{t_1} (s_2, x_2, v_2) \ ... \xrightarrow{t_n} (s_n, x_n, v_n)$ est une séquence de taille n alternant états et transitions avec $|\gamma| = n$.

$1 - 0.2$ Le modèle global de procédé est constitué d'un réseau de m automates temporisés représentant les éléments du procédé, $\mathcal{RA} = \mathcal{A}^1 \| \mathcal{A}^2 \| ... \| \mathcal{A}^m$. L'état global du réseau est un triplet $state_{\mathcal{RA}} = (rs, rv, rx) \in STATE_{RA}$ où :
- $rs \in RS = S^1 \times S^2 \times ... \times S^m$ est le produit cartésien des localités,
- $rv \in RV = V^1 \cup V^2 \cup ... \cup V^m$ est l'ensemble des valuations des variables,
- $rx \in RX = X^1 \cup X^2 \cup ... \cup X^m$ est l'ensemble des valuations des horloges.

Un état du réseau d'automate RA peut évoluer suite :
- à une évolution locale dans un des automates du réseau sous réserve que cette évolution ne soit pas due au franchissement d'une transition comportant un label de synchronisation,
- au franchissement simultanée de deux transitions t_p^α, t_q^β d'un couple d'automates (A^α, A^β) avec t_p^α contenant l'étiquette l_p^α ! et t_q^β contenant l'étiquette l_q^β ? telles que $l_p^\alpha = l_q^\beta$, les localités sources de ces transitions étant actives et leurs gardes étant satisfaites.

Une trace d'exécution sur un réseau d'automate $\Gamma = S_1 \xrightarrow{T_1} S_2 \ ... \xrightarrow{T_N} S_N$ est une séquence alternant états et transitions où S_i est un état du réseau d'automates $\in STATE_{RA}$ et T_i est soit une transition d'un automate du réseau, soit un couple de transitions (t_p^α, t_q^β) franchies simultanément dans le cas d'un canal de synchronisation.

Il convient de préciser que les différents types d'évolutions d'états (que ce soit au niveau d'un automate ou du réseau d'automates) ne sont affectés d'aucune priorité. Les évolutions d'un réseau RA sont donc fondamentalement non déterministes.

Les canaux de synchronisations entre les modèles de procédé sont de deux natures :
- les actions représentent la sollicitation d'un élément de procédé par un autre élément de procédé ; à titre d'exemple, si l'évolution entre deux états d'un modèle de fonction nécessite une modification d'état sur un modèle d'équipement, un label de synchronisation sera défini pour ces deux modèles, de type émission (noté !) pour le modèle de fonction, de type réception (noté ?) pour le modèle d'équipement ;
- les observations représentent une information (d'état, de disponibilité, ...) émise par un élément de procédé à destination d'un autre ; elles permettent donc de conditionner l'évolution d'un modèle (par exemple de fonction) à l'état d'un autre (par exemple d'équipement).

Cette définition des canaux de synchronisation définit donc implicitement une structuration hiérarchique des modèles de procédé. A titre d'exemple, les modèles d'équipements peuvent être sollicités par les modèles de fonctions, eux-mêmes pouvant être sollicités par les modèles de recettes et ainsi de suite. Pour les modèles situés au plus haut niveau de cette structure hiérarchique, s'il existe des labels de réception (l?), ils seront supposés émis (l!) par un modèle d'environnement du procédé, qui dans la pratique peut correspondre à un modèle de l'opérateur

humain ou d'un système automatisé de pilotage. Dans la section 2.1.2, nous présenterons ce modèle d'environnement comme étant la représentation d'un objectif de conduite.

Par ailleurs, la synchronisation entre éléments de procédé ne se traduit pas uniquement par la définition de canaux de synchronisation. En effet, les systèmes complexes critiques étudiés sont également caractérisés par diverses contraintes fonctionnelles ou de sécurité qui se traduiront par des gardes présentes sur les modèles de procédé :

- des **contraintes de sécurité** G_s : il s'agit de conditions sur l'état d'autres éléments du même niveau hiérarchique n, ainsi que sur des variables physiques, par exemple de conditions de précédence sur l'enclenchement d'éléments de même niveau ou encore de bonnes pratiques pour le pilotage du système ;
- des **contraintes de disponibilité** G_d : il s'agit de conditions portant sur la disponibilité (c'est-à-dire la possibilité à être manipulé) des éléments d'intérêt ;
- des **contraintes fonctionnelles** G_f : il s'agit, pour un élément de niveau n, de conditions sur l'état des éléments de niveau $n-1$ et/ou sur les valeurs des variables physiques.

Exemple :
Contraintes de sécurité G_s : une pompe ne doit pas être actionnée s'il n'existe pas un chemin pour mener d'une source de fluide à l'entrée de cette pompe.
Contraintes de disponibilité G_d : il arrive qu'un élément ne doivent pas être manipulé pour des raisons de sécurité (généralement lors d'opérations de maintenance), on dit alors qu'il est condamné. Dans ces conditions, cet élément n'est pas disponible. De la même manière, si un élément de niveau n dépend d'un élément indisponible de niveau $n-1$, alors l'élément de niveau n sera également considéré comme indisponible.
Contraintes fonctionnelles G_f : la configuration d'une fonction est dépendante de l'état des équipements qu'elle nécessite (vannes ouvertes/fermées par exemple).

2.1.1.2 Modélisation des séquences de conduite

Une procédure de conduite est une suite d'étapes spécifiant des actions et des observations à réaliser sur le procédé permettant de passer d'un état donné (considéré comme initial) du procédé à un état recherché (considéré comme un état final). Une telle procédure peut être formalisée sous la forme d'une trace d'exécution définie comme une séquence alternant états et transitions $S_0 \xrightarrow{t_1} S_1 \ ... \xrightarrow{t_n} S_n$.

Dans ce manuscrit, on s'intéresse en particulier à la génération de **séquences d'actions de conduite**. Les séquences d'actions sont semblables à des procédures dont les observations auraient été filtrées, c'est-à-dire que l'on s'occupe uniquement des actions à opérer sur le procédé. Les séquences d'actions sont construites par projection à partir des traces d'exécution générées, de manière à ne conserver que les valeurs d'horloges et les transitions franchies, pour fournir une suite de couples $(date, événement)$.

2.1.2 Génération de séquences par recherche d'atteignabilité

La génération de séquences repose sur la recherche d'une trace d'exécution conduisant d'un état donné à un état recherché. Les techniques de vérification formelle par model-checking [Clarke et al., 1999] permettent, grâce à des mécanismes automatiques de parcours d'espaces

d'état, de prouver qu'un modèle satisfait ou ne satisfait pas une propriété formelle, qui peut être une propriété d'atteignabilité. Dans le cas d'une propriété d'atteignabilité, les outils de model-checking fournissent généralement une trace d'exécution permettant d'atteindre la situation recherchée. Il est donc tout à fait naturel d'envisager la mise en œuvre de ce type de technique pour la génération de séquences. Pour cela, il est nécessaire de formaliser la propriété à vérifier. Dans notre cas, celle-ci sera modélisée au travers :

- d'un modèle observateur représentant notre objectif de conduite,
- d'une formule exprimée dans une logique temporelle, qui permet d'exprimer une propriété d'accessibilité de l'état cible du modèle observateur.

Le modèle observateur, que nous appellerons *séquenceur* dans la suite de ce mémoire, correspond au modèle d'environnement évoqué en section 2.1.1.1 et représente l'objectif de conduite à mettre en œuvre. Il sera donc spécifique pour chaque génération des séquences d'actions de conduite. En d'autres termes, ce modèle comportera une localité initiale qui sera supposée être la situation courante et une localité cible qui représente la situation de conduite à atteindre. Les labels de synchronisation présents dans le modèle *séquenceur* traduisent les états souhaités des modèles du procédé (généralement du modèle de plus haut niveau hiérarchique).

La propriété d'accessibilité de l'état cible du modèle *séquenceur* sera exprimée à l'aide d'une formule de logique temporelle. Il existe deux principales logiques temporelles : LTL (Linear Temporal Logic) [Pnueli, 1981] et CTL (Computation Tree Logic) [Clarke and Emerson, 1981]. CTL permettant de définir une propriété dans le cadre de l'exploration d'un futur décrit sous la forme d'un arbre, elle semble plus adaptée à notre problème. Elle propose des opérateurs de chemins (A pour *All* décrivant tous les chemins, E pour *Exists* décrivant au moins un chemin) et des opérateurs temporels (F pour *Finally* décrivant un état du futur, G pour *Globally* tous les états du futur). Dans le modèle *sequenceur*, si l'état recherché à partir de l'état initial est $s_{objectif}$, la propriété s'écrira sous la forme :

$$\| \texttt{EF sequenceur.s}_{objectif} \tag{2.1}$$

Si un chemin permettant d'atteindre l'état cible du modèle séquenceur existe, alors la propriété sera vérifiée et un outil de model-checking pourra fournir une trace d'exécution conduisant à cet état cible, à partir de laquelle il est possible de déduire une séquence de conduite. Il est important de noter que la trace générée est souvent le premier chemin, conduisant à la vérification de la propriété, rencontré lors de l'exploration de l'espace d'état. Elle est donc totalement dépendante de la façon dont le model-checker conduit son exploration (en largeur ou en profondeur par exemple). Certains model-checker offrent également la possibilité d'exhiber une trace optimale (en termes de longueur ou de durée des séquences) mais avec un inconvénient majeur lié au problème d'explosion combinatoire puisque cela nécessite d'explorer l'ensemble de l'espace d'état.

Un échec de preuve signifiera que l'objectif de conduite est inaccessible dans le contexte modélisé et qu'il n'existe donc pas de séquence permettant de l'atteindre.

Pour conclure cette section, la FIGURE 2.1 représente synthétiquement les principes de base proposés pour la génération de séquences d'actions de conduite.

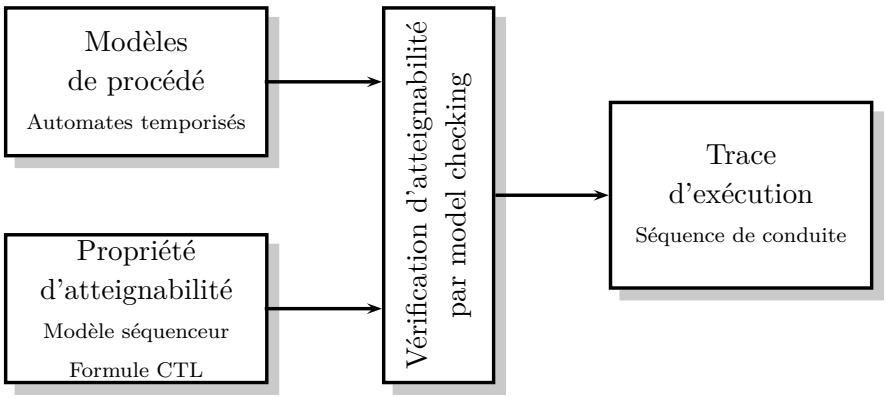


FIGURE 2.1 – Méthodologie de génération de traces d'exécution utilisée

Pour illustrer les principes généraux sur lesquels se base la génération de séquences, la section suivante présente une première approche « naïve » de modélisation et de génération basée sur un exemple simple. Les résultats obtenus sont discutés en fin de section pour justifier nos contributions en termes de systématisation de la démarche de modélisation et décomposition modulaire de la démarche de génération de séquences.

2.1.3 Illustration sur un cas d'étude "jouet"

2.1.3.1 Présentation du cas d'étude "jouet"

On considère un système hydraulique composé de vannes et de pompes (FIGURE 2.2). Le système peut exécuter trois fonctions F_1 à F_3 (respectivement refroidissement par la voie A, refroidissement par la voie B et remplissage d'une bâche alimentaire) par le biais de différentes configurations de trois vannes ($V1$, $V2$ et $V3$) et de trois pompes ($PO1$, $PO2$ et $PO3$). Ces équipements peuvent prendre deux états : *ouvert* et $fermé$ (notés O et F) pour les vannes et *arrêt* et *enclenché* (notés A et E) pour les pompes. Lorsque l'état d'un équipement est sans influence sur la configuration d'une fonction, il sera dit « indifférent », et sera noté « - ».

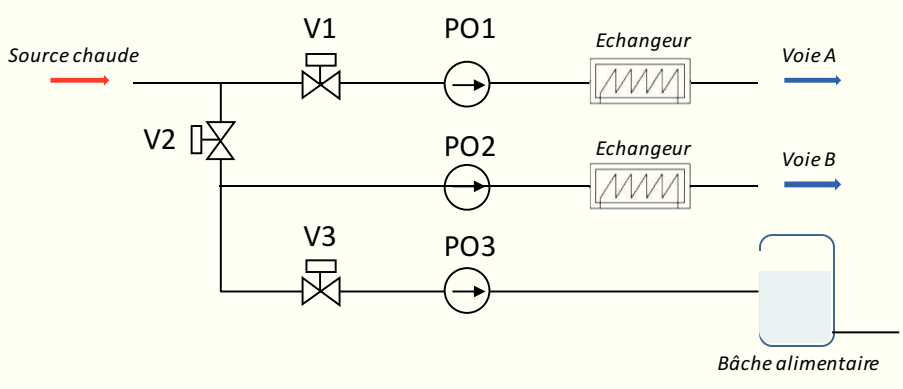


FIGURE 2.2 – Schéma de principe du cas d'étude "jouet"

La TABLE 2.1 définit les configurations des trois fonctions (F_1, F_2 ou F_3) en fonction de l'état attendu des équipements du système. Lorsque ces fonctions sont configurées, le service qu'elle supporte est supposé rendu. Ces contraintes de configurations sont des **contraintes fonctionnelles** G_f au sens défini en section 2.1.1.1. Par ailleurs, la TABLE 2.1 permet d'identifier des **contraintes d'exclusion** entre fonctions lorsque leurs configurations sont incompatibles. Par exemple, il est impossible d'exécuter simultanément les fonctions F_1 et F_2 car elles utilisent au moins un équipement en commun (par exemple $V2$) dans un état attendu différent pour chacune d'entre elles. En revanche, les fonctions F_2 et F_3 font appel à des configurations compatibles et sont donc exécutables simultanément.

Fonctions	$V1$	$V2$	$V3$	$PO1$	$PO2$	$PO3$	Refroidissement	Remplissage
F_1	O	F	-	E	-	-	✓(voie A)	✗
F_2	F	O	-	-	E	-	✓(voie B)	✗
F_3	-	O	O	-	-	E	✗	✓

TABLE 2.1 – Configurations des fonctions de l'exemple et effets sur α et β

Par ailleurs, on considère que pour des raisons de sécurité, afin d'éviter le phénomène de cavitation des pompes, les contraintes suivantes devront être respectées (il s'agit ici de **contraintes de synchronisation**) :

- $PO1$ ne peut pas être enclenchée si $V1$ n'est pas dans l'état *ouvert*,
- $PO2$ ne peut pas être enclenchée si $V2$ n'est pas dans l'état *ouvert*,
- $PO3$ ne peut pas être enclenchée si $V2$ et $V3$ ne sont pas dans l'état *ouvert*.

Enfin, on considère que le système est caractérisé par trois situations stabilisées :

- Situation stabilisée A (notée Sit_A) : le système est à l'arrêt,
- Situation stabilisée B (notée Sit_B) : le système est en pleine puissance, ce qui signifie que le refroidissement est effectué soit par la voie A soit par la voie B (le fluide en sortie des échangeurs est caractérisé par une température T),
- Situation stabilisée C (notée Sit_C) : lorsqu'il est en pleine puissance, le système peut également effectuer une mission de secours d'alimentation en eau d'un système hors du périmètre d'étude (cette mission se traduit par le remplissage d'une bâche caractérisée par une variable de niveau H).

2.1.3.2 Modélisation du cas d'étude "jouet"

La modélisation du procédé associé au cas d'étude simplifié fait apparaître une décomposition hiérarchique du système en trois niveaux : les modèles d'équipements associés aux vannes et aux pompes [5], les modèles de fonctions qui représentent leurs configurations et leurs effets sur les grandeurs physiques et enfin, le modèle de la recette qui représente les différentes situations stabilisées du système ainsi que les conditions de passage d'une situation à une autre. Ainsi, le modèle global du procédé sera composé de :

- six automates temporisés associés aux vannes et aux pompes,

5. Dans tous les exemples de ce manuscrit, les échangeurs, qui sont des équipements régis par des lois de fonctionnement dynamique complexes, sont considérés comme toujours prêts pour effectuer leurs fonctions

– trois automates temporisés associés aux fonctions F_1, F_2 et F_3,
– un automate temporisé représentant la recette.

Modèles des équipements

Les éléments de plus bas niveau hiérarchique du système sont des vannes et des pompes « tout ou rien », c'est à dire possédant deux états stables. Ces états sont représentés par les localités des modèles (par exemple *ouvert* / $fermé$ pour une vanne). Un changement d'état est provoqué par une sollicitation en provenance du niveau supérieur, formalisée sous la forme de canaux de synchronisation (événements *OuvrirVanne* et *FermerVanne* pour une vanne, événements *EnclencherPompe* et *ArreterPompe* pour une pompe). Ces changements d'état ne sont autorisés que si les conditions, notamment de sécurité, sont réunies. Elles sont modélisées dans les gardes *Conditions_ouverture* et *Conditions_fermeture* pour les vannes et *Conditions_enclenchement* et *Conditions_arret* pour les pompes. Ces gardes peuvent faire intervenir les états d'autres équipements, en fonction de la topologie de l'installation (par exemple l'interdiction d'enclencher une pompe si les vannes amont ne sont pas ouvertes). Pour cela, l'état des équipements sera également consigné dans des variables booléennes *Etat_vanne* et *Etat_pompe*. Enfin, la disponibilité de l'équipement est une condition nécessaire pour autoriser un changement d'état. Elle est représentée par une variable booléenne *dispo*. Une indisponibilité peut être due à une défaillance ou à une action délibérée de condamnation de l'équipement pour des raisons de maintenance. Ces aspects étant hors du périmètre de notre étude, la valeur de cette variable booléenne sera définie à l'initialisation du modèle et ne pourra pas évoluer. Elle permet de déterminer l'atteignabilité d'une situation en tenant compte de la disponibilité effective des équipements.

La Figure 2.3 présente les modèles de la vanne V1 et de la pompe PO1 (les autres équipements sont modélisés de manière identique).

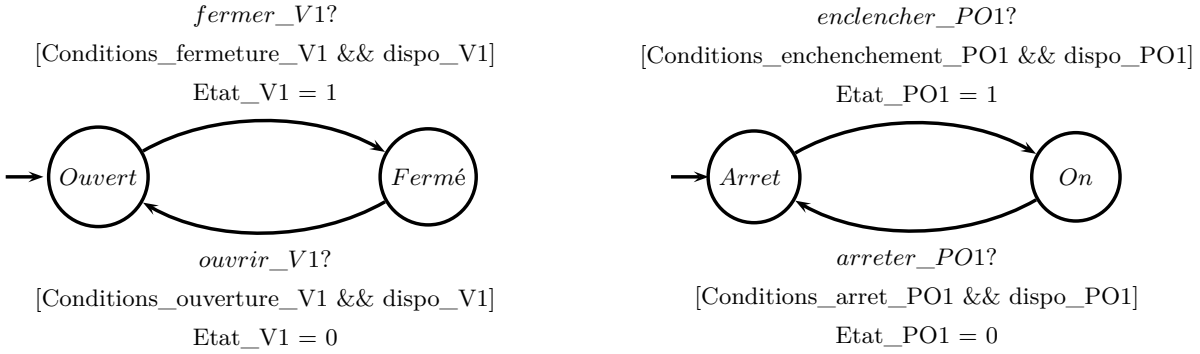


Figure 2.3 – Modèles des équipements V1 et PO1

Modèles des fonctions

Les fonctions peuvent être caractérisées par deux états principaux :
– en attente : la fonction ne s'exécute pas et est en attente d'une sollicitation en provenance du niveau supérieur ;
– en exécution : la fonction s'exécute et les variables physiques associées évoluent.

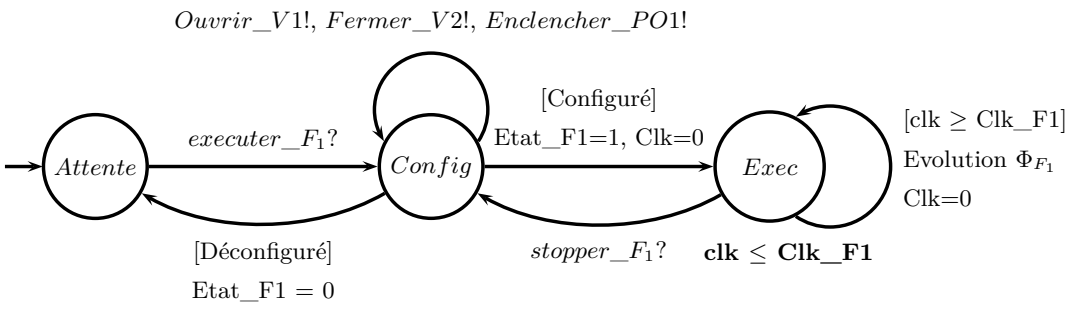


FIGURE 2.4 – Modèle de la fonction F_1

Le passage d'un état d'attente à un état d'exécution nécessite la configuration (ou le lignage) des équipements impliqués dans la réalisation de la fonction. A titre d'exemple, l'exécution de la fonction F_1 nécessite que la vanne V1 soit dans l'état *Ouvert*, la vanne $V2$ dans l'état $Fermé$ et la pompe PO1 dans l'état *Enclenché*. Concrètement, au niveau du modèle des fonctions, cela se traduit par un modèle structuré autour d'états stables (*Attente* et *Exec*) et d'un état transitoire $Config$. La sortie de cet état transitoire est conditionnée par les états des équipements auxquels fait appel la fonction ; cette condition est matérialisée par les gardes [$Configuré$] et [$Déconfiguré$] (notons que pour que cette dernière garde soit validée, il faut que tous les équipements ne soient pas dans l'état attendu pour l'exécution de la fonction). Dans l'état $Configuration$, les fonctions ont donc la possibilité d'émettre des requêtes à destination des équipements jusqu'à ce que les états de ces derniers soient ceux attendus.

Une fois configurée, la fonction s'exécute : les variables physiques évoluent (incrémentation ou décrémentation des variables Φ_{F_1}), a un rythme déterminé par la valeur d'une horloge associée à la fonction (Clk_F_1).

A titre d'exemple, la FIGURE 2.4 présente le modèle de la fonction F_1 [6].

Modèles de recette

Le modèle de recette décrit les enchaînements entre les différentes situations stabilisées du système Sit_A (système à l'arrêt), Sit_B (système en pleine puissance) et Sit_C (système en puissance et mode secours). Il sera donc caractérisé par trois localités associées aux situations stabilisées et par des localités intermédiaires correspondant aux phases transitoires de passage d'une situation stabilisée à l'autre. Le passage d'une situation stabilisée à une autre sera provoqué par des demandes externes au procédé représentées sous la forme de trois canaux de synchronisation : *puissance*?, *secours*?, *arret*?.

Les évolutions entre situations stabilisées seront par ailleurs conditionnées par les valeurs physiques (dans notre cas T et H) avant évolution (on peut imaginer que le refroidissement ne peut être enclenché que lorsque la température dépasse un certain seuil) et après évolution (c'est à dire les valeurs attendues dans la situation cible). Ces conditions sont formalisées à l'aide de gardes associées aux transitions

6. La notation $a!, b!$ associée à une transition a la même signification que celle utilisée pour les événements dans les automates à états finis : elle représente deux transitions ayant les mêmes états source et cible et associées aux canaux $a!$ et $b!$ (dans le cas présent, trois transitions bouclées sur l'état $Config$ associées respectivement aux canaux $Ouvrir\_V1!$, $Fermer\_V2!$, $Enclencher\_PO1!$

Pour atteindre les valeurs des grandeurs physiques requises par la situation stabilisée cible, il sera nécessaire de faire appel aux différentes fonctions du système via les canaux de synchronisation (*exécuter_F$_i$*!, *stopper_F$_i$*!). Ces appels sont bien sûr conditionnés par l'ensemble des contraintes d'exclusion entre fonctions et sont représentés sous la forme de gardes.

La FIGURE 2.5 présente un exemple de ce que pourrait être un modèle de recette relatif au cas d'étude "jouet". Cet exemple est bien sûr proposé à titre d'illustration de la démarche dans la mesure où plusieurs solutions de modélisation sont envisageables à ce stade. Une discussion à ce sujet est proposée en fin de section 2.1.3.

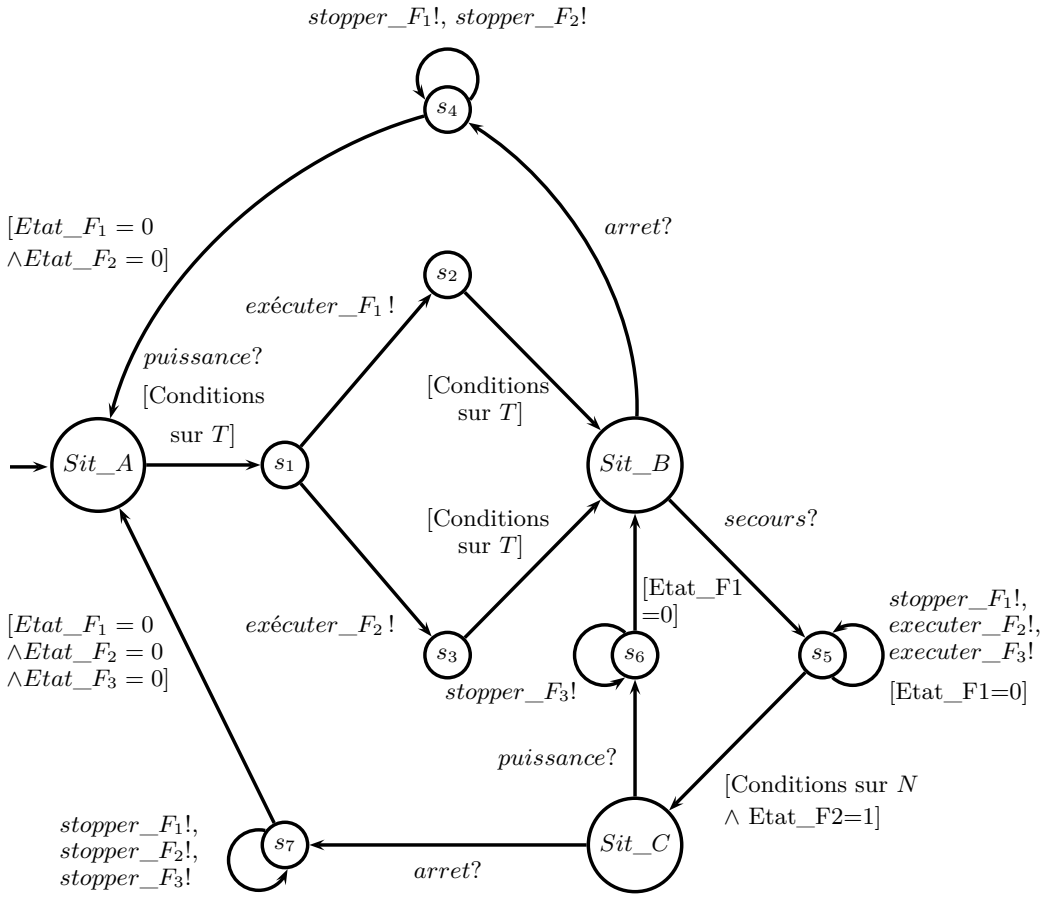


FIGURE 2.5 – Modèle de la recette

2.1.3.3 Génération de séquences

Pour procéder à la génération de séquences, la première étape consiste à modéliser l'objectif de conduite que l'on cherche à atteindre. Comme indiqué à la section 2.1.2, cette modélisation est effectuée au travers d'un automate observateur, appelé *sequenceur*. Son rôle est de solliciter le modèle du plus haut niveau hiérarchique (pour notre cas d'étude, il s'agit du modèle de recette) en fonction de la situation stabilisée que l'on cherche à atteindre. La FIGURE 2.6 présente un exemple dans lequel on cherche à atteindre la situation *Sit_B* (puissance) depuis la situation initiale *Sit_A* (arrêt).

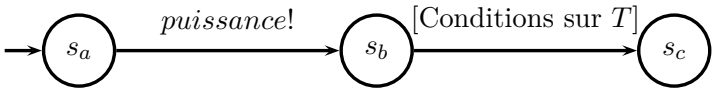


FIGURE 2.6 – Modèle du séquenceur

Afin de déterminer s'il existe une possibilité d'atteindre cet état cible, il faut vérifier la propriété suivante : « il existe un chemin tel que l'automate *sequenceur* atteigne la localité s_c ». Elle est exprimée formellement en CTL par l'EQUATION 2.2 :

$$\| \texttt{EF sequenceur.s}_c \tag{2.2}$$

La vérification de la propriété d'atteignabilité (EQUATION 2.2) sur les modèles présentés ci-dessus peut s'effectuer à l'aide d'un outil de modélisation et de model checking. Cette vérification permet, si la propriété est vérifiée, d'obtenir une trace d'exécution comprenant l'ensemble des transitions franchies par les modèles de procédé ainsi que les valeurs des variables physiques et d'horloges. En ne considérant dans cette trace que les transitions associées à un canal de synchronisation, il est possible d'extraire une séquence d'actions de conduite permettant d'atteindre la situation Sit_B. Ainsi, on peut, par exemple, obtenir la séquence suivante :

clk = 0
- puissance
  - executer_Fct2
    - ouvrir_V2
    - fermer_V1
    - fermer_V2
    - ouvrir_V2
    - enclencher_PO1

Cette séquence montre que pour atteindre l'état Sc, il est possible à t = 0 de demander la configuration de la fonction f_2, et que cela s'effectue par des changement d'états sur les équipements $V1$, $V2$ et $PO1$. Une fois la fonction F_2 configurée, la variable physique T évolue jusqu'à atteindre une plage de valeurs compatible avec la situation Sit_B qui est alors atteinte. On remarque ici que deux changements d'états successifs et antagonistes sont demandés sur l'équipement $V2$. Ceci est dû au fait que l'espace d'état est parcouru de manière aléatoire, sans recherche d'optimum, et qu'il est donc possible de générer des séquences pour lesquelles des actions antinomiques sont proposées. Ce problème sera abordé dans la dernière partie de ce chapitre.

2.1.3.4 Discussion

Au delà de leur caractère illustratif, la modélisation et la génération de séquences du cas d'étude "jouet" soulèvent plusieurs questions relatives :
- aux choix de modélisation réalisés spécifiquement pour ce cas d'étude,
- à la capacité à générer dans un délai raisonnable des séquences pour des systèmes de grandes tailles.

Conclusion

L'approche de modélisation présentée pour le cas d'étude "jouet" met en évidence deux voies d'amélioration :

1. accroître la généricité de la modélisation pour faciliter sa mise en œuvre dans un processus d'ingénierie, notamment en systématisant la démarche de construction des modèles et en facilitant leur retrait ou ajout,
2. relâcher les contraintes induites par les modèles (en termes de choix de trajectoires de conduite) afin d'élargir la recherche de séquences.

La SECTION 2.2 apporte des éléments de réponse à ces deux objectifs par une formalisation de la démarche de modélisation au travers de patrons de modélisation génériques et instanciables pour un usage à plus grande échelle.

2.2 Formalisation de la démarche de modélisation

Conformément aux conclusions de la section précédente, cette section présente une formalisation de la démarche de modélisation au travers de la définition de mécanismes génériques pour la synchronisation entre modèles de procédé et par la proposition de patrons de modélisation instanciables.

2.2.1 Définition des mécanismes de synchronisation

Nous considérons qu'un procédé structuré en N niveaux hiérarchiques peut être décrit par des ensembles de modèles notés M_i associés à chaque niveau hiérarchique i. L'ensemble de modèles de plus haut niveau (le modèle de recette dans l'exemple de la section précédente) sera noté M_N tandis que l'ensemble de modèles de plus bas niveau (les modèles d'équipements dans l'exemple de la section précédente) sera noté M_1. A un niveau hiérarchique i, chaque élément de procédé k appartenant à ce niveau est décrit par un modèle $M_i^k \in M_i$.

Par ailleurs, nous considérons, dans un premier temps, que le modèle de séquenceur, qui représente l'objectif de conduite à atteindre sous la forme d'un automate observateur (destiné à la vérification de la propriété d'atteignabilité), est unique.

La synchronisation entre ces différents modèles (procédé et séquenceur) joue différents rôles :

- sollicitation d'éléments de procédé de niveau i par un élément de procédé de niveau $i+1$ (une fonction sollicite des équipements, une recette sollicite des fonctions, ...),
- conditions d'évolution d'un élément de procédé de niveau i dépendantes des états des éléments de procédé de niveau inférieur $i-1$ ou de même niveau i,
- synchronisation entre le séquenceur et le modèle de procédé de plus haut niveau N.

Selon les cas de figures, ces synchronisations sont supportées par des canaux de synchronisation et/ou des variables globales d'état.

2.2.1.1 Synchronisations entre éléments de procédé

Dans l'exemple simplifié de la section précédente, les éléments de procédé étaient sollicités via des canaux de synchronisation associés à un et un seul changement d'état spécifique de

l'élément de procédé sollicité : $ouvrir\_V_i$, $fermer\_V_i$, $exceuter\_F_i$, $stopper\_F_i$ par exemple. Si cette représentation paraît naturelle, elle impose, à un certain niveau, d'expliciter les actions de conduite à réaliser sur les éléments de niveau inférieur. La conséquence est que la génération de séquences sera limitée aux seules trajectoires de conduite préalablement identifiées dans les modèles.

Une manière de contourner cette limite consiste à considérer que si une situation de conduite cible ne peut être atteinte dans l'état courant de l'ensemble des modèles de procédé, cela signifie nécessairement que des changements d'état doivent être opérés sur un ou plusieurs éléments de procédé. Plus précisément pour un modèle de niveau $i \in M_i$, cela signifie que des changements d'états doivent être opérés sur un ou plusieurs modèles $M_{i-1}^p \in M_{i-1}$.

Les sollicitations des éléments de procédé sont donc représentées par un canal de synchronisation **unique** Req_i^p partagé entre un des modèles $M_{i+1}^k \in M_{i+1}$ et un modèle $M_i^p \in M_i$ avec $i \in \{1, ..., N\}$:

$$\left\| \begin{array}{l} Req_i^p \; ! \in L_e(M_{i+1}^k) \\ Req_i^p \; ? \in L_r(M_i^p) \end{array} \right. \tag{2.3}$$

La signification des canaux Req_i^p est la suivante : un élément de niveau $i+1$ sollicite un changement d'état sur l'élément p de niveau i. Par hypothèse, les modèles de plus bas niveau M_1^p ne peuvent solliciter aucune ressource, aucun label $Req_1^p!$ n'est donc défini. Par ailleurs, la notation du canal de synchronisation ne fait pas apparaître l'indice k du modèle émetteur dans la mesure où un même élément peut recevoir une sollicitation identique (modélisée par le même canal de synchronisation) en provenance de différents éléments de niveau supérieur.

En réponse à une sollicitation de changement d'état en provenance d'un modèle de niveau $i+1$, le modèle M_i^p sollicité émettra un message de compte-rendu lorsque le changement d'état aura été effectué. Ce message sera modélisé par un canal de synchronisation Rep_i^p signifiant qu'un élément p de niveau i informe les modèles de niveau $i+1$ qu'un changement d'état a été effectué avec :

$$\left\| \begin{array}{l} Rep_i^p \; ! \in L_e(M_i^p) \\ Rep_i^p \; ? \in L_r(M_{i+1}^k) \end{array} \right. \tag{2.4}$$

La synchronisation entre les différents modèles de procédé s'effectue donc via des canaux de synchronisation Req_i^p (requêtes) et Rep_i^p (réponses) comme le montre la FIGURE 2.7. L'orientation des arcs représente une synchronisation du label émetteur (!) vers le label récepteur (?) avec les notations suivantes (EQUATION 2.5) où N est le nombre de niveaux et m_i le nombre d'éléments du niveau i.

$$\left\| \begin{array}{l} Req_i = \{Req_i^p\}_{1 \leq p \leq m_i} \\ Rep_i = \{Rep_i^p\}_{1 \leq p \leq m_i} \end{array} \right. \tag{2.5}$$

De manière identique aux canaux de synchronisation Req_i utilisés pour les sollicitations de ressources, et pour les mêmes raisons, les canaux de synchronisation Rep_i ne sont porteurs d'aucune information relative aux états atteints suite à une sollicitation. Afin de pouvoir considérer ces derniers dans l'évaluation des conditions d'évolution d'un modèle tierce

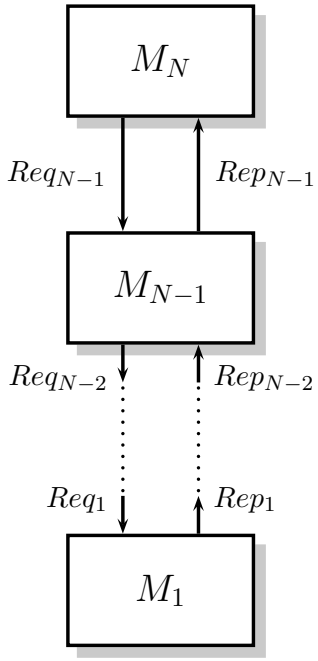


FIGURE 2.7 – Architecture hiérarchique du modèle du procédé à conduire

(notamment pour le traitement de contraintes fonctionnelles, de synchronisation, d'exclusion ou encore de disponibilité) et conformément à la définition d'un état dans le formalisme des automates temporisés, les modèles comportent un ensemble de variables :

- observant certaines localités actives des modèles (les états ou situations stables), notées $Obs\_M_i^k$; ces variables peuvent être booléennes, dans le cas où le modèle ne comporte que deux localités d'intérêt, ou entières s'il en comporte plusieurs ; ces variables sont mises à jour lorsque les localités d'intérêt sont atteintes ;
- caractérisant les grandeurs physiques Φ sur lesquelles le modèle considéré a une action ; ces variables sont entières et sont mises à jour lors de certaines évolutions du modèle.

Enfin, la disponibilité des équipements est modélisée par une variable $Disp\_M_i^k$ associée à chaque élément de procédé. Ces variables seront affectées en début de recherche de séquences d'actions de conduite pour tenir compte de l'état courant du procédé à partir duquel on souhaite effectuer la recherche :

- en ingénierie pour évaluer la capacité d'atteindre une situation malgré l'indisponibilité de certains éléments (cette indisponibilité « testée » est donc connue avant le lancement de la recherche de séquences)
- en exploitation pour trouver une solution admissible compte tenu de l'indisponibilité connue de certains éléments.

Par ailleurs, cette approche statique de la disponibilité est justifiée par le fait qu'un traitement dynamique impliquerait une représentation probabiliste des défaillances portée par des transitions conduisant à une localité explicite d'indisponibilité. Cet aspect, orienté sûreté de fonctionnement et robustesse de la séquence d'actions vis à vis des aléas, est hors de notre périmètre d'étude.

2.2.1.2 Synchronisation entre modèles de procédé et séquenceur

D'après les définitions précédentes, un modèle de procédé M_N^k (c'est à dire un modèle de plus haut niveau hiérarchique) peut contenir :

- des canaux de synchronisation émetteur $Req_{N-1}^p! \in Req_{N-1}$, permettant de solliciter les ressources p, et des canaux de synchronisation récepteur $Req_{N-1}^p? \in Rep_{N-1}$ accusant réception de la réponse des ressources,
- un canal récepteur $Req_N^k? \in Req_N$ correspondant à une requête de changement d'état du modèle M_N^k et un canal émetteur $Rep_N^k! \in Rep_N$ correspondant au compte-rendu d'exécution de la requête.

Ces deux derniers canaux $Req_N^k?$ et $Rep_N^k!$ doivent impérativement avoir leur correspondant (respectivement $Req_N^k!$ et $Rep_N^k?$) dans un des modèles du système afin de garantir une propriété de non blocage du modèle M_N^k. En effet, dans le cas contraire, les localités « source » des transitions portant ces canaux deviendraient inévitablement des localités bloquantes (*deadlock*).

D'un point de vue intuitif, ces deux canaux correspondent respectivement à des requêtes formulées par l'environnement du procédé (opérateur de conduite, système de pilotage, ...) et à leur compte-rendu d'exécution. Dans notre cas, nous ne disposons pas d'un tel modèle d'environnement puisque ce sont précisément les trajectoires de conduite que celui-ci pourrait produire que nous cherchons à identifier. Comme indiqué précédemment, le modèle d'environnement est donc remplacé par un automate observateur, nommé *sequenceur*, qui formalise les objectifs de conduite que l'on cherche à satisfaire.

L'automate *sequenceur* devra reconnaître une séquence alternant demandes de changement d'état des modèles de procédé $Req_N^k!$ et comptes-rendus d'exécution $Rep_N^k!$ jusqu'à atteindre l'état du procédé recherché. Cette condition s'exprime à partir des variables d'observation des localités, et des grandeurs physiques. Un exemple d'automate séquenceur respectant ces contraintes est présenté par la FIGURE 2.8, où la localité s_c représente la situation cible recherchée.

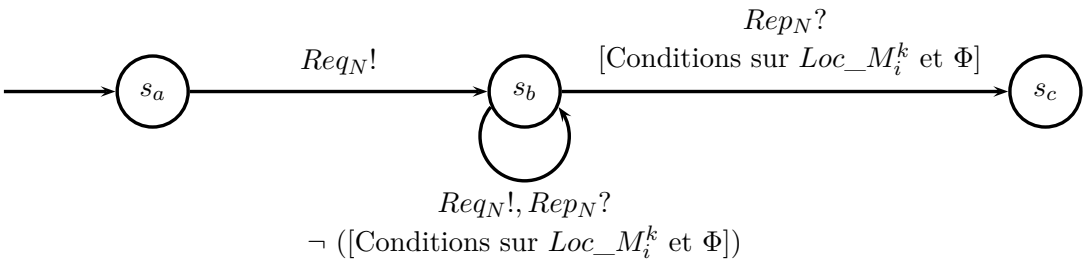


FIGURE 2.8 – Structure générique de modèle du séquenceur

Le principe suivi peut être mis en parallèle avec la théorie de la commande supervisée de [Ramadge and Wonham, 1987] et plus particulièrement des modèles de spécification. En effet, dans le cadre de cette théorie, la composition entre les modèles de procédé et les modèles de spécification permet, après élimination des états défendus et faiblement défendus, d'obtenir un modèle du procédé sous contrôle. Dans notre cas, l'algorithme de synthèse est remplacé par une recherche d'atteignabilité sur le réseau d'automates composé des modèles de procédé et du séquenceur comme le montre la FIGURE 2.9.

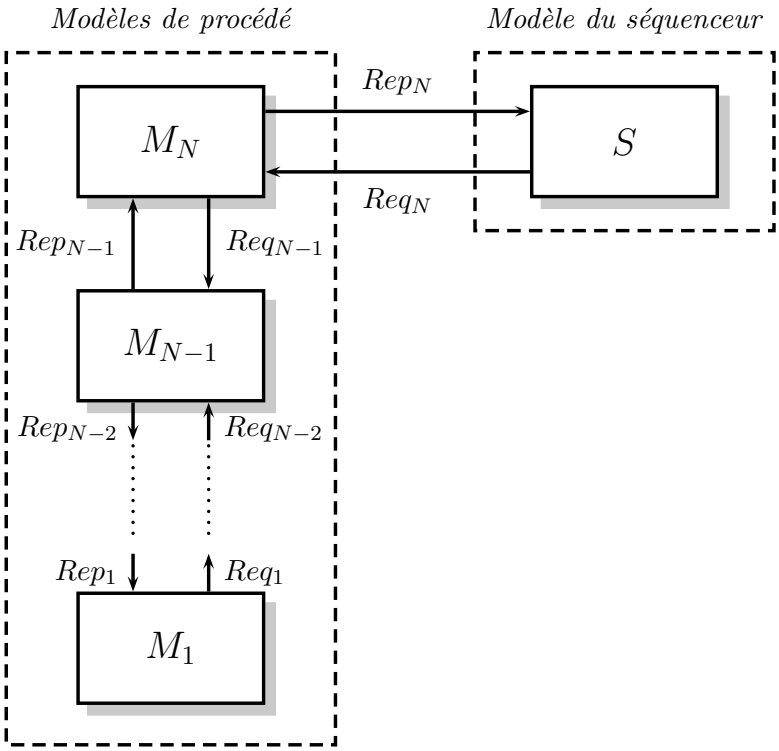


FIGURE 2.9 – Interactions entre modèles du procédé et séquenceur

Exemple :
Sur la base d'une décomposition hiérarchique à trois niveaux (recette, fonctions, équipements en accord avec la norme ISA88 [ISA, 1998]), l'application de notre démarche sur l'exemple présenté en SECTION *2.1.3 aboutit à :*

***Modèles** :*
- $M_3 = \{M_3^k\}$ *avec* $k = 1$ *pour la recette (modèle unique) ;*
- $M_2 = \{M_2^k\}$ *avec* $k = [1...3]$ *pour les fonctions du système (trois modèles pour les fonctions* F_1 *à* F_3*) ;*
- $M_1 = \{M_1^k\}$ *avec* $k = [1...6]$ *pour les équipements du système (six modèles) ;*
- S *: modèle de séquenceur.*

Canaux de synchronisation :
- Req_3^k *et* Rep_3^k *avec* $k = 1$ *pour la sollicitation et la réponse de la recette ;*
- Req_2^k *et* Rep_2^k *avec* $k = [1...3]$ *pour les sollicitations/réponses des fonctions ;*
- Req_1^k *et* Rep_1^k *avec* $k = [1...6]$ *pour les sollicitations/réponses des équipements.*

Variables
- *physiques (débits en sortie des vannes et des pompes mis à jour par les modèles d'équipements, température et niveau mis à jour par les modèles de fonctions) ;*
- *d'observation de certaines localités actives d'intérêt des modèles d'équipements, de fonctions et de recette.*

2.2.2 Proposition de patrons pour la modélisation du procédé

Les mécanismes de synchronisation étant modélisés de manière similaire quel que soit le niveau hiérarchique des modèles de procédé, il est tout à fait envisageable de proposer des guides génériques pour faciliter et systématiser leur construction.

Ces guides seront considérés comme des patrons de modélisation, au sens défini par [CENELEC, 1995] ou encore comme [Alexander et al., 1977] :

> *- a textual or graphical artefact devised to represent in an orderly way the diverse information on common properties and elements of a collection of phenomena [CENELEC, 1995]*
> *- un moyen de résoudre un problème qui se manifeste constamment dans notre environnement, et qui décrit le cœur de la solution de ce problème, d'une façon telle que l'on puisse réutiliser cette solution des millions de fois sans jamais le faire deux fois de la même manière [Alexander et al., 1977]*

Dans notre cas, les patrons de modélisation sont des structures élémentaires, modélisées sous la forme d'automates, que l'on retrouvera, plus ou moins à l'identique, dans l'ensemble des modèles de procédé. L'intérêt principal réside :

- dans le gain de temps obtenu pour construire les modèles d'un procédé ou pour prendre en compte des évolutions mineures de celui-ci puisque l'élaboration des modèles est faite au travers de la spécialisation, voire de l'instanciation des patrons ;
- dans l'homogénéisation des modèles qui facilite leur compréhension, leur analyse, voire leur vérification.

2.2.2.1 Principes généraux

Les patrons de modélisation proposés reposent sur des structures génériques décrivant les évolutions entre deux localités stables, dont la définition est donnée ci-dessous, et un ensemble de localités transitoires entre les deux localités stables.

Définition 1 *Une localité stable appartenant à un modèle M_i^k est une localité dont la désactivation requiert la réception d'une requête de changement d'état :*

- *toutes les transitions en sortie de la localité devront être au moins étiquetées par un label de synchronisation Req_i^k ? $\in L_r$.*
- *par symétrie, toutes les transitions en entrée de la localité devront être étiquetées par des labels de synchronisation Req_i^k ! $\in L_e$.*

L'évolution entre deux localités stables est représentée par une structure générique qui s'inspire de la séquence d'actions définie par [Vogel, 1988] autour de trois actèmes « *Préparer* », « *Exécuter* » et « *Clôturer* ». La phase de préparation est constituée des activités nécessaires à établir l'ensemble des conditions nécessaires à l'exécution de l'action, la phase d'exécution concerne les transformations principales associées à l'action tandis que la phase de clôture consiste à libérer les ressources utilisées et à les restituer dans l'environnement de l'action. Appliquée dans le domaine des systèmes manufacturiers de production, cette séquence permet

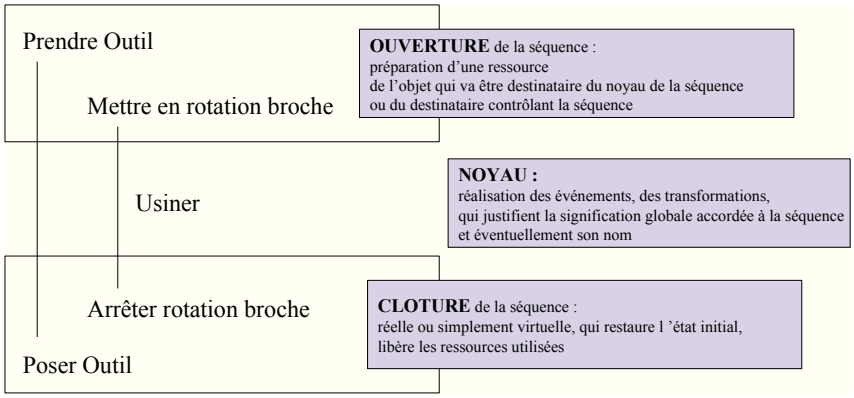


FIGURE 2.10 – Exemple d'actions antinomiques [Morel, 1992]

de « représenter et de structurer les actions de commande à réaliser sous forme de séquences logiques réutilisables »[Morel, 1992] sous la forme d'actions antinomiques (FIGURE 2.10).

L'application de ces principes à la modélisation du procédé aboutit à la structure générique, donnée par la FIGURE 2.11, pour décrire l'évolution entre deux localités stables st_a et st_b.

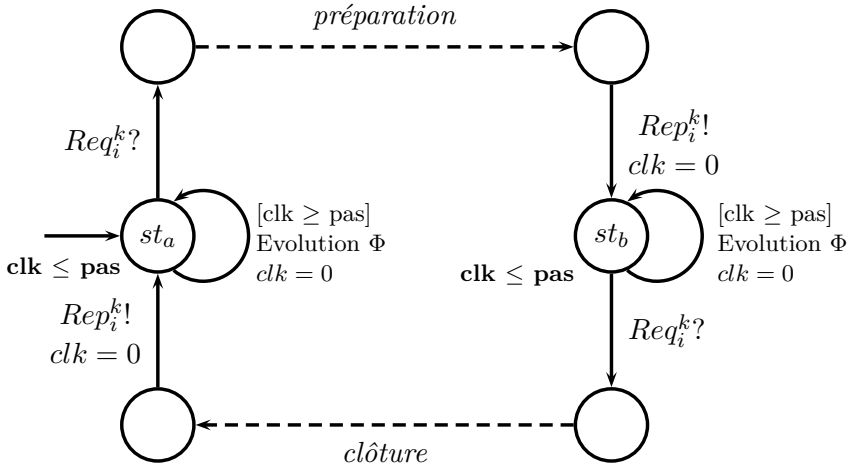


FIGURE 2.11 – Ébauche de structure entre deux localités stables

Dans cette structure, chaque *localité stable* pourra être le siège d'une variation des grandeurs physiques correspondant à la phase « *Exécuter* ». Ces variations sont modélisées grâce à une ou plusieurs transitions bouclées sur une localité stable. Le pas de temps avec lequel évoluent les grandeurs physiques est donné par le paramètre *pas* qui conditionne le franchissement de la transition bouclée. En vertu des principes sous-jacents aux patrons de modélisation, les boucles sur les *localités stables* (st_a et st_b) ne sont pas obligatoires et pourront être adaptées selon l'élément de procédé modélisé.

Les phases de préparation et de clôture ont, quant à elles, deux missions :
– sollicitation des éléments de procédé de niveau inférieur (émission de requêtes de changement d'état) afin de les mettre en conformité vis à vis des objectifs assignés à

la réalisation des transformations portées par la *localité stable* cible ;
- vérification des conditions de changement d'état (contraintes fonctionnelles, de sécurité et de disponibilité) avant ou après exécution des actions de préparation, selon les cas.

Dans la mesure où les modèles de procédé M_1 ne peuvent solliciter aucune ressource, la structure générique de changement d'état entre deux localités stables des éléments de niveau 1 sera différente et beaucoup plus simple que celle des modèles de niveau supérieur. Les deux sections suivantes présentent cette structure pour des modèles de niveau 1 et pour des modèles de niveau n avec $n > 1$.

2.2.2.2 Structure générique pour les modèles M_1

Pour les modèles de procédé de plus bas niveau M_1 (des équipements de type vanne, pompe, capteurs, ...), la phase de préparation (respectivement de clôture) ne comporte aucune action spécifique puisque ces éléments ne peuvent solliciter aucune ressource.

En termes de vérification, les contraintes fonctionnelles sont sans objet dans la mesure où aucune action de préparation (clôture) n'est effectuée (si elles existent, elles sont donc invariantes entre les différentes localités stables). Les contraintes de sécurité dépendent, quant à elles, du contexte dans lequel l'élément est sollicité (et notamment des autres éléments sollicités conjointement), elles seront donc gérées par les niveaux supérieurs. La seule contrainte présente dans les modèles de plus bas niveau est donc la contrainte de disponibilité : la requête de sollicitation d'un élément M_i^k ne sera acceptée que si cet élément est disponible. Elle se traduit par une garde $G_d = Disp\_M_i^k$, où $Disp\_M_i^k$ est une variable booléenne représentant la disponibilité de l'élément M_i^k, associée aux transitions portant les labels $Req\_M_i^k$?.

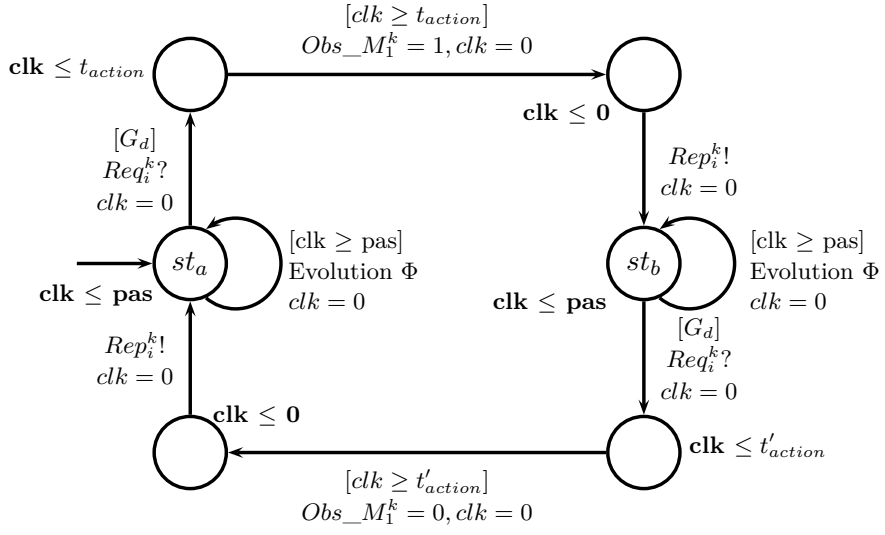


FIGURE 2.12 – Structure entre deux localités stables pour un modèle M_1

La structure générique se limite donc à celle présentée en (FIGURE 2.12). Elle est obtenue en remplaçant la transition en pointillé de la FIGURE 2.11 par une transition dans laquelle :
- la variable d'observation $Obs\_M_1^k$ est mise à jour (une valeur de 1 peut signifier un état

ouvert d'une vanne tandis qu'une valeur de 0 peut représenter un état fermé) après une durée t_{action} correspondant au temps de transfert d'une localité à l'autre (par exemple temps d'ouverture ou de fermeture d'une vanne) ;
– la garde G_d représente la vérification de la contrainte de disponibilité.

Exemple :
Sur le cas d'étude "jouet" de la SECTION *2.1.3, le modèle M_1^1 de la vanne $V1$ (vanne tout ou rien possédant deux états ouvert et fermé) est donné par la* FIGURE *2.13.*

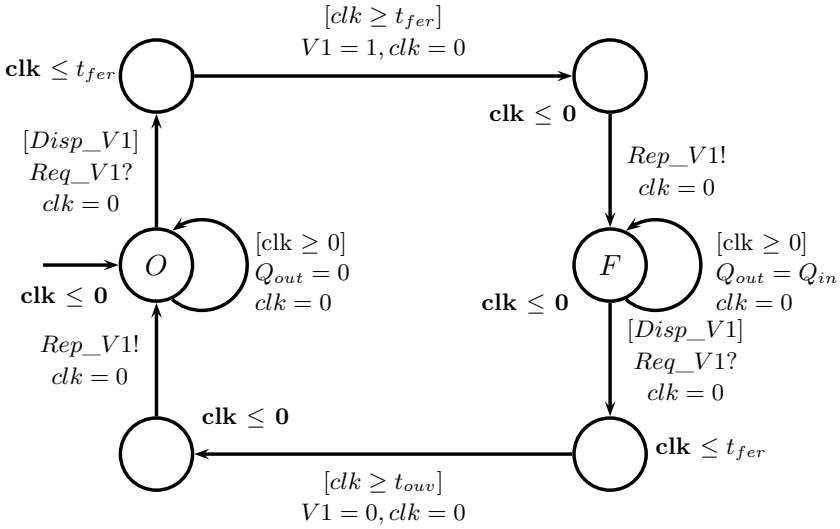


FIGURE 2.13 – Modèle M_1^1 de la vanne $V1$

avec les notations suivantes :
– *O et F sont les localités stables représentant les états ouvert et fermé de $V1$,*
– *Q_{out} et Q_{in} sont les débits en sortie et en entrée de V1,*
– *$t_{action} = t_{fer}$ et $t'_{action} = t_{ouv}$ sont les temps de fermeture et d'ouverture de V1,*
– *$Obs\_M_1^1 = V1$ est une variable représentant l'état de $V1$ (égale à un si la vanne est fermée, égale à zéro si la vanne est ouverte),*
– *$Disp\_M_1^1 = Disp\_V1$ représente la disponibilité de $V1$,*
– *$Req_1^1 = Req\_V1$, $Rep_1^1 = Rep\_V1$ sont la sollicitation et le compte-rendu de $V1$.*

2.2.2.3 Structure générique pour les modèles M_n

Pour les modèles de niveau M_n avec $n > 1$, les phases de préparation et de clôture permettent l'évolution entre deux localités stables de ces modèles.

Comme pour les modèles de niveau 1, l'évolution entre deux localités stables répond à une sollicitation $Req_i^k?$ émise par un modèle de niveau supérieur. Cette sollicitation ne peut être acceptée que si l'ensemble des ressources mobilisées par le modèle M_i^k, pour évoluer d'une situation stable à l'autre, est disponible. Cette contrainte de disponibilité est formalisée par une garde booléenne $G_d^{(s_a,s_b)}$ définie pour un couple de localités stables (s_a, s_b) par :

$$\left\| \; G_d^{(s_a,s_b)} = f_d^{(s_a,s_b)}(Disp\_M_{i-1}^p) \right. \tag{2.6}$$

Par ailleurs, une *localité stable* d'un modèle M_i^k est caractérisée par une configuration de ses ressources, c'est à dire des localités et/ou des grandeurs physiques portées par les modèles M_{i-1}^p de niveau inférieur. A titre d'exemple, la configuration nécessaire à l'exécution d'une fonction impose des états particuliers sur un sous-ensemble d'équipements, une recette sollicite l'exécution d'un sous-ensemble de fonctions pour évoluer d'un état de procédé à un autre, ... La conformité d'une configuration est représentée par une contrainte fonctionnelle, associée à chaque localité stable lst, formalisée par une garde booléenne G_f^{lst} définie par :

$$\left\| G_f^{lst} = f_p^{lst}(Obs\_M_{i-1}^p, \Phi) \right. \tag{2.7}$$

Enfin, lorsque la contrainte fonctionnelle n'est pas vérifiée, cela signifie que des demandes de changement d'état des éléments de niveau inférieur doivent être formulées jusqu'à ce qu'elle le soit. Ainsi pour un modèle M_i^k, la phase de préparation (respectivement de clôture) intègre les sollicitations de diverses ressources sous la forme de requêtes (Req_{i-1}^p) jusqu'à ce que la configuration des éléments de procédé de niveau inférieur M_{i-1}^p soit conforme au requis permettant atteindre la *localité stable* cible du modèle M_i^k. Néanmoins, la sollicitation de ces éléments doit obéir à certaines règles de sécurité telles que des contraintes de précédence ou d'exclusion (à titre d'exemple, l'enclenchement d'une pompe ne peut être requis que si les vannes amont sont ouvertes, les fonctions F_1 et F_2 du cas d'étude "jouet" ne peuvent être exécutées simultanément). Pour un modèle M_i^k, une contrainte de sécurité est associée à chaque requête Req_{i-1}^p! sous la forme d'une garde booléenne G_s^p définie comme par :

$$\left\| G_s^p = f_s^p(Obs\_M_{i-1}^l, \Phi) \right. \tag{2.8}$$

avec $l \neq p$ (la contrainte de sécurité relative à la sollicitation d'un élément p de niveau $i-1$ porte sur les états d'autres équipements l de même niveau).

La définition des fonctions booléennes $f_d^{(s_a,s_b)}$, f_p^{lst} et f_s^p dépend bien sûr de chaque modèle de procédé. Elles sont par ailleurs spécifiques à chaque séquence de transitions d'une localité stable source vers une localité stable cible. Ainsi, pour un modèle M_i^k possédant deux localités stables a et b, il sera nécessaire de définir :

- une fonction $f_d^{(s_a,s_b)}$ pour la contrainte de disponibilité,
- deux fonctions f_p^a et f_p^b (configurations pour les localités a et b),
- autant de fonctions f_s^p qu'il y a de requêtes $(Req_i^1, ..., Req_i^P)$, avec P nombre d'éléments sollicités par le modèle M_i^k. Pour tous les éléments m non soumis à des contraintes de sécurité, la fonction f_s^p sera la fonction booléenne constante égale à vrai.

Exemple :
Sur le cas d'étude "jouet" de la Section *2.1.3, le modèle M_2^1 caractérisant la fonction F_1 comportera les gardes suivantes :*

- $G_d^{(s_a,s_f)} = Disp\_V1 \wedge Disp\_V2 \wedge Disp\_PO1$ *où les localités s_a et s_f représentent F_1 à l'arrêt et en marche,*
- $G_f^{St1} = \neg Obs\_V1 \wedge Obs\_V2 \wedge \neg Obs\_PO1$ *où $St1$ est la localité correspondant à la fonction à l'arrêt,*
- $G_f^{St2} = Obs\_V1 \wedge \neg Obs\_V2 \wedge Obs\_PO1$ *où $St2$ est la localité correspondant*

respectivement à la fonction en marche,
- $G_s^4 = \neg Obs\_PO1 \vee (Obs\_V1 \wedge Q_{out} \neq 0)$ *associée à la requête* Req_1^4 *de démarrage de la pompe* $P01$ *et où* Q_{out} *est le débit en sortie de* $V1$ *; cette contrainte est la traduction logique de* $Obs\_PO1 \Rightarrow (Obs\_V1 \wedge Q_{out} \neq 0)$ *;* $G_s^k = true$ *pour* $k \neq 4$.

Comme indiqué précédemment, les sollicitations d'éléments de procédé de niveau inférieur sont limitées à une demande de changement d'état (Req_{i-1}^k) afin de limiter au maximum les connaissances relatives aux trajectoires de conduite qui sont embarquées dans les différents modèles. Néanmoins, afin de ne pas multiplier inutilement le nombre de transitions porteuses de ces sollicitations dans un modèle M_i^k (potentiellement égal au nombre d'éléments de niveau $i-1$), il est possible de les restreindre aux seules ressources de niveau $i-1$ employées par l'élément M_i^k. Cette idée, discutée dans [Kurshan, 1994], est reprise et adaptée par [Clarke et al., 1999] car elle permet de réduire la taille de l'espace d'état parcouru en se focalisant uniquement sur les éléments nécessaires. Cet ensemble, appelé cône d'influence [Clarke et al., 2001]et noté $\mathcal{C}(M_i^k)$, contient l'ensemble des requêtes Req_{i-1}^p! émises vers les modèles M_{i-1}^p observés dans une des contraintes fonctionnelles de M_i^k.

Définition 2 $\mathcal{C}(M_i^k) = \{Req_{i-1}^p\}$ *pour tout* p *tel que, soit* $Obs\_M_{i-1}^p$ *est un terme des gardes* G_f^{lst} *pour toute localité stable* lst *du modèle* M_i^k*, soit* $\Phi(M_{i-1}^p)$ *est un terme des gardes* G_f^{lst} *pour toute localité stable* lst *du modèle* M_i^k.

De manière similaire à la contrainte de disponibilité, le cône d'influence défini pour un modèle M_i^k (définition 2) peut être réduit aux seules ressources nécessaires pour évoluer d'une localité stable s_a à une autre s_b. L'intérêt est de circonscrire d'avantage encore la recherche de séquences aux seuls éléments de procédé influents. En revanche, cela nécessite la formalisation d'une connaissance préalable plus importante sur les trajectoires de conduite en risquant d'exclure, a priori, des séquences contenant des éléments dont le lien avec l'objectif de conduite n'est pas explicite ou envisagé. Dans ce cas, la définition 2 devient :

Définition 3 $\mathcal{C}(s_a, s_b) = \{Req_{i-1}^p\}$ *pour tout* p *tel que, soit* $Obs\_M_{i-1}^p$ *est un terme des gardes* $G_f^{s_a}$ *ou* $G_f^{s_b}$ *du modèle* M_i^k*, soit* $\Phi(M_{i-1}^p)$ *est un terme des gardes* $G_f^{s_a}$ *ou* $G_f^{s_b}$ *du modèle* M_i^k.

Exemple :
Sur le cas d'étude "jouet" de la Section *2.1.3, on a entre autres :*
- $\mathcal{C}(M_2^1) = \{Req_1^1, Req_1^2, Req_1^4\}$ *où* M_1^1 *est le modèle de la fonction* F_1 *et* Req_1^1, Req_1^2, Req_1^4 *sont les sollicitations respectives des vannes* $V1$, $V2$ *et de la pompe* $PO1$,
- $\mathcal{C}(M_3^1) = \{Req_2^1, Req_2^2, Req_2^3\}$ *où* M_3^1 *est le modèle de recette et* Req_2^1, Req_2^2, Req_2^3 *les sollicitations respectives des fonctions* F_1, F_2 *et* F_3,
- $\mathcal{C}(s_1, s_2) = \{Req_2^1, Req_2^2\}$ *où* s_1 *et* s_2 *sont des localités stables du modèle de recette* M_3^1 *caractérisant respectivement le système à l'arrêt et en refroidissement.*

La structure générique entre deux localités stables d'un modèle M_i^k pour $i \neq 1$ est donnée par la Figure 2.14 [7].

7. **Notation** : sur cette figure, la notation $\{Req_{i-1}^p\}$ représente un ensemble de transitions associées à chaque élément de $\{Req_{i-1}^p\}$. A chacune de ces transitions, est associée une garde de sécurité G_s^p conformément aux définitions précédentes.

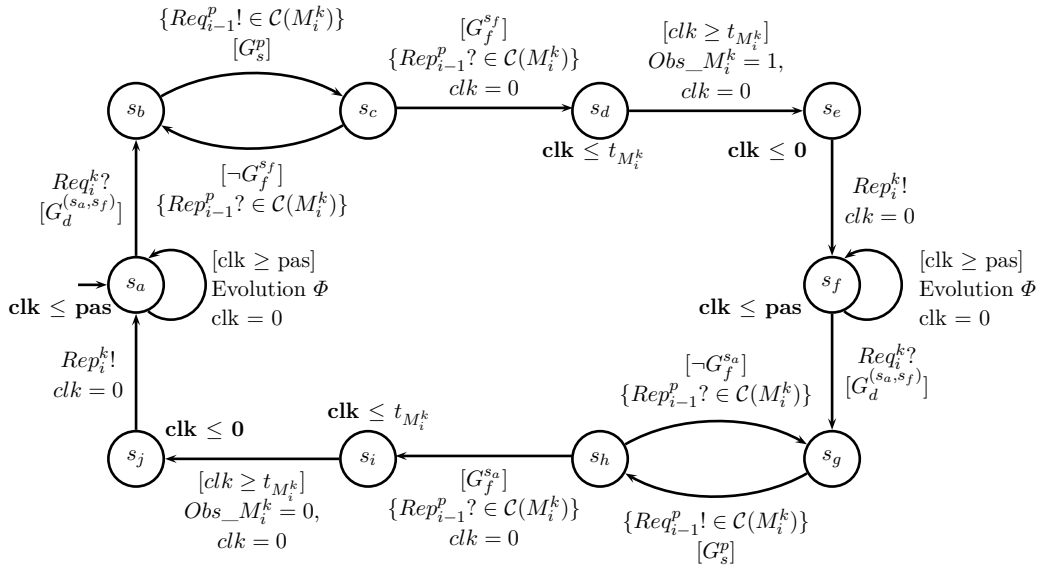


FIGURE 2.14 – Structure générique entre deux localités stables de M^k_i

Dans cette structure, l'acceptation d'une sollicitation de changement d'état $Req^k_i?$ du modèle M^k_i est conditionnée à une garde G_d dépendant de la disponibilité des ressources présentes dans le cône d'influence du modèle M^k_i.

Pour la phase de préparation, les boucles entre les localités s_b et s_c représentent l'émission successive de requêtes $Req^p_{i-1} \in \mathcal{C}(M^k_i)$ jusqu'à obtention de la configuration requise pour la localité stable s_f. Lors de la réception d'un compte-rendu Rep^p_{i-1} de l'élément sollicité :

- si la garde fonctionnelle $G^{s_f}_f$ de la localité cible n'est pas vérifiée, les transitions sortantes de s_c renvoient à la localité s_b où de nouvelles requêtes à destination des éléments présents dans le cône d'influence du modèle sont effectuées,
- dans le cas contraire, si la garde fonctionnelle $G^{s_f}_f$ est vérifiée, alors la phase de préparation se termine par la mise à jour de la variable d'observation du modèle $Obs\_M^k_i$ après une durée $t_{M^k_i}$ (qui peut être nulle), l'émission d'un compte-rendu de changement d'état $Rep^k_i!$ et l'activation de la nouvelle localité stable s_f.

Les évolutions des grandeurs physiques sont, comme pour les modèles M_1, effectuées dans les localités stables selon un pas de temps défini comme paramètre du modèle. Compte tenu de la nature du formalisme des automates temporisés mais aussi des besoins en termes de modélisation relatifs à la génération des trajectoires de conduite, les évolutions des grandeurs physiques seront représentées de manière discrète par la mise à jour de variables entières. Dans les cas d'étude traités lors du projet CONNEXION, cette approximation discrète est jugée suffisante, dans le cadre des études de conduite, pour représenter l'impact d'un élément de procédé sur les grandeurs physiques.

Une démarche équivalente est appliquée en phase de clôture pour les localités s_g, s_h et s_i.

Exemple :
La FIGURE *2.15 présente le modèle de la fonction F_1 du cas d'étude "jouet". Cette fonction possède deux localités stables s_a (arrêt) et s_f (marche). Les gardes G_d, G_f^{St1}, G_f^{St2} et G_s^4 sont celles définies précédemment. Le canal Req_F_1 est un renommage du canal Req_2^1, les canaux Req_V1, Req_V2, Req_PO1, Rep_V1, Rep_V2 et Rep_PO1 sont ceux de la* FIGURE *2.13.*

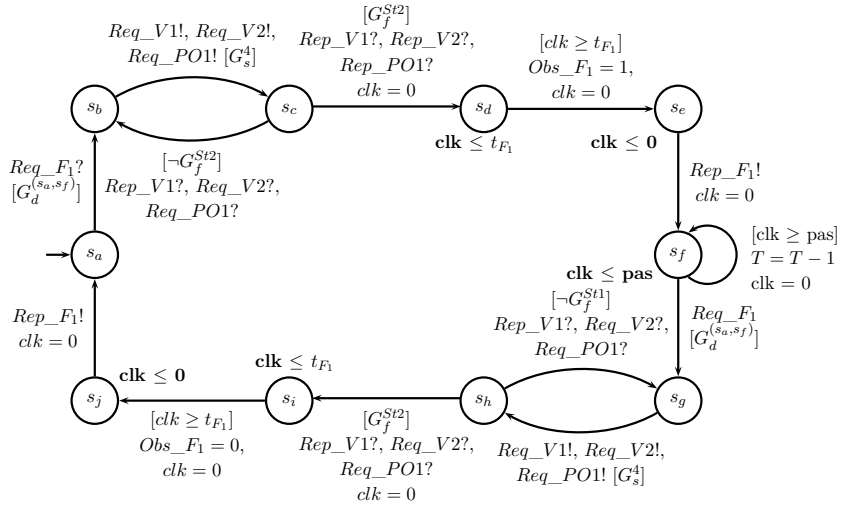


FIGURE 2.15 – Modèle de la fonction F_1

La FIGURE *2.16 présente les évolutions entre deux localités stables du modèle de recette : système à l'arrêt (s_a) et refroidissement (s_f).*

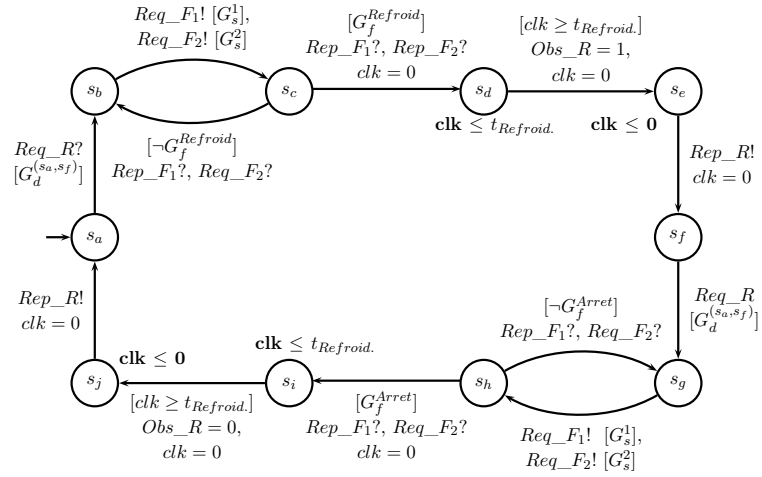


FIGURE 2.16 – Structure entre deux localités stables du modèle de Recette

- $G_d^{(s_a,s_f)} = Disp\_F_1 \vee Disp\_F_2$,
- $G_f^{Refroid} = Obs\_F_1 \vee Obs\_F_2$ *et* $G_f^{Arret} = \neg Obs\_F_1 \wedge \neg Obs\_F_2$
- *Req_R (resp. Req_R) est le canal de sollicitation (resp. de compte-rendu) d'un changement d'état du modèle de recette,*
- $Req_2^1 = Req\_F_1$, $Rep_2^1 = Rep\_F_1$, $Req_2^2 = Req\_F_2$, $Req_2^2 = Req\_F_2$

2.2.2.4 Modèles de procédé basés sur les structures génériques

Les structures génériques proposées en sections 2.2.2.2 et 2.2.2.3 permettent d'obtenir, de manière systématique, le modèle des évolutions entre deux localités stables d'un élément de procédé. Si cet élément de procédé ne possède que deux états stables, ce qui est souvent le cas (vanne tout ou rien, pompe, fonctions ayant deux modes tels que arrêt et marche, ...), la structure générique est en réalité le modèle de procédé lui-même.

Dans le cas contraire où les éléments de procédé disposent de plusieurs états de fonctionnement (vannes trois voies, fonctions possédant plusieurs modes tels que arrêt, en marche, dégradé, ...), leur modèle est obtenu par application des structures génériques entre toutes les localités stables de l'élément considéré entre lesquelles une séquence d'évolution est autorisée.

Exemple :
Considérons un système de turbo-pompes qui possède trois situations stables : à l'arrêt, marche à 60% et marche en pleine puissance. Le modèle de recette, qui décrit les évolutions entre ces trois situations et fait appel aux fonctions d'alimentation, contient trois localités stables s_1 (arret), s_2 (marche à 60%) et s_3 (pleine puissance), associées à ces trois modes de fonctionnement. Les seules évolutions autorisées entre ces trois localités stables sont : s_1 vers s_2 et s_2 vers s_1 (modélisées par une structure générique de préparation/clôture), s_2 vers s_3 et s_3 vers s_2 modélisées également par une une structure générique de préparation/clôture. En d'autres termes, nous considérons sur cet exemple qu'il est impossible d'atteindre la phase pleine puissance depuis une situation d'arrêt sans passer par une situation intermédiaire de fonctionnement à puissance réduite.

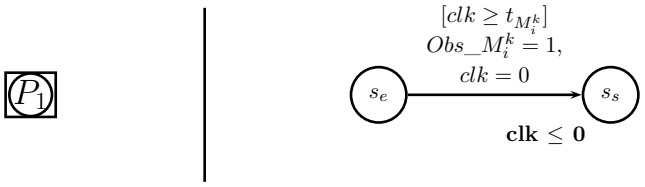
Pour simplifier la représentation de nos modèles, nous définissons une notation graphique que nous appellerons macro-localité en référence à la macro-étape des modèles Grafcet [IEC, 2002]. Dans notre cas, une macro-localité, dont la notation est présentée dans la partie gauche de la FIGURE 2.17, sera caractérisée par :

- une localité d'entrée s_e, une localité de sortie s_s et des localités internes s_{int},
- dans un modèle de procédé, la transition d'une localité s_l vers une macro-localité est une représentation graphique de la transition de cette localité s_l vers la localité d'entrée s_e de la macro-localité ; la transition d'une macro-localité vers une localité s_g est une représentation graphique de la transition de la localité de sortie de la macro-localité s_s vers cette localité s_g.

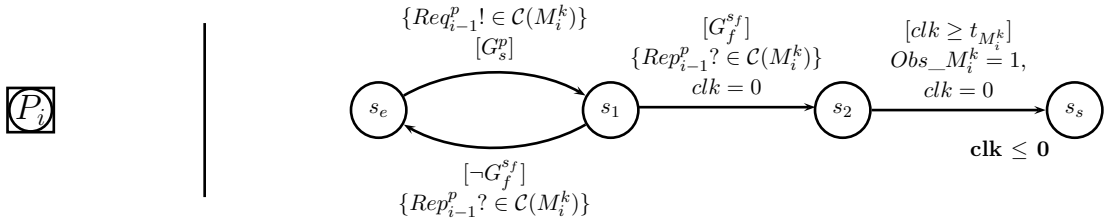
La partie droite de FIGURE 2.17 donne l'expansion d'une macro-localité représentant une phase de préparation ou de clôture et associée aux modèles de niveaux 1 (2.17a), comportant 2 localités, et de niveaux $n > 1$ comportant 4 localités (2.17b). Cette notion n'étant qu'un artifice graphique, la syntaxe et la sémantique des notations utilisées dans les gardes des transitions présentes dans ces macro-localités sont identiques à celles utilisées pour la représentation des structures génériques des figures 2.12 et 2.14.

La construction d'un modèle intégrant des macro-localités suit les règles suivantes :

- la transition d'une localité stable vers une autre ne peut impliquer qu'une seule macro-localité,



(a) Macro-localité pour les modèle M_1^k



(b) Macro-localité pour les modèle M_i^k

FIGURE 2.17 – Macro-localité

– le modèle doit être connexe, c'est à dire que quelles que soient les localités stables s_p et s_q de ce modèle, il existe une séquence de transitions de s_p vers s_q (en d'autres termes, il n'existe pas de groupes de localités absorbants).

Dans ce contexte, un modèle M_i^k possédant deux localités stables s_l et s_g est donné par le modèle de la FIGURE 2.18. Conformément aux deux règles précédentes, il est obtenu en plaçant deux macro-localités P_{lg} et P_{gl} respectivement des localités s_l vers s_g et s_g vers s_l.

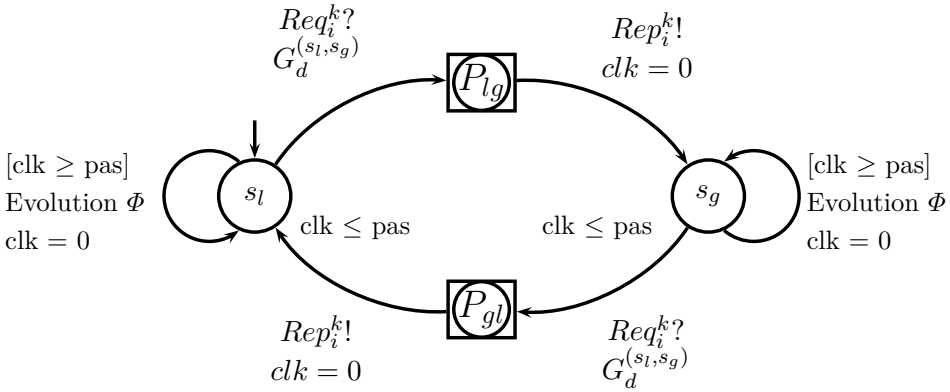


FIGURE 2.18 – Modèle d'un élément de procédé à deux localités stables

La construction des modèles comportant k localités stables dépend des transitions existant entre les différentes localités stables. Si l'évolution d'une localité stable s_a vers une localité stable s_b est possible, alors une macro-localité P_{ab} est placée entre ces deux localités stables, une transition est définie entre s_a et la localité d'entrée de P_{ab} et une transition est définie entre la localité de sortie de P_{ab} et s_b. Plusieurs remarques peuvent être formulées :

– le nombre maximal de macro-localités présentes dans un modèle possédant k localités

stables est égal au nombre de 2-arrangements sur l'ensemble des localités stables, soit :

$$\frac{k!}{(k-2)!}$$

– le nombre minimal de macro-localités présentes dans un modèle possédant k localités stables est égal à k de sorte à garantir la connexité du modèle.

Sous réserve que la propriété de connexité du modèle soit préservée, toutes les solutions comprises entre ces deux bornes peuvent être admissibles en fonction, bien sûr, du comportement spécifique modélisé.

Exemple :
Considérons le système de turbo-pompes de l'exemple précédent : il contient trois localités stables s_1 (arret), s_2 (marche à 60%) et s_3 (pleine puissance). La FIGURE *2.19 donne une solution envisageable pour le modèle de recette décrivant ce système. Cette solution considère que le passage d'un mode arrêt à un mode en pleine puissance (et inversement) impose le passage par un mode intermédiaire de puissance à 60%.*

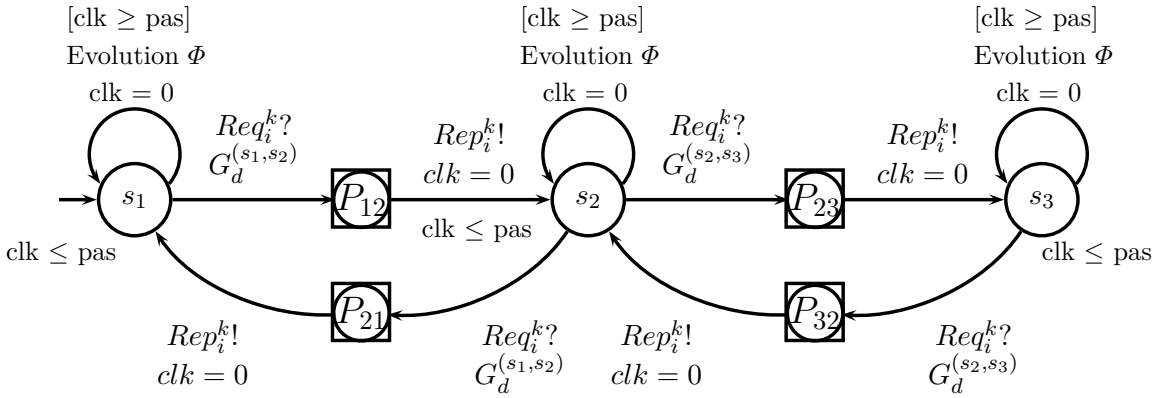


FIGURE 2.19 – Exemple de modèle de recette à trois localités stables

Si l'on considère maintenant le cas d'étude jouet, une solution de modélisation du niveau "recette" est donnée par la FIGURE *2.20 où s*1*, s_2 et s_3 correspondent respectivement à un état d'arrêt, de refroidissement (par la voie A ou par la voie B) et de remplissage en mode secours.*

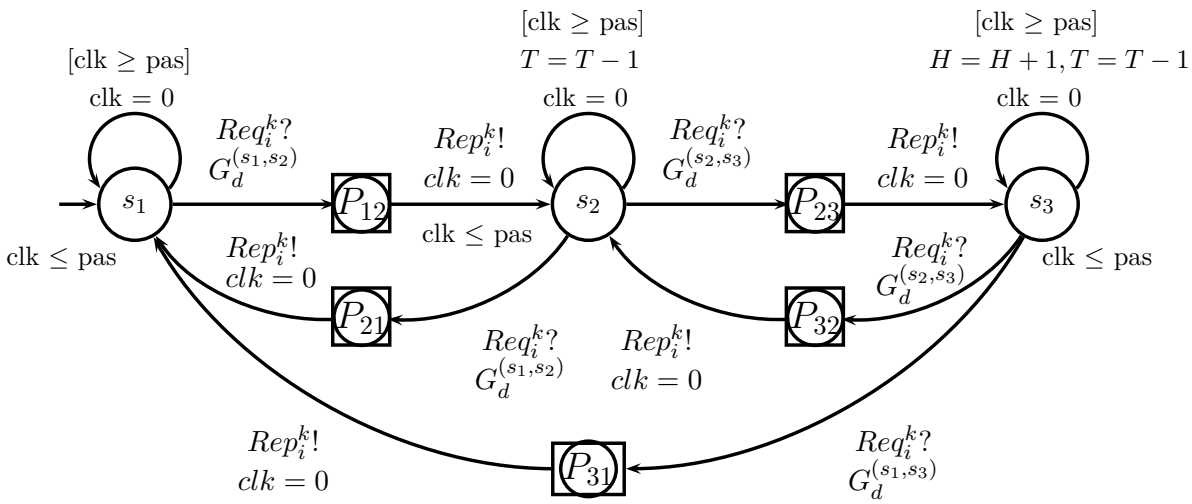


FIGURE 2.20 – Exemple de modèle de recette du cas d'étude "jouet"

Le cône d'influence mis en œuvre dans P_{12} concerne les fonctions F_1 et F_2 puisque le refroidissement peut être effectué soit par la voie A (F_1) soit par la voie B (F_2), la garde de sécurité associée à la requête sur F_1 est donnée par $\neg Obs\_F_2$ (et inversement pour F_2), la garde fonctionnelle conditionnant l'accès à la localité de sortie de P_{12} est donnée par $Obs\_F_1 \vee Obs\_F_2$.

Deux solutions sont envisageables pour le cône d'influence de P_{23} :

- *toutes les fonctions sont présentes et la garde fonctionnelle est donnée par $Obs\_F_2 \wedge Obs\_F_3$: cela signifie que le passage d'un refroidissement à un remplissage en mode secours nécessitera l'émission de requêtes en P_{23} pour stopper F_1 si elle est active puis pour démarrer F_2 et F_3 ;*
- *seule la fonction de remplissage F_3 est présente dans le cône d'influence, la garde fonctionnelle restant identique à la solution précédente : cela signifie que si F_1 est la fonction active en s_2, il faudra préalablement évoluer vers s_1 pour la stopper (requête en P_{21}) puis enclencher F_2 pour atteindre s_2 (requête en P_{12}) puis F_3 pour atteindre s_3 (requête en P_{23}).*

Enfin, il n'y pas d'évolution possible de s_1 vers s_3 car l'activation du remplissage en mode secours ne peut être effectuée que si le mode refroidissement (par la voie A ou par la voie B) est actif.

Remarque 1 : la construction du modèle de recette pour le cas d'étude jouet met en évidence le non déterminisme de nos modèles (c'est par exemple le cas des transitions sortantes de s_2 vers P_{21} ou P_{23}). Cette propriété ne constitue pas un handicap dans le cadre de notre approche puisque la génération d'une séquence d'actions de conduite est basée sur la recherche d'atteignabilité d'une localité cible du modèle « *séquenceur* » et donc sur un parcours de l'espace d'état.

Remarque 2 : le modèle de recette du cas d'étude "jouet" de la FIGURE 2.20 est une version « généralisée » du modèle de la FIGURE 2.5 où les requêtes spécifiques sont remplacées

par des requêtes de changement d'état, où la structure adopte la forme générique proposée par les patrons de modélisation et où l'utilisation des gardes fonctionnelles, de disponibilité et de sécurité a été systématisée. Ces deux représentations restent néanmoins cohérentes vis à vis du comportement du cas d'étude "jouet".

Conclusion

Dans cette section, notre objectif était de proposer des patrons de modélisation génériques pour aider le modélisateur en systématisant la démarche de construction des modèles. Pour cela, les modèles intègrent une structure générique incluant des canaux de synchronisation pour les requêtes et les comptes-rendus d'exécution ainsi que des gardes fonctionnelles, de disponibilité et de sécurité dont la formulation dépend du procédé modélisé.

Par ailleurs, les patrons de modélisation proposés reposent sur la généralisation d'une requête de changement d'état d'un élément de procédé (en lieu et place de requêtes spécifiques dédiées à cet élément de procédé) afin de limiter au maximum la connaissance relatives aux trajectoires de conduite que le modélisateur formalise dans les modèles et ainsi favoriser la génération de séquences admissibles mais a priori inconnues.

Enfin, ces patrons ne constituent bien sûr qu'une aide à la modélisation et ne prétendent pas couvrir en totalité la diversité des comportements à modéliser. Il reste donc toujours possible de recourir à des modèles spécifiques si besoin.

La section 2.3 illustre l'utilisation de tels modèles pour la génération de séquences d'actions de conduite sur le cas d'étude "jouet".

2.3 Génération de séquences d'actions de conduite

2.3.1 Génération des traces d'exécution

La démarche de recherche de séquences de conduite débute par la vérification de la propriété d'atteignabilité $EF\ Sequenceur.FinSequence$ sur l'ensemble des modèles du système $S_N, M_N, M_{N-1}, \ldots, M_1$. Si la propriété est satisfaite, une trace d'exécution vérifiant la propriété sur le modèle du système est générée. Le processus d'obtention de cette trace d'exécution est décrit par l'Equation 2.9.

$$\left\| \Gamma \leftarrow (M_N \parallel M_{N-1} \parallel ... \parallel M_1 \parallel S_N) \models EF\ S_i.FinSequence \right. \tag{2.9}$$

Cette trace d'exécution contient l'ensemble des états et des transitions parcourus menant à l'atteinte de la situation désirée.

2.3.2 Génération d'une séquence admissible

Une séquence d'actions de conduite est une séquence constituée des différentes opérations qui doivent être réalisées sur les éléments d'une installation. Ceux-ci peuvent être, selon le contexte :

- les systèmes d'actionnement et de mesure qui correspondent, dans le cadre de notre démarche, aux éléments décrits par les modèles de procédé de plus bas niveau (ouverture/fermeture d'une vanne, démarrage/arrêt d'une pompe, d'un moteur, configuration d'un transmetteur, ...),

– les éléments de contrôle-commande ou de conduite qui correspondent, dans le cadre de notre démarche, aux éléments décrits par les modèles de procédé situés hiérarchiquement au dessus du plus bas niveau (démarrage/arrêt d'une fonction, d'un mode de fonctionnement, d'un sous-système, ...).

La séquence est obtenue à partir de la trace d'exécution permettant d'atteindre la localité cible du modèle *sequenceur*. La trace permettra d'obtenir une séquence d'actions de conduite à opérer sur les systèmes d'actionnement et de mesure qui composent le système.

Ces séquences d'actionnement et de mesure peuvent s'avérer utile, en phase d'ingénierie, pour vérifier l'existence d'une trajectoire de conduite dans des conditions aux limites (notamment en cas d'indisponibilité de certains équipements) et, en phase d'exploitation, pour préparer un plan d'intervention ou proposer une aide à la définition de stratégies de conduite.

Compte tenu de la formalisation proposée dans le cadre de notre approche, volontairement limitée à des requêtes de changements d'état pour ne pas restreindre l'exploration des trajectoires de conduite aux seules trajectoires connues, les séquences d'actions de conduite doivent contenir l'identification de l'équipement sollicité, sa situation de départ et la situation atteinte après changement d'état.

Les séquences doivent donc être définies par un quadruplet (Req_i^p, v_d, v_f, n), où :

– Req_i^p indique que l'élément p de niveau i doit subir un changement d'état (par exemple une requête de changement d'état de la vanne $V1$),
– v_d, v_f indiquent les valeurs des variables d'observations ($Obs\_M_i^p$) respectivement avant et après le changement d'état Req_i^p (par exemple : Req_V1, v_d = *vanne V1 ouverte*, v_f = *vanne V1 fermée*),
– n indique le niveau du modèle de l'élément sollicité dans la hiérarchie.

L'Algorithme 1 permet l'obtention de la séquence d'actions de conduite associée à Γ à opérer sur les éléments du système.

Exemple :
A titre d'exemple, la séquence générée par recherche d'atteignabilité sur le cas d'étude « jouet » produit entre autres la séquence :
Σ = *Req_S, Req_R, Req_F*$_1$*, Req_V*1*, Req_P*1*, Req_F*$_2$*, Req_V*2*, Req_P*2*, Req_R, Req_F*$_1$*, Req_V*1*, Req_S, Req_R, Req_F*$_3$*, Req_V*3
où Req_S est une requête de niveau "système" (niveau 4), Req_R une requête de niveau "recette" (niveau 3), Req_F$_i$ *des requêtes de niveau "fonction" (niveau 2) et Req_Vi et Req_Pi sont des requêtes de niveau "équipements" (vannes et pompes de niveau 1).*

2.3.3 Discussion

La séquence particulière présentée en Section 2.3.2 a été déterminée en 369 ms, ce qui est tout à fait acceptable pour le cas d'étude "jouet". De plus, si le processus de model-checking est lancé plusieurs fois, il est alors possible, puisque le processus d'exploration est aléatoire, de générer différentes séquences d'actions de conduite pour atteindre la situation désirée. Une

Algorithme 1 Génération d'une séquence d'actions Σ pour le modèle $S_N \| M_N \| M_{N-1} \| ... \| M_1$

Entrées :

% Trace d'exécution %

$\Gamma = E_1 \overset{T_1}{\rightarrow} E_2 ... \overset{T_N}{\rightarrow} E_X$

Sorties :

$\Sigma = \sigma_1 \sigma_2 ... \sigma_X$ avec $\sigma_s = (l \in L, v_d \in V, v_f \in V, n \in \mathbb{N}^*)$

Longueur de la séquence : $|\Sigma|$

$k \leftarrow 1$; $s \leftarrow 1$

Pour k=1 **à** X **faire**

  Si $T_k = (t_\alpha, t_\beta) \wedge t_\alpha \in M_i \wedge t_\beta \in M_{i-1}^p \wedge l_k \in L_r^{M_{i-1}^p} = Req_{i-1}^p$ **alors**

    $label \leftarrow l_k$; $v_d \leftarrow v(E_k)$; $n \leftarrow i - 1$

    $v_f \leftarrow v(E_{k+1})$ *% $v(E_{k+1})$ valeur des variables dans l'état E_{k+1} de la trace %*

    $\sigma_s \leftarrow (label, v_d, v_f, n)$

    $s \leftarrow s + 1$

    $k \leftarrow k + 1$

  Fin Si

  $k \leftarrow k + 1$

Fin Pour

Retourner $\Sigma = \sigma_1 \sigma_2 ... \sigma_{s-1}$

campagne a été menée afin d'effectuer 5 000 vérifications de la même propriété sur les mêmes modèles et conduisant à différentes séquences. Une synthèse des temps de génération est fournie en FIGURE 2.21 : le temps minimal d'exploration est de 270 ms, le temps moyen d'exploration est de 402 ms et le temps maximal d'exploration est de 3 787 ms.

Ces résultats, bien qu'apparaissant globalement bons, doivent toutefois être nuancés par le fait que les modèles sont de petites tailles. On ne peut que craindre que le passage à l'échelle sur des systèmes comportant plusieurs dizaines, voire plusieurs centaines d'équipements, se traduise par des phénomènes d'explosion combinatoire conduisant au mieux à des temps de calculs longs et à l'utilisation d'un espace mémoire important et au pire à un échec du processus de vérification [Cochard et al., 2015b].

Une approche de contournement classique de ce problème de passage à l'échelle consiste à réduire la taille des modèles par des techniques de décomposition (des modèles et des propriétés à vérifier) et d'abstraction. Cet artifice de vérification s'avère naturel et pertinent dans notre cas dans la mesure où les besoins en phase d'ingénierie ou d'aide à la conduite sont souvent limités à un sous-ensemble du problème traité : connaître par exemple les fonctions à solliciter pour atteindre une situation (sans se préoccuper des actions à réaliser sur les équipements), connaître la séquence de configuration d'une fonction en respectant les contraintes de sécurité, ...

Conclusion

Ce chapitre apporte des éléments de modélisation génériques, réutilisables pour la modélisation d'un procédé à grande échelle. L'approche de génération de séquences présentée

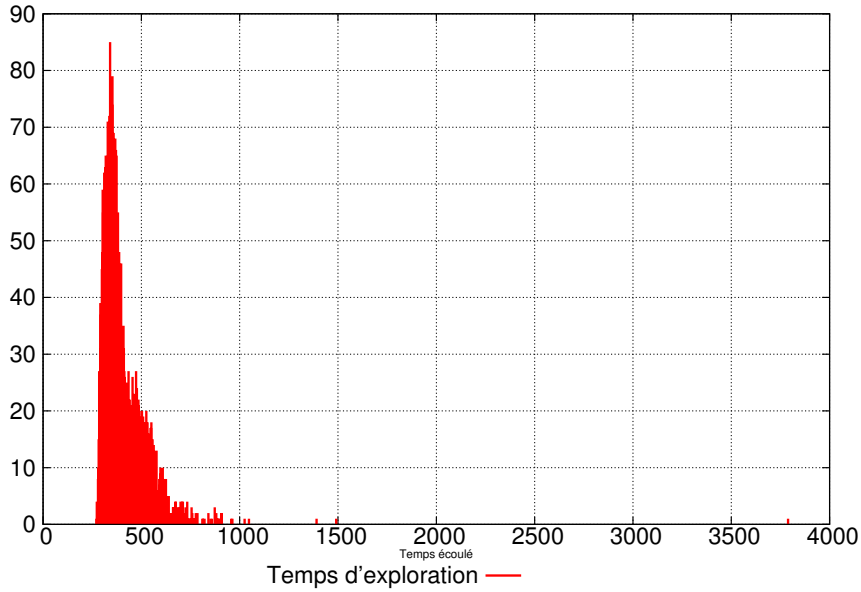


FIGURE 2.21 – Temps d'explorations pour la vérification d'une propriété d'atteignable sur l'exemple d'illustration

pour le cas d'étude, bien que générique et fonctionnelle, soulève la question du risque de phénomène d'explosion combinatoire encouru lors du passage à l'échelle. Son utilité est donc limitée en l'état actuel, c'est pourquoi cette question sera traitée dans le Chapitre 3 par la proposition d'une démarche de génération itérative des séquences basée sur la décomposition et l'abstraction des modèles de procédé selon leur niveaux.

Chapitre 3

Génération itérative de séquences d'actions de conduite sûres

Sommaire

Introduction

Ce chapitre présente une méthodologie itérative pour la génération de séquences d'actions de conduite sûres. La méthodologie proposée repose sur le cadre de modélisation présenté au chapitre précédent et sur des mécanismes de recherche d'atteignabilité pour extraire une trace d'exécution conduisant d'un état source à un état recherché.

Dans le chapitre précédent, la recherche d'atteignabilité est appliquée sur un modèle global du procédé construit à partir de la synchronisation de l'ensemble des modèles de la structure hiérarchique définie au chapitre 1 : équipements, fonctions, recette, ... Si cette approche s'avère efficace pour générer des séquences de conduite d'un système simple, elle est vite confrontée à un problème d'explosion combinatoire, lié à la taille de l'espace d'état à parcourir, qui limite son usage dans un contexte industriel.

La suite de ce chapitre est donc consacrée à une proposition méthodologique visant à réduire les limites évoquées précédemment par la proposition d'une démarche itérative de génération basée sur la décomposition en niveaux hiérarchiques et sur des mécanismes d'abstraction, permettant notamment de réduire la taille des modèles utilisés et de l'espace d'état parcouru.

3.1 Approche itérative pour la génération de séquences

La réduction de l'espace d'état nécessaire pour la vérification de propriété par des techniques automatiques est un challenge adressé de façon régulière, pouvant se traiter sous deux angles :

- d'une part, un travail sur les modèles peut être effectué pour réduire la taille des données prises en compte ;
- d'autre part, des algorithmes d'exploration plus performants peuvent être développés.

Dans ce manuscrit, nous proposons d'améliorer les méthodes de génération de traces d'exécutions par méthodes automatiques en se focalisant sur l'angle de la modélisation. Pour cela, des techniques d'abstraction [Faraut et al., 2009, Bouajjani et al., 2004] peuvent se révéler efficaces pour réduire la quantité de données nécessaire à l'étude. Dans d'autres travaux, on peux noter également des méthodologies de réduction de l'espace d'état basées sur l'estimation de la perte de données par des approximations qualifiées d'acceptables. Il s'agit là d'un exemple des limitations de la réduction des modèles, que la perte d'informations peut également mener à des résultats inacceptables sur un modèle plus complet. Bien sûr, le choix de l'outil de model checking peut influencer les performances en matière de rapidité d'exploration de l'espace d'état. On trouve divers outils logiciels, tels que SPIN [Holzmann, 1997], NuSMV [Cimatti et al., 2002], Uppaal [Behrmann et al., 2006a], ... Tous ont leurs forces et leurs faiblesses [D'silva et al., 2008], et leur efficacité dépend en outre des aspects de modélisation que l'on souhaite prendre en compte (temporisation, hiérarchie, modèle hybride, ...).

3.1.1 Techniques d'abstraction

La génération des séquences via des mécanismes de recherche d'atteignabilité sur l'ensemble des modèles de procédé (de M_N à M_1), couplés à un séquenceur unique partageant des canaux de synchronisation avec le (ou les) modèles de procédé de plus haut niveau, présente des risques d'explosion combinatoire comme nous l'avons souligné en fin de Section 2.3.

Les techniques d'abstraction [Clarke et al., 1994] permettent, par la prise en compte des seuls éléments d'intérêt, de réduire le nombre des données prises en compte, avec parfois le risque de pertes d'informations.

Exemple :
Si l'interrogation posée est de déterminer le signe du résultat de la multiplication 684×-652*, il n'est pas nécessaire de faire le calcul et d'obtenir la valeur numérique, mais simplement de considérer les signes de chaque nombre et d'utiliser la « règle des signes ». Il suffit alors de savoir que si l'on multiplie un nombre positif par un nombre négatif, le résultat sera négatif.*

Chaque abstraction que l'on souhaite effectuer nécessite donc de préserver l'information nécessaire à la cohérence du modèle en accord avec l'utilisation que l'on souhaite en faire et la/les propriété(s) que l'on souhaite vérifier. Pour mieux expliquer comment utiliser ce principe pour diminuer la taille des modèles, considérons l'exemple ci-dessous issu de [Clarke et al., 2000], basé sur le fonctionnement d'un feu tricolore FIGURE 3.1.

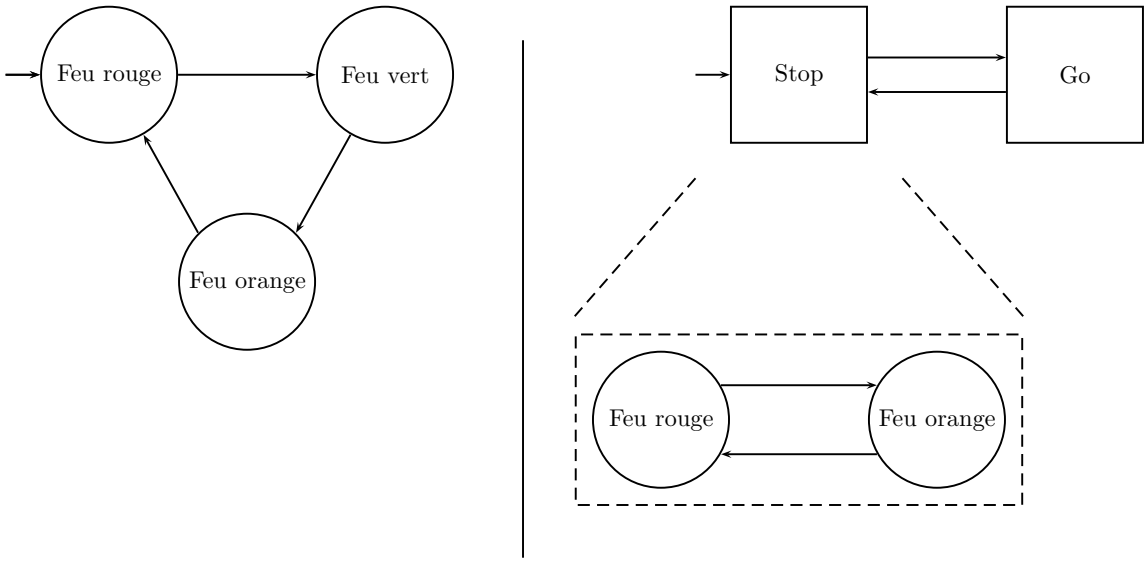


FIGURE 3.1 – Exemple simple d'abstraction

Si l'on souhaite disposer d'un modèle pour la gestion des modes de marche des feux de signalisation, il faut conserver l'information de feu orange. Cependant, si l'analyse porte sur le dimensionnement du trafic à un carrefour, l'information nécessaire se limite à savoir si un véhicule peut passer ou non.

Ce principe d'abstraction va être utilisé dans notre proposition en vue de réduire la taille de l'espace d'état parcouru. Le principe retenu est de ne considérer qu'une partie des niveaux hiérarchiques à la fois, plus précisément, des ensembles de deux niveaux.

3.1.2 Présentation générale de l'approche

Pour limiter les risques d'explosion combinatoire, on propose ici une approche de génération itérative de séquences utilisant des techniques d'abstraction pour réduire le nombre et la taille des modèles considérés ([Cochard et al., 2016]). Nous proposons, pour cela, de limiter l'exploration de l'espace d'état en se focalisant uniquement sur une paire de niveaux successifs de modèles M_i et M_{i-1} et sur un modèle séquenceur partageant des canaux de synchronisation avec M_i (Section 3.1.3). Dans ce contexte, deux processus sont à l'œuvre :

- un processus d'abstraction (Section 3.1.4) permettant de définir pour un modèle M_i, une version abstraite des modèles du niveau inférieur en les limitant aux éléments partagés par les deux niveaux considérés ; ce mécanisme est appliqué dans le cadre d'une **démarche ascendante** qui, partant de l'ensemble connu des modèles de procédé, construit progressivement des abstractions des modèles M_{i-1} depuis M_1 jusqu'à M_{N-1} ;
- un processus de génération itératif des modèles séquenceur S_i (Section 3.1.5) utilisables pour la recherche d'atteignabilité sur les modèles de la paire (M_i, M_{i-1}) ; ce mécanisme est appliqué dans le cadre d'une **démarche descendante**, qui, partant du modèle séquenceur de plus haut niveau S_N, supposé connu, génère de manière itérative les séquenceurs S_i associés aux paires (M_i, M_{i-1}) jusqu'au modèle séquenceur de niveau 2.

La Figure 3.2 présente une vision synthétique et globale de la démarche proposée dont les étapes sont détaillées dans les trois sections suivantes ([Cochard et al., 2017]).

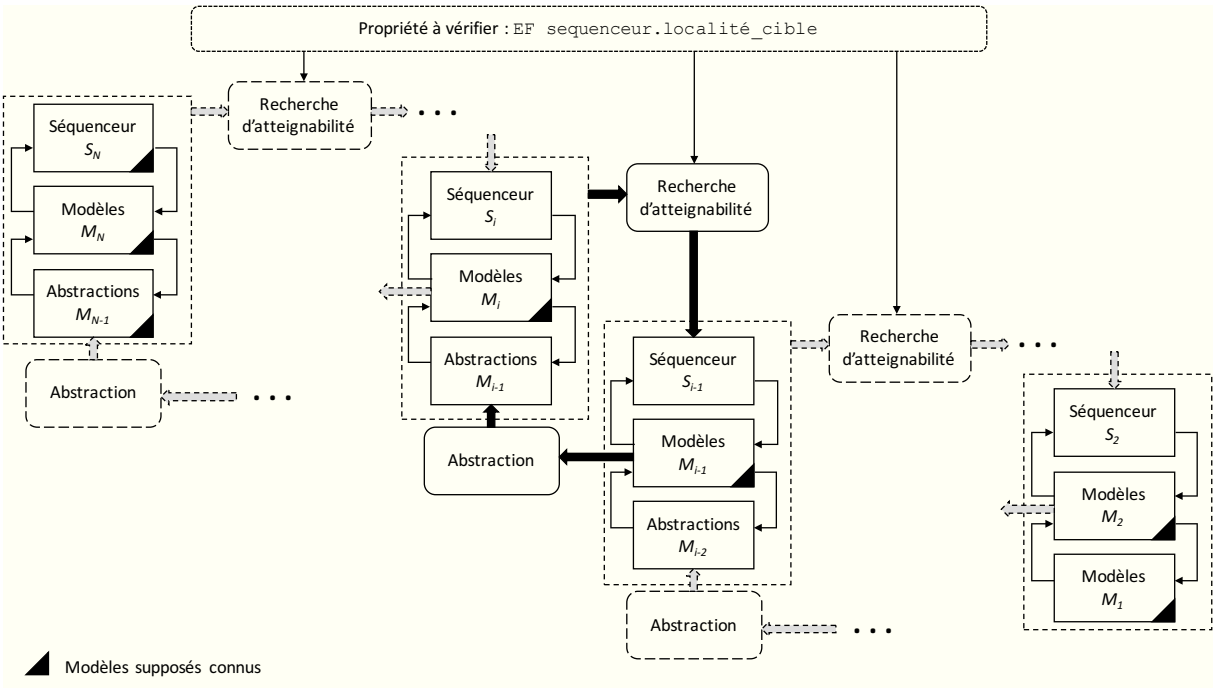


Figure 3.2 – Vue de synthèse de la méthodologie globale

3.1.3 Exploration de l'espace d'état par paires de niveaux

Une itération de recherche d'atteignabilité permet de générer une séquence à partir des modèles M_i et M_{i-1} ainsi que d'un séquenceur S_i. En d'autres termes, la séquence obtenue traduit les actions à réaliser sur les éléments de procédé de niveau $i-1$ pour faire évoluer les modèles de niveau i conformément aux sollicitations du modèle séquenceur S_i requises pour atteindre une localité cible.

Exemple :
Pour le cas d'étude "jouet", deux itérations de recherche d'atteignabilité sont mises en œuvre : la première pour les couples d'ensembles de modèles des niveaux (recette, fonctions), la seconde pour les niveaux (fonctions, équipements).
La première indique quelles sont les fonctions à démarrer ou à stopper pour atteindre la situation (arrêt, refroidissement ou remplissage) requise par le modèle séquenceur S_3. La seconde indique quelles sont les actions à opérer sur les équipements pour permettre l'exécution d'une des trois fonctions. Le modèle séquenceur S_2 de ce niveau devra donc clairement expliciter la fonction que l'on souhaite démarrer ou stopper.

Les données supposées connues pour conduire les itérations de recherche d'atteignabilité sont les ensembles de modèles de procédé M_1 à M_N ainsi que le modèle séquenceur de plus haut niveau S_N. L'interface de synchronisation entre les modèles de procédé a été définie, par la section précédente, sous la forme de canaux standard Req_i^k et Rep_i^k.

Il s'en suit qu'un modèle M_i^k comportant des canaux $Req_i^k?$, $Rep_i^k!$ (requête/compte-rendu de changement d'état du modèle M_i^k) et $Req_{i-1}^p!$, $Rep_{i-1}^p?$ (requête/compte-rendu de changement d'état des modèles de niveau inférieur) sera bloquant :

- s'il n'existe pas dans les modèles de niveau supérieur M_{i+1} au moins une transition étiquetée avec un label $Req_i^k!$ et une transition étiquetée avec un label $Rep_i^k?$,
- s'il n'existe pas dans les modèles de niveau inférieur M_{i-1} au moins une transition étiquetée avec un label $Req_{i-1}^p?$ et une transition étiquetée avec un label $Rep_{i-1}^p!$.

Lorsque l'on considère une itération de recherche d'atteignabilité sur les modèles M_i, M_{i-1} et S_i, les conditions de non blocage deviennent :

1. les modèles M_{i-1}^p comportent au moins une transition étiquetée avec chacun des labels $Req_{i-1}^p?$ et $Rep_{i-1}^p!$ assurant la synchronisation avec les modèles M_i^k,
2. les modèles M_{i-1}^p ne comportent aucune transition étiquetée avec un des labels $Req_{i-2}^t!$ et $Rep_{i-2}^t?$ utilisés pour la sollicitation des éléments de niveau inférieur M_{i-2}.
3. le modèle de séquenceur S_i comporte au moins une transition étiquetée avec chacun des labels $Req_i^k!$ et $Rep_i^k?$ des modèles M_i^k.

La condition 1 est respectée par hypothèse si les modèles de procédé sont construits en suivant les préconisations de la Section 2.2.

La condition 2 nécessite une modification des modèles M_{i-1}^p puisque ceux-ci contiennent, par construction, des labels de synchronisation avec les modèles M_{i-2}^t alors même que ces derniers sont exclus de la recherche d'atteignabilité par paires (M_i, M_{i-1}). La seule exception concerne

les modèles de plus bas niveau M_1 qui, par construction, ne sont synchronisés à aucun modèle de niveau inférieur. Les modèles utilisés pour une itération sur la paire (M_i, M_{i-1}) seront donc construits par abstraction des modèles M_{i-1} et notés $\mathcal{A}(M^p_{i-1})$.

Enfin, la condition 3 devra être vérifiée par construction des séquenceurs S_i dans les deux cas de figure suivants :

- un modèle séquenceur S_i est construit manuellement pour répondre de manière *ad-hoc* à une recherche de séquences sur la paire de niveaux considérée (M_i, M_{i-1}) ; cela peut être le cas lorsqu'un agent de conduite recherche les actions à opérer sur les équipements pour assurer le démarrage d'une fonction sans se préoccuper des niveaux supérieurs ou bien s'il recherche les fonctions à activer pour atteindre un état désiré de son système sans se préoccuper des actions à opérer sur les équipements ;
- le modèle séquenceur S_i est généré automatiquement lors de la recherche d'atteignabilité effectuée sur la paire précédente (M_{i+1}, M_i) en considérant que le séquenceur S_N de plus haut niveau est connu et correct par construction.

La FIGURE 3.3 présente la structuration des modèles M_i, M_{i-1} et S_i sur lesquels est effectuée une itération de recherche d'atteignabilité. Les deux sections suivantes présentent les algorithmes mis en œuvre pour obtenir les modèles abstraits $\mathcal{A}(M^p_{i-1})$ et le modèle séquenceur S_i.

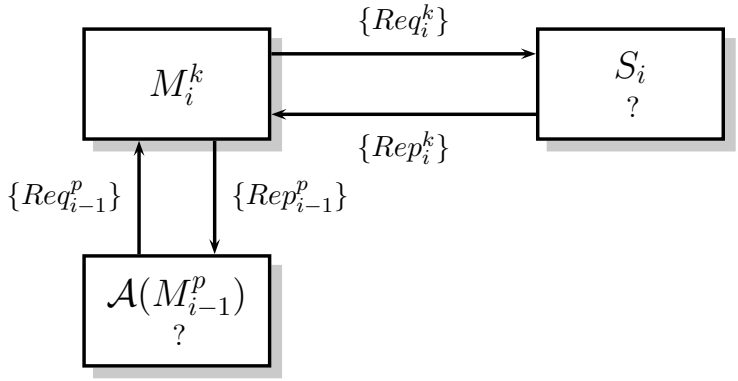


FIGURE 3.3 – Modèles utilisés pour une itération de recherche d'atteignabilité

3.1.4 Réduction de modèle par abstraction

Pour effectuer une recherche d'atteignabilité par paires de niveau (M_i, M_{i-1}), il est donc nécessaire de supprimer au préalable toute forme de synchronisation avec les modèles M_{i-2} exclus de cette recherche. D'après la démarche formalisée en SECTION 2.2, les modèles concernés par ces synchronisations sont les modèles M_{i-1}. Leur abstraction $\mathcal{A}(M^p_{i-1})$ consiste à supprimer les transitions sur lesquelles figurent des événements ou des gardes relatifs aux modèles M_{n-2}, c'est-à-dire les transitions modélisant le processus de configuration de l'élément considéré. Cette abstraction repose sur deux hypothèses :

1. lorsqu'une demande de changement d'état est formulée pour un modèle M^p_{i-1}, on suppose qu'il existe au moins une configuration des ressources de M^p_{i-1} (modèles M^m_{i-2}) qui permette de satisfaire une de ses contraintes fonctionnelles. En d'autres termes, on suppose que les ressources présentes dans son cône d'influence sont suffisantes pour lui permettre un

changement d'état. A titre d'exemple, si la fonction F_1 possède deux ressources $R1$ et $R2$, alors il existe au moins une configuration de $R1$ et $R2$ qui permette le démarrage de F_1 ;

2. les contraintes éventuelles d'exclusion entre deux éléments de même niveau (M^m_{i-2} et M^n_{i-2}), pouvant être sollicités par différents éléments de niveau supérieur, sont traitées par les contraintes de sécurité au niveau des modèles M_{i-1}.

Lorsque ces deux hypothèses sont vérifiées, on considère que les phases de préparation et de clôture pourront toujours admettre une séquence de sollicitations des éléments M^m_{i-2} permettant d'atteindre la localité stable de M^p_{i-1} recherchée. On suppose donc que les enchaînements de requêtes / réponses des ressources du niveau inférieur sont effectués : en réalité, ils seront considérés lors de l'itération de recherche d'atteignabilité concernant la paire (M_{i-1}, M_{i-2}). Le modèle abstrait ne reprendra donc que les informations partagées par les éléments des niveaux supérieurs en faisant abstraction des phases de préparation et de clôture. Cette réduction des modèles M_{i-1} est réaliste tant que ces derniers contiennent les informations nécessaires pour trouver une configuration des modèles M_{i-1} permettant d'atteindre la localité recherchée d'un modèle M^k_i.

L'abstraction $\mathcal{A}(M^p_{i-1})$ d'un modèle M^p_{i-1} consiste donc à supprimer toutes les transitions du modèle M^p_{i-1} étiquetées par un label de synchronisation partagé avec un modèle de niveau inférieur M_{i-2}, c'est-à-dire les labels Req^m_{i-2} et Rep^m_{i-2}. La suppression d'une transition d'une localité s_1 vers une localité s_2 se traduit par :

- la fusion de ces deux localités en une localité s_{12} si et seulement si la localité s_2 n'est plus atteignable depuis l'état initial ;
- si s_1 et s_2 sont fusionnées et qu'il existe une transition de s_2 vers s_1 (non supprimée par l'abstraction), alors cette transition est conservée en boucle sur la localité s_{12} issue de la fusion.
- si s_1 et s_2 sont fusionnés et qu'elles sont associées à deux invariants I_{s_1} et I_{s_2}, alors la localité fusionnée est associée à un invariant $I_{s_{12}}$ représentant la somme logique des invariants I_{s_1} et I_{s_2} ;
- les mises à jour de variables sont reportées sur les transitions entrantes de la localité fusionnée s_{12}.

Par ailleurs, l'abstraction $\mathcal{A}(M^p_{i-1})$ d'un modèle M^p_{i-1} nécessite de supprimer, dans les gardes de ce modèle, toute référence à une observation ($Obs\_M^m_{i-2}$ ou $Disp\_M^m_{i-2}$) associée aux modèles de niveau inférieur. Cette contrainte se traduit par :

- la suppression des gardes de sécurité G^m_s associées à une requête de changement d'état d'un élément de niveau inférieur $Req^m_{i-2}!$: cela n'engendre aucune perte d'information compte tenu des hypothèses 1 et 2 (on ne considère pas les phases de préparation et de clôture qui seront traitées par l'itération de recherche d'atteignabilité suivante) ;
- la suppression des gardes fonctionnelles G^{lst}_f associées à un compte-rendu de changement d'état d'un élément de niveau inférieur $Rep^m_{i-2}?$: cela n'engendre aucune perte d'information dans la mesure où, toujours selon les hypothèses 1 et 2, il existe au moins une configuration des ressources, que l'on ne cherche pas à déterminer à ce stade, permettant de les satisfaire ;
- les gardes de disponibilités $G^{(s_a,s_b)}_d$ dépendent de variables représentant la disponibilité des

éléments de procédé de niveau inférieur ; la valeur de la garde (*true* ou $false$) devra donc être évaluée et affectée avant le lancement de l'itération de recherche d'atteignabilité ; dans notre cas, cette affectation peut être aisément réalisée dans la mesure où la disponibilité n'est pas considéré comme une variable dynamique mais comme un paramètre fixé pour une recherche d'une séquence de conduite ; la démarche d'abstraction étant ascendante, c'est-à-dire partant des modèles de plus bas niveau vers les modèles de plus haut niveau, la connaissance de ces paramètres pour les éléments de procédé de plus bas niveau permet l'évaluation successive des gardes de disponibilité pour chaque niveau d'abstraction.

Ces principes sont formalisés dans l'ALGORITHME 2 permettant l'abstraction d'un modèle M^p_{i-1}. Cet algorithme est applicable aux patrons de modélisation proposés précédemment mais aussi à tout modèle de procédé développé spécifiquement pour les besoins particuliers d'un système. Il traduit une forme de projection [8] du langage associé à M^p_{i-1} sur les alphabets des modèles M^k_i considérés dans l'itération de recherche d'atteignabilité par paire.

L'application de cet algorithme au patron de modélisation présenté par la FIGURE 2.14 aboutit au modèle abstrait de la FIGURE 3.4. Sa structure est cohérente avec celle du patron de modélisation des sous-systèmes M_1 de plus bas niveau (FIGURE 2.12) dans ma mesure où ces modèles n'intègrent pas de phase de préparation et de clôture. L'algorithme d'abstraction permet toutefois de généraliser la démarche et de l'appliquer à tout type de modèle.

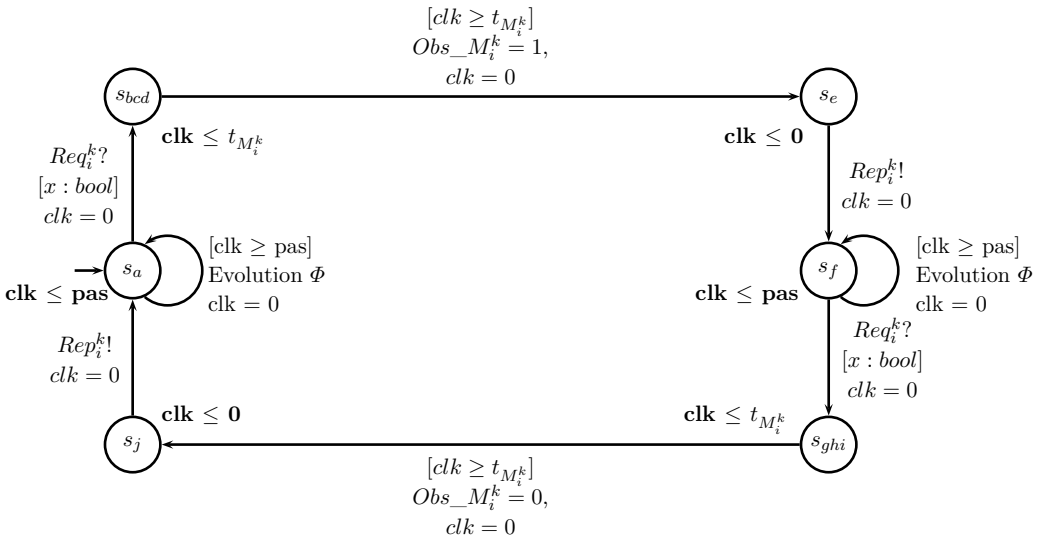


FIGURE 3.4 – Modèle abstrait $\mathcal{A}(M^k_i)$ obtenu par application de l'algorithme 2

Cette méthode d'abstraction permet donc d'obtenir un ensemble de modèles $(M_i, \mathcal{A}(M_{i-1}))$ nécessaire pour une recherche d'atteignabilité par paires de niveau. Pour pouvoir effectuer cette recherche, il convient également de définir le modèle séquenceur S_i contenant l'objectif

8. L'opérateur de projection d'une séquence σ sur un alphabet Σ_p peut être défini de manière récursive par :

$$\forall \sigma = \sigma_1\sigma_2...\sigma_{|\sigma|} \in \Sigma, Proj(\sigma/\Sigma_p) : \begin{cases} \epsilon & \text{si } \sigma = \epsilon \\ Proj(\sigma/\Sigma_p) & \text{si } \sigma = \delta\alpha \text{ et } \alpha \notin \Sigma_p \\ Proj(\sigma/\Sigma_p)\alpha & \text{si } \sigma = \delta\alpha \text{ et } \alpha \in \Sigma_p \end{cases}$$

de conduite associé à cette paire (voir FIGURE 3.3).

Algorithme 2 Construction d'un modèle abstrait de M^p_{i-1}

Entrées :
$M^p_{i-1} = (S^{M^p_{i-1}}, V^{M^p_{i-1}}, X^{M^p_{i-1}}, I^{M^p_{i-1}}, L^{M^p_{i-1}}, T^{M^p_{i-1}}, S_m^{M^p_{i-1}}, s_0^{M^p_{i-1}}, v_0^{M^p_{i-1}})$
$M_{i-2} = \cup_{q=1}^{|M_{i-2}|} M^q_{i-2}$
Sorties :
$\mathcal{A}(M^p_{i-1}) = (S^{\mathcal{A}(M^p_{i-1})}, V^{M^p_{i-1}}, X^{M^p_{i-1}}, I^{\mathcal{A}(M^p_{i-1})}, L^{\mathcal{A}(M^p_{i-1})}, T^{\mathcal{A}(M^p_{i-1})}, S_m^{M^p_{i-1}}, s_0^{M^p_{i-1}}, v_0^{M^p_{i-1}})$

$\mathcal{A}(M^p_{i-1}) \leftarrow M^p_{i-1}$
Pour tout $t_i = (s_i, l_i, G_d^{(s_i,s_j)}, m_i, s_j) \in T^{M^p_{i-1}}$ **faire** %*Valuation des gardes* G_d%
$x_i : bool \leftarrow f_d^{(s_i,s_j)}(Disp\_M^q_{i-2})$; $t_i\prime = (s_i, l_i, x_i, m_i, s_j)$
$T^{\mathcal{A}(M^p_{i-1})} \leftarrow \{T^{M^p_{i-1}}\} - t_i + t_i\prime$
Fin Pour

Pour tout $t_i = (s_i, l_i, g_i, m_i, s_j) \in T^{M^p_{i-1}}$ **faire**
Si $l_i \in L^{M_{i-2}}$ **alors** %*Suppression des Req et Rep de niveau inférieur*%
$T^{\mathcal{A}(M^p_{i-1})} \leftarrow \{T^{M^p_{i-1}}\} - t_i$
$L^{\mathcal{A}(M^p_{i-1})} \leftarrow \{L^{M^p_{i-1}}\} - l_i$
Pour tout $t_k = (s_k, l_k, g_k, m_k, s_h) \in T^{M^p_{i-1}}$ tel que $s_h = s_i$ **faire**
$m_k \leftarrow m_k \cup m_i$
Fin Pour
Si $t_j = (s_j, l_j, g_j, m_j, s_i) \notin T^{\mathcal{A}(M^p_{i-1})}$ **alors** %*Fusion des localités* s_i *et* s_j%
$S^{\mathcal{A}(M^p_{i-1})} \leftarrow \{S^{M^p_{i-1}}\} - s_i - s_j + s_{ij}$
$I_{ij} \leftarrow I_i \cup I_j$; $I^{\mathcal{A}(M^p_{i-1})} \leftarrow \{S^{M^p_{i-1}}\} - I_i - I_j + I_{ij}$
Pour tout $t_k = (s_k, l_k, g_k, m_k, s_h) \in T^{M^p_{i-1}}$ tel que $s_h = s_i$ ou $s_h = s_j$ **faire**
$t_k\prime = (s_k, l_k, g_k, m_k, s_{ij})$; $T^{\mathcal{A}(M^p_{i-1})} \leftarrow \{T^{M^p_{i-1}}\} - t_k + t_k\prime$
Fin Pour
Pour tout $t_k = (s_k, l_k, g_k, m_k, s_h) \in T^{M^p_{i-1}}$ tel que $s_k = s_i$ ou $s_k = s_j$ **faire**
$t_k\prime = (s_{ij}, l_k, g_k, m_k, s_h)$; $T^{\mathcal{A}(M^p_{i-1})} \leftarrow \{T^{M^p_{i-1}}\} - t_k + t_k\prime$
Fin Pour
Fin Si
Fin Si
Fin Pour

3.1.5 Génération itérative des séquenceurs

Compte tenu de la structure hiérarchique de nos modèles de procédé, la recherche d'atteignabilité par paire de niveaux consiste donc à appliquer de manière itérative une recherche d'atteignabilité opérant sur un ensemble cohérent de modèles $(S_i, M_o, \mathcal{A}(M_{i-1}))$. Chaque itération permet la génération d'une séquence d'actions de conduite de niveau i agissant uniquement sur les éléments de niveau $i-1$ (FIGURE 3.5). Ainsi, sur une décomposition en N

niveaux, $N-1$ modèles séquenceur interviennent dans les différentes itérations. Pour rappel, le modèle séquenceur de plus haut niveau S_N, qui modélise l'objectif de conduite à atteindre au niveau système, est supposé connu et les modèles de plus bas niveau M_1 ne nécessitent pas d'être abstraits pour conduire la recherche d'atteignabilité sur la paire de niveau (M_2, M_1).

Exemple :
En se référant à l'organisation en trois niveaux hiérarchiques de l'exemple de la SECTION *2.1.3.2, les itérations de recherche d'atteignabilité sont appliquées sur les triplets (Séquenceur$_3$, recette, $\mathcal{A}(fonctions)$) et (Séquenceur$_2$, $fonctions$, équipements). Le modèle Séquenceur$_3$ est supposé connu et les modèles de plus bas niveau (équipements) ne sont font pas l'objet d'une abstraction.*

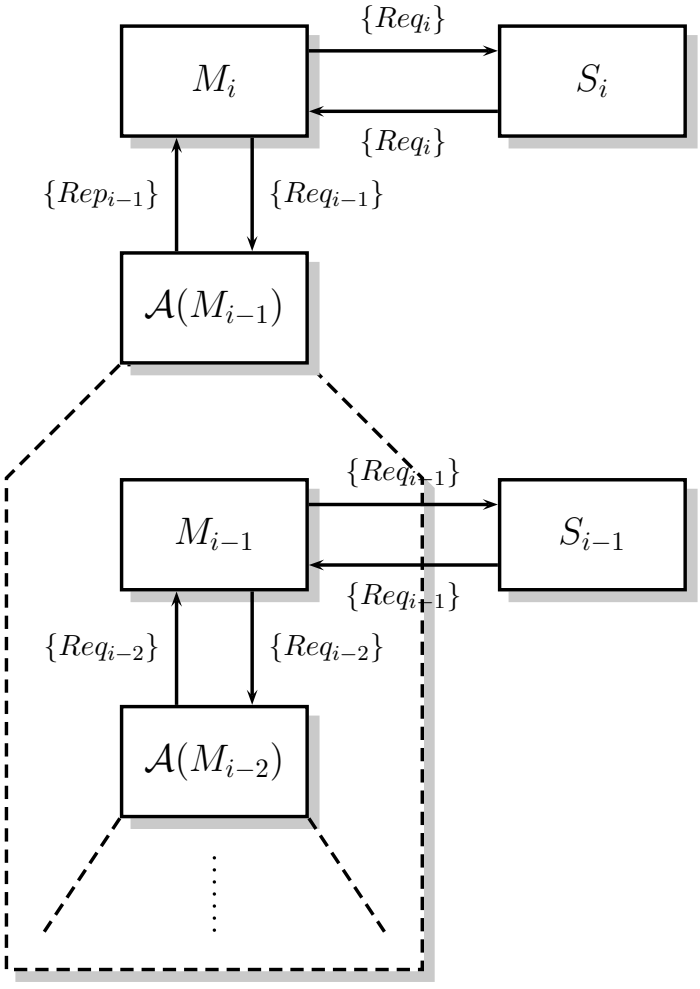


FIGURE 3.5 – Organisation des modèles par paires de niveaux

Disposant d'une trace d'exécution $\Gamma_i = E1 \overset{T_1}{\rightarrow} E2 \ldots \overset{T_{|\Gamma_i|}}{\rightarrow} E_{|\Gamma_i|}$, générée par une itération de recherche d'atteignabilité sur l'ensemble de modèles $(S_i, M_i, \mathcal{A}(M_{i-1}))$, l'idée est de l'utiliser pour déterminer le modèle séquenceur S_{n-1} opérant sur la paire de niveau inférieur $(M_{i-1}, \mathcal{A}(M_{i-2}))$.

En effet, cette trace Γ_i contient l'ensemble des états du réseau d'automates parcourus et des transitions franchies pour atteindre la localité cible du modèle séquenceur S_n depuis les états initiaux des modèles (M_i, M_{i-1}). Elle contient en particulier des canaux de synchronisation partagés entre les modèles M_i et le modèle séquenceur S_i (Req_i, Rep_i), mais aussi entre les modèles M_i et M_{i-1} (Req_{i-1}, Rep_{i-1}). Les premiers correspondent à des décisions de changement d'état des modèles M_i, tandis que les seconds correspondent à des sollicitations de ressources M_{i-1} pour rendre effectives ces décisions.

Dans la mesure où, par construction, un modèle séquenceur S_{i-1} ne doit contenir que des canaux de synchronisation et variables partagées avec les modèles M_{i-1}, la trace Γ_i devra être expurgée de tout autre élément, et en particulier des canaux de synchronisation partagés entre les modèles M_i et le modèle séquenceur S_i. Le séquenceur S_{i-1} est donc obtenu à partir de cette trace en s'inspirant de l'opérateur de projection défini en note de bas de page [8]. L'ALGORITHME 3 procède à la génération d'un séquenceur S_{i-1} à partir de la trace d'exécution Γ_i en examinant successivement l'ensemble des transitions de cette trace $E_i \xrightarrow{T_i} E_{i+1}$.

Algorithme 3 Génération du séquenceur S_{i-1} à partir de la trace Γ_i

Entrées :

Trace d'exécution $\Gamma_i = E_1 \xrightarrow{T_1} E_2 ... \xrightarrow{T_N} s_N$

Sorties :

Automate séquenceur $S_{i-1} = (P^{S_{i-1}}, V^{S_{i-1}}, X^{S_{i-1}}, I^{S_{i-1}}, L^{S_{i-1}}, T^{S_{i-1}}, p_m^{S_{i-1}}, p_0^{S_{i-1}}, v_0^{S_{i-1}})$

$I^{S_{i-1}} \leftarrow \emptyset$; $X^{S_{i-1}} \leftarrow \emptyset$; $p_0^{S_{i-1}} \leftarrow s_1$
$k \leftarrow 1$; $\sigma \leftarrow 1$
Pour k=1 to N **faire**

% Mémorisation des changements de valeurs des variables des modèles M_{i-1} %
Pour tout $v \in X^{M_{i-1}}$ **faire**
Si $v(S_k) \neq v(S_{k+1})$ **alors**
$V^{S_{i-1}} \leftarrow V^{S_{i-1}} \cup v$
$g_v \leftarrow [v = v(S_{k+1})]$
Fin Si
Fin Pour

% Traitement des transitions T_k qui partagent un canal de synchronisation entre M_i et M_{i-1} %
Si $T_k = (t_\alpha, t_\beta) \wedge t_\alpha \in M_i \wedge t_\beta \in M_{i-1}$ **alors**
$g \leftarrow \prod_{v \in V^{S_{i-1}}} g_v$
Si $l_k \in L_r^{M_{i-1}}$ **alors** $T_\sigma \leftarrow (s_\sigma, l_k!, g, \emptyset, s_{\sigma+1})$ **Fin Si**
Si $l_k \in L_e^{M_{i-1}}$ **alors** $T_\sigma \leftarrow (s_\sigma, l_k?, g, \emptyset, s_{\sigma+1})$ **Fin Si**
$P^{S_{i-1}} \leftarrow P^{S_{i-1}} \cup s_\sigma \cup s_{\sigma+1}$; $T^{S_{i-1}} \leftarrow T^{S_{i-1}} \cup T_\sigma$; $L^{S_{i-1}} \leftarrow L^{S_{i-1}} \cup I_k$
$p_m^{S_{i-1}} \leftarrow s_{\sigma+1}$
$\sigma \leftarrow \sigma + 1$
Pour tout $v \in X^{M_{i-1}}$ **faire** $g_v \leftarrow true$ **Fin Pour**
Fin Si
$k \leftarrow k + 1$
Fin Pour

Les principes mis en œuvre par cet algorithme sont les suivants :

- si la transition examinée est associée à un canal de synchronisation entre les modèles M_i et M_{i-1}, alors cette transition ainsi que ses deux états source et cible sont conservés pour le séquenceur S_{i-1} ; le canal de synchronisation associé est un canal d'émission lorsqu'il est présent en tant que canal de réception dans les modèles M_{i-1} et inversement,
- lorsqu'un changement de valeur d'une variable des modèles M_{i-1} est observé entre deux états successifs de la trace et que la transition entre ces deux états n'est pas conservée dans le séquenceur S_{i-1}, il est mémorisé afin d'être réutilisé ultérieurement comme garde de la prochaine transition qui sera conservée dans le séquenceur ; ce mécanisme est indispensable pour conserver les conditions sur les états des modèles M_{i-1} (observations sur les localités et grandeurs physiques) qui devront être respectées pour atteindre la situation de conduite recherchée,
- les invariants et les horloges, sans intérêt pour la modélisation des objectifs de conduite car relatifs à des contraintes comportementales inhérentes au procédé (et donc prises en compte par les modèles de procédé) ne sont pas conservés.

Exemple :
On considère l'itération de recherche d'atteignabilité effectuée sur le triplet (S_3, recette, $\mathcal{A}(fonctions)$) relatif à l'exemple de la Section *2.1.3 dont les modèles de recette et de fonctions sont donnés respectivement par la* Figure *2.15 et la* Figure *2.16 et le modèle séquenceur par la* Figure *3.6 (atteinte du mode refroidissement depuis l'état initial).*

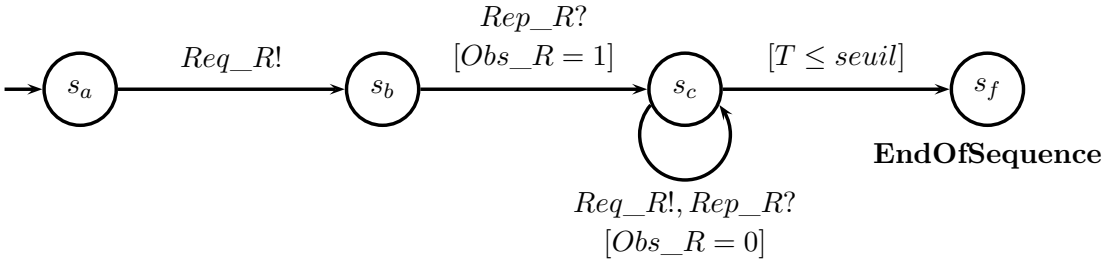


Figure 3.6 – Modèle du séquenceur de plus haut niveau S_3

La Figure *3.7 propose une trace d'exécution admissible vérifiant la propriété $EF\ S_3.EndOfSequence$ où le refroidissement est effectué via la voie A ($F1$).*
Les transitions T_1 et T_6 correspondent au franchissement simultané de deux transitions des modèles de recette et du séquenceur S_3 associées, pour T_1, au canal Req_R et, pour T_6, au canal Rep_R. Les transitions T_2 et T_4 correspondent au franchissement simultané de deux transitions des modèles de recette et $\mathcal{A}(F1)$ et des associées, pour T_1, au canal Req_F1 et, pour T_6, au canal Rep_F1.
Les transitions T_3 et T_5 correspondent respectivement à une transition du modèle $\mathcal{A}(F1)$ dont le franchissement est conditionné par la durée de mise en service de la fonction $F1$ et à une transition du modèle de recette dont le franchissement est conditionné par la durée de mise en service du refroidissement.
Enfin, la transition T_7 correspond au franchissement de la transition bouclée sur la localité s_f du modèle $\mathcal{A}(F1)$ qui assure les changements de valeur de la variable physique de température T.

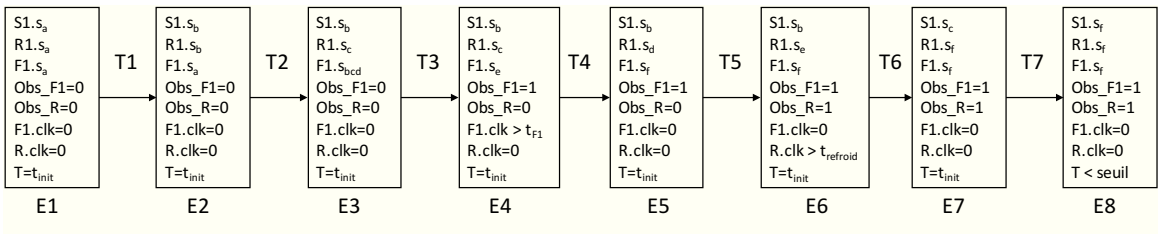


FIGURE 3.7 – Trace d'exécution pour le triplet ($Séquenceur_3$, $recette$, $\mathcal{A}(fonctions)$)

*Le modèle de séquenceur $Séquenceur_2$ permettant d'effectuer une itération de rercherche d'atteignabilité sur la paire de modèles ($fonctions$, équipements) est obtenu par application de l'*ALGORITHME *3 sur cette trace. Il est représenté par la* FIGURE *3.8.*

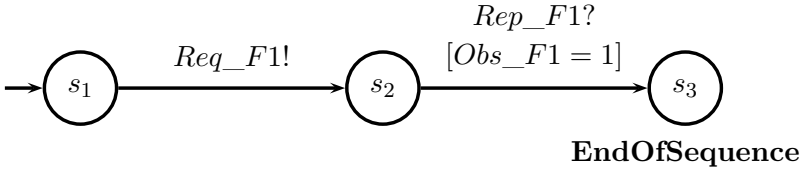


FIGURE 3.8 – Séquenceur S_2 généré

3.2 Génération itérative des séquences d'actions de conduite

3.2.1 Génération itérative des traces d'exécution et construction de séquences

La démarche de recherche de séquences de conduite par itérations successives sur des paires de niveau débute nécessairement par les modèles de plus haut niveau (S_N, M_N, M_{N-1}). A ce niveau, une trace d'exécution vérifiant une propriété d'atteignabilité sur le modèle de séquenceur S_N est générée. L'utilisation de l'ALGORITHME 3 permet d'obtenir le modèle de séquenceur S_{N_1} servant de base à l'itération de recherche d'atteignabilité suivante. Ce processus, décrit par l'ALGORITHME 4 peut être poursuivi jusqu'à un niveau décrété par l'utilisateur, selon ses besoins, ou au pire jusqu'au niveau le plus bas. Le résultat est un ensemble de traces d'exécution correspondant à chaque paire de niveau.

Cette approche permet de limiter de manière très significative les phénomènes d'explosion combinatoire en réduisant le spectre d'exploration mais présente, en contre-partie, des limites quant à l'optimalité de la solution proposée. En effet, lorsqu'une itération est réalisée, elle aboutit à une trace qui constitue un choix de conduite indépendant des conséquences qu'il pourra avoir sur les éléments de procédé de niveaux inférieurs. Par construction, les itérations suivantes sur les paires de niveaux inférieurs permettront de trouver une solution mais la solution globale peut ne pas être la plus pertinente. Ces questions font l'objet de la section suivante.

La séquence est obtenue à partir de la trace d'exécution permettant d'atteindre la localité cible du modèle séquenceur. Selon la démarche mise en œuvre, la trace permettra :

– d'obtenir une séquence d'actions de conduite à opérer sur les systèmes d'actionnement et de mesure si la recherche d'atteignabilité est effectuée :

  - sur le triplet (S_2, M_2, M_1) dans le cadre de l'approche de génération itérative de séquences proposée en SECTION 3.1,
  - sur l'ensemble des modèles $(M_1, ..., M_N)$ et du séquenceur S_N dans le cadre de l'approche globale proposée en SECTION 2.2.
- d'obtenir une séquence d'actions de conduite à opérer sur les éléments de niveau i si la dernière recherche d'atteignabilité est effectuée sur le triplet (S_{i+1}, M_{i+1}, M_i) dans le cadre de l'approche de génération itérative de séquences proposée en SECTION 3.1 (le processus itératif débute nécessairement par une recherche d'atteignabilité avec le modèle séquenceur de plus haut niveau mais peut être stoppé à tout moment).

Algorithme 4 Génération des traces d'exécution par paire de niveau

Entrées :

Ensemble de modèles de procédé M_i pour $i \in [1..N]$

Modèle séquenceur de plus haut niveau : S_N Niveau d'arrêt : n_{stop} (par défaut égal à 1)

Sorties :

Traces d'exécution : Γ_i correspondant à une recherche d'atteignabilité sur $(S_i, M_i, \mathcal{A}(M_{i-1}))$

$i \leftarrow N$

Tant que $i \neq n_{stop}$ **faire**

$\Gamma_i \leftarrow (M_i \parallel \mathcal{A}(M_{i-1}) \parallel S_i) \models EF\ S_i.FinSequence$

Si $i > 2$ **alors** $S_{i-1} \leftarrow$ ALGORITHME 3(Γ_i) **Fin Si**

$i \leftarrow i - 1$

Fin Tant que

Retourner $\Gamma_N, \Gamma_{N-1}, ...\Gamma_{n_{stop}+1}$

L'ALGORITHME 5 permet l'obtention d'une séquence d'actions de conduite à opérer sur les éléments des modèles M_i et M_{i-1}.

A partir des séquences d'actions de conduite établies par paire de niveaux, il est également possible de procéder à la reconstruction d'une séquence globale qui combine les sollicitations sur les éléments des différents niveaux. Celle-ci offre une vue globale de la trajectoire que n'offre pas les séquences d'actions par paire de niveau et qui peut s'avérer très utile en conduite.

L'ALGORITHME 6 qui permet la reconstruction d'une séquence globale repose sur un parcours itératif des séquences générées. Lorsque l'action de la séquence examinée fait appel à un élément de plus bas niveau, l'algorithme passe à l'examen de la séquence inférieure. Lorsque la séquence examinée est celle de plus bas niveau (séquence Σ_2 lorsque toutes les séquences sont générées), on conserve les actions de cette séquence tant qu'elles concernent des éléments de plus bas niveau (éléments de niveau 1 lorsque toutes les séquences sont générées). Lorsque cela n'est plus le cas, un parcours ascendant est proposé jusqu'à rencontrer une action d'une séquence Σ_i faisant appel à un élément de niveau inférieur $i - 1$.

Algorithme 5 Génération d'une séquence d'actions Σ_i pour la paire (M_i, M_{i-1})

Entrées :

% Trace d'exécution obtenue sur (S_i, M_i, M_{i-1}) dans le cas d'une approche itérative %

$\Gamma_i = E_1 \overset{T_1}{\rightarrow} E_2 ... \overset{T_N}{\rightarrow} E_X$

Sorties :

$\Sigma_i = \sigma_1^i \sigma_2^i ... \sigma_X^i$ avec $\sigma_s^i = (l \in L^{M_i}, v_d \in V^{M_i} \cup V^{M_{i-1}}, v_f \in V^{M_i} \cup V^{M_{i-1}}, n \in \mathbb{N}^*)$

Longueur de la séquence : $|\Sigma_i|$

$k \leftarrow 1$; $s \leftarrow 1$

Pour k=1 to X **faire**

Si $T_k = (t_\alpha, t_\beta) \wedge t_\alpha \in M_i \wedge t_\beta \in M_{i-1}^p \wedge l_k \in L_r^{M_{i-1}^p} = Req_{i-1}^p$ **alors**

$label \leftarrow l_k$; $v_d \leftarrow v(E_k)$; $n \leftarrow i - 1$

$index \leftarrow k$

Répéter *% Recherche du label dual Rep_{i-1}^p %*

$k \leftarrow k + 1$

Jusqu'à $T_k = (t_\alpha, t_\beta) \wedge t_\alpha \in M_i \wedge t_\beta \in M_{i-1}^p \wedge l_k = Rep_{i-1}^p$

$v_d \leftarrow v(E_{k+1})$ *% $v(E_{k+1})$ valeur des variables dans l'état E_{k+1} de la trace %*

$\sigma_s \leftarrow (label, v_d, v_f, n)$

$s \leftarrow s + 1$; $k \leftarrow index$

sinon

Si $T_k = (t_\alpha, t_\beta) \wedge t_\alpha \in S_i \wedge t_\beta \in M_i^q \wedge l_k \in L_r^{M_i^q} = Req_i^q$ **alors**

$label \leftarrow l_k$; $v_d \leftarrow v(E_k)$; $n \leftarrow i$

$index \leftarrow k$

Répéter *% Recherche du label dual Rep_i^q %*

$k \leftarrow k + 1$

Jusqu'à $T_k = (t_\alpha, t_\beta) \wedge t_\alpha \in S_i \wedge t_\beta \in M_i^q \wedge l_k = Rep_i^q$

$v_d \leftarrow v(E_{k+1})$

$\sigma_s \leftarrow (label, v_d, v_f, n)$

$s \leftarrow s + 1$; $k \leftarrow index$

Fin Si

Fin Si

$k \leftarrow k + 1$

Fin Pour

Retourner $\Sigma_i = \sigma_1 \sigma_2 ... \sigma_{s-1}$

Exemple :

A titre d'exemple, si les itérations de recherche d'atteignabilité produisent les trois séquences suivantes :

$\Sigma_4 = Req\_S, Req\_R, Req\_R, Req\_R, Req\_S$

$\Sigma_3 = Req\_R, Req\_F1, Req\_F2, Req\_R, Req\_F3, Req\_R, Req\_F4$

$\Sigma_2 = Req\_F1, Req\_V1, Req\_P1, Req\_F2, Req\_V2, Req\_P2, Req\_F3, Req\_V3, Req\_F4, Req\_V4$

*où Req_S est une requête de niveau "système" (niveau 4), Req_R une requête de niveau "recette" (niveau 3), Req_Fi des requêtes de niveau "fonction" (niveau 2) et Req_Vi et Req_Pi sont des requêtes de niveau "équipements" (niveau 1). La séquence globale Σ_G produite par L'*ALGORITHME *6 peut être représentée sous forme structurée par niveau :*

$$
\begin{aligned}
Req\_S \rightarrow Req\_R \rightarrow Req\_F1 & \\
& Req\_V1 \\
& Req\_P1 \\
\rightarrow Req\_F2 & \\
& Req\_V2 \\
& Req\_P2 \\
\rightarrow Req\_R \rightarrow Req\_F3 & \\
& Req\_V3 \\
Req\_S \rightarrow Req\_R \rightarrow Req\_F4 & \\
& Req\_V4
\end{aligned}
$$

La séquence obtenue est admissible, car elle permet d'atteindre l'état recherché à partir d'un état de départ, et sûre car elle respecte l'ensemble des contraintes fonctionnelles, de disponibilité et de sécurité, garanties par construction des modèles.

En revanche, la pertinence de cette trace vis à vis d'un savoir-faire métier de conduite ne peut être garantie. En effet, cette trace étant obtenue en effectuant une vérification de propriété d'atteignabilité, elle dépend de la façon dont le model-checker :

- explore l'espace de l'état (recherche en profondeur, recherche en largeur, recherche aléatoire par exemple) ;
- des conditions d'arrêt qu'il utilise (arrêt de l'exploration dès qu'une trace est obtenue, exploration complète de l'espace d'état pour trouver la trace la plus courte en longueur ou en temps d'horloge).

Dans la mesure où l'exploration complète est rejetée pour des raisons liées au facteur d'échelle industriel auquel s'adressent ces travaux de thèse, la trace obtenue correspondra donc à la première solution rencontrée par le model-checker. Il est évident que, dans ces conditions, elle pourra être entachée d'opérations redondantes inutiles et n'offrira aucune garantie quant à sa longueur. Nous proposons en ce sens des méthodes d'optimisation des séquences générées permettant, par quelques principes basiques, de simplifier les séquences obtenues [9].

9. Il peut également être noté que le processus de model checking peut être répété à souhait pour obtenir diverses séquences admissibles et sûres

Algorithme 6 Construction d'une séquence globale d'actions de conduite Σ_G

Entrées :

Séquences d'actions de conduite par paires de niveaux définies pour $i \in [M...N]$ avec N nombre de niveaux de modélisation et M niveau de séquence le plus bas (si l'approche de génération des séquences est complète, M est égale à 2) :
$\Sigma_i = \sigma_1^i \sigma_2^i ... \sigma_{|\Sigma_i|}^i$ avec $\sigma_j^i.n$ niveau associé à l'action σ_j^i et $|\Sigma_i|$ longueur de la séquence.

Sorties :

$\Sigma_G = \alpha_1 \alpha_2 ... \alpha_{|\Sigma_G|}$

For $i = 2$ **à** N **faire** $p_i \leftarrow 1$ **Fin Pour** *% p_i pointeur sur la séquence Σ_i %*
$i \leftarrow N$
Répéter *% Examen de l'action $\sigma_i^{p_i}$ %*
  Si $\sigma_i^{p_i}.n = i$ **alors** *% action de même niveau que la séquence %*
    Si $ascendant = false$ **alors**
      $\Sigma_G \leftarrow \Sigma_G.\sigma_i^{p_i}$
      $p_i \leftarrow p_i + 1$
    sinon
      Si $j = N$ **alors** $ascendant \leftarrow false$ **sinon** $i \leftarrow i + 1$ **Fin Si**
    Fin Si
  sinon *% $\sigma_i^{p_i}.n < i$, action sur un élément de niveau inférieur %*
    Si $i = M$ **alors**
      $\Sigma_G \leftarrow \Sigma_G.\sigma_i^{p_i}$
      $p_M \leftarrow p_M + 1$; $ascendant \leftarrow true$
    sinon
      $p_i \leftarrow p_i + 1$; $ascendant \leftarrow false$
      $i \leftarrow i - 1$
    Fin Si
  Fin Si
  $arret \leftarrow true$
  For $i = M$ **à** N **faire** $arret \leftarrow arret \wedge (p_i > |\Sigma_i|)$
Jusqu'à arret
Retourner Σ_G

3.2.2 Génération d'une séquence optimisée

3.2.2.1 Simplification des séquences

Le parcours de l'espace d'état peut engendrer des séquences où deux actions de conduite, potentiellement antagonistes, se succèdent (par exemple une séquence d'actions contenant une requête d'ouverture de la vanne $V1$ immédiatement suivie d'une requête de fermeture de cette même vanne). Plus formellement, ces actions suspectes peuvent s'écrire sous la forme : $\sigma_k^i = (Req_{i-1}^p, x, y, i-1)$ et $\sigma_{k+1}^i = (Req_{i-1}^p, y, x, i-1)$ avec $\sigma_k^i, \sigma_{k+1}^i \in \Sigma_i$ et où x, y sont les valeurs prises par la variable d'observation de l'état de l'élément M_{i-1}^p dans les deuxième (avant exécution de la requête) et troisième (après exécution de la requête) termes des quadruplets.

A priori, il semble naturel d'éliminer ces actions de la séquence. Toutefois, il est possible que du temps se soit écoulé entre ces deux requêtes et que, dans un de ses états stables, l'élément M_{i-1}^{p} ait modifié des variables physiques. Dans la mesure où ces modifications peuvent s'avérer primordiales pour l'exécution d'autres éléments, il est, en l'état des connaissances incluses dans la séquence, difficile de procéder à la suppression de ces actions suspectes.

A titre d'exemple, considérons une vanne conditionnant la vidange d'une cuve. Après avoir demandé l'ouverture de la vanne pour faire une vidange, il est possible de demander sa fermeture une fois la cuve vidée et d'enclencher alors une autre fonction. Deux cas de figure peuvent se présenter :

- la variable de niveau est mise à jour au niveau du modèle de la vanne ; dans ce cas, la séquence contiendra des actions suspectes qu'il ne sera pas possible de supprimer,
- la variable de niveau est mise à jour dans un modèle de la fonction *vidange* : dans ce cas, cela suppose que les requêtes d'ouverture et de fermeture seront précédées respectivement d'une requête de démarrage et d'arrêt de la fonction *vidange*, ce qui conduira à une séquence ne comportant pas d'actions suspectes.

Même si le deuxième cas de figure de ce petit exemple semble de loin le plus probable, il est difficile d'écarter le premier sans informations supplémentaires sur la manière dont le système a été modélisé et sur le savoir-faire métier. Ce constat peut être étendu à des morceaux de séquences potentiellement antagonistes (*ouvrir V1*, *démarrer PO1*, *arrêter PO1*, *fermer V1*) sans pouvoir y apporter plus de réponses définitives. Pour ces raisons, le choix a été fait de ne procéder à aucune simplification mais de privilégier une optimisation des séquences de conduite selon des critères temporels ou de longueur.

3.2.2.2 Optimisation locale par paire de niveaux

La vérification d'une propriété d'atteignabilité par model-checking permet d'effectuer une optimisation mono-critère à l'aide du processus itératif suivant :

- le critère à minimiser (ou à maximiser) est intégré dans la propriété d'atteignabilté à vérifier sous la forme d'une borne (critère inférieur à la borne si l'on cherche à le minimiser ou supérieur à la borne si l'on cherche à le maximiser),
- si la propriété est vérifiée, cela signifie que la borne est respectée et on peut alors la diminuer (si l'on cherche à minimiser le critère) ou l'augmenter (si l'on cheche à le maximiser),
- la procédure de décrémentation ou d'incrémentation de la borne se poursuit tant que le model-checker parvient à vérifier la propriété,
- lorsque le model-checker ne parvient plus à vérifier la propriété, la dernière valide de la borne est considérée comme la valeur optimale du critère.

Cette technique a été avantageusement mise en œuvre par [Ruel, 2009] et [Lemattre, 2013] pour résoudre respectivement des problèmes d'optimisation de performances et d'allocation d'architectures de contrôle-commande en réseau. Elle présente cependant une limite dans la mesure où l'optimalité de la solution proposée ne peut être garantie. En effet, le fait que le model-checker ne parvienne pas à vérifier la propriété ne signifie pas, pour autant, qu'elle n'est pas valide. L'échec peut en effet être dû à un problème d'explosion combinatoire qui stoppe le

processus de preuve. Néanmoins, la dernière valeur vérifiée de la borne constitue une très bonne valeur approchée de l'optimal.

Dans notre cas, cette technique, même si elle ne garantit pas l'optimalité, reste très pertinente pour résoudre la plupart des problèmes de simplification évoqués précédemment. La procédure d'optimisation sera conduite localement, pour une recherche d'atteignabilité sur une paire de niveaux $(S_i, M_i, \mathcal{A}(M_{i-1}))$. Deux critères sont utilisés :

- la longueur $|\Gamma_i|$ de la trace Γ_i vérifiant la propriété d'atteignabilité ; selon l'outil de model-checking utilisé, l'obtention de cette valeur peut nécessiter d'incrémenter un compteur, modélisé sous la forme d'une variable globale commune à l'ensemble des modèles, à chaque franchissement de transition,
- le temps d'horloge écoulé pour parcourir l'ensemble de la trace Γ_i vérifiant la propriété d'atteignabilité ; cette valeur peut être facilement obtenue en déclarant une horloge globale CLK, commune à l'ensemble des modèles, qui n'est jamais remise à zéro ; ainsi, en fin de parcours de la trace, la valeur de cette horloge globale fournit son temps de parcours.

Cette approche d'optimisation est mise en œuvre par l'ALGORITHME 7 qui intègre les deux critères relatifs à la longueur et au temps. Il présente une phase d'initialisation où une première recherche d'atteignabilité est effectuée sans considération des bornes afin de fixer leur valeur initiale. La deuxième phase concerne les boucles d'itérations effectuées en décrémentant les bornes (de longueur ou de temps) tant qu'il existe une trace vérifiant la propriété d'atteignabilité.

Ces procédures d'optimisation locales par paires de niveaux ne permettent pas de garantir un optimum global (la somme des optimum locaux n'est égal à l'optimum global). En effet, une trace Γ_i peut parfaitement optimiser l'utilisation d'éléments de niveaux i et i_1 mais ne peut pas considérer les éléments de niveau $i-2$ sollicités dans la trace Γ_{i-1}. Or ceux-ci peuvent remettre en cause les choix initiaux à la fois sur les critères de longueur et de temps.

Ce constat doit néanmoins être relativisé dans la mesure où ces optimisations locales permettent d'obtenir des séquences pertinentes dans la plupart des cas. A titre d'exemple, considérons l'exemple "jouet" de ce chapitre avec un objectif de conduite conduisant d'un mode *arrêt* à un mode *remplissage*. D'après le modèle de recette, cet objectif nécessite deux changements d'état : du mode *arrêt* au mode *refroidissement* puis du mode *refroidissement* au mode remplissage. De plus, le mode *refroidissement* impose l'usage de la fonction F_2, elle-même incompatible avec la fonction F_1. ll est donc clair que la séquence consistant à démarrer F_2 pour atteindre le mode *refroidissement* puis la fonction F_3 pour atteindre le mode *remplissage* sera préférable à celle consistant à démarrer F_1 pour atteindre le mode *refroidissement* puis à arrêter F_1 et démarrer F_2 et F_3 pour atteindre le mode de *remplissage*. Cette optimisation est prise en charge par notre algorithme.

Néanmoins, face à la complexité inhérente à l'obtention d'une séquence globale optimisée, une alternative consiste à proposer un ensemble de séquences globales admissibles, de les classer selon des critères tels que la longueur ou le temps, et de laisser le soin aux experts en conduite de procéder à un choix parmi les séquences admissibles. Ce point fait l'objet de la section suivante.

Algorithme 7 Optimisation locale par paire de niveau $(M_i, \mathcal{A}(M_{i-1}))$

Entrées :

Ensembles de modèles : $S_i, M_i, \mathcal{A}(M_{i-1})$

Propriété : $p = EF\ S_i.FinSequence$

Choix du type d'optimisation (temps, longueur) : *choix*

Sorties :

Trace d'exécution optimisée en temps : $\Gamma_i^{\triangleright}$

Séquence d'action optimisée en temps : $\Sigma_i^{\triangleright}$

Trace d'exécution optimisée en longueur : Γ_i°

Séquence d'action optimisée en longueur : Σ_i°

Si $(M_i \parallel \mathcal{A}(M_{i-1}) \parallel S_i) \models p$ **alors Retourner** Γ_i *% Initialisation des bornes %*

Fin SI

$\Sigma_i \leftarrow$ ALGORITHME_5(Γ_i)

$H \leftarrow CLK$; $L \leftarrow |\Sigma_i|$

Tant que $(M_i \parallel \mathcal{A}(M_{i-1}) \parallel S_i) \models p_{optim}$ **faire** *% Itérations de décrémentation des bornes %*

Si $choix = temps$ **alors**

Retourner $\Gamma_i^{\triangleright}$

$\Sigma_i^{\triangleright} \leftarrow$ ALGORITHME_5($\Gamma_i^{\triangleright}$)

$H \leftarrow H - 1$

$p_{optim} = p \wedge CLK < H$

sinon

Retourner Γ_i°

$\Sigma_i^{\circ} \leftarrow$ ALGORITHME_5(Γ_i°)

$L \leftarrow L - 1$

$p_{optim} = p \wedge |\Gamma_i| < L$

Fin Si

Fin Tant que

Retourner $\Gamma_i^{\triangleright}, \Sigma_i^{\triangleright}, \Sigma_i^{\circ}$

3.2.3 Génération et classification d'un ensemble de séquences

La génération d'un ensemble de séquences d'actions de conduite, et donc de traces d'exécution, peut s'opérer de deux manières distinctes :

- des éléments caractéristiques de la séquence obtenue (nombre de fois où un élément est sollicité, valeurs des variables physiques en fin de séquence, ...) sont introduits dans la propriété d'atteignabilité sous une forme complémentée (localité du modèle séquenceur à atteindre et pas éléments caractéristiques),
- certains model-checker offrent la possibilité de faire une exploration aléatoire de l'espace d'état ; dans ce cas, deux vérifications successives fourniront probablement des traces différentes même si les paramètres des modèles et de la propriété à vérifier demeurent identiques.

Dans notre cas, compte tenu du model-checker pour le développement du prototype présenté au chapitre suivant, la deuxième solution a été privilégiée. Le mode d'exploration aléatoire est

ainsi sélectionné pour réaliser plusieurs exécutions de l'ALGORITHME 4 qui fournissent différents ensembles de traces $\{\Gamma_2, ..., \Gamma_N\}$ où N est le nombre de niveaux. A partir de ces différents ensembles de traces, les ALGORITHMES 5 et 6 permettent d'obtenir différents ensembles de séquences d'actions $\{\Sigma_2, ..., \Sigma_N\}$ et enfin un ensemble de séquences globales d'actions de conduites Σ_G.

Une fois cet ensemble de séquences d'actions de conduite Σ_G généré, les séquences sont évaluées pour fournir à l'expert de conduite deux indicateurs permettant de guider son choix : longueur et temps de parcours d'une séquence Σ_G. L'obtention de la longueur d'une séquence Σ_G est triviale. En revanche, l'évaluation du temps de parcours d'une séquence Σ_G nécessite un traitement complémentaire formalisé par l'(ALGORITHME 8). En effet, le temps de parcours d'une trace Γ_i associée à une paire de niveau $(M_i, \mathcal{A}(M_{i-1}))$ contient des durées relatives aux équipements de niveau (i) mais aussi de niveau $(i-1)$. Par conséquent, la somme des temps de parcours sur les séquences $\Gamma_2, ..., \Gamma_N$ comptabilisera deux fois les durées relatives aux équipements de niveau i pour $2 < i \leq N$.

Algorithme 8 Temps de parcours global des séquences

Entrées :
Ensemble de traces : $\Gamma_2, ..., \Gamma_N$
Temps de parcours des traces : $CLK(\Gamma_i)$
Sorties :
Temps de parcours global CLK_G de la trace Γ_G et de la séquence globale Σ_G

$CLK_G \leftarrow CLK(\Gamma_2)$; $i \leftarrow 3$
Tant que $i \leq N$ **faire**
  $j \leftarrow 2$
  Tant que $j \leq |\Gamma_i|$ **faire** *% $|\Gamma_i|$ est le nombre d'états présents dans la trace Γ_i %*
    Pour tout $M_{i-1}^p \in M_{i-1}$ **faire** *% $M_{i-1}^p.clk(E_j)$ valeur de l'horloge du %*
      Si $M_{i-1}^p.clk(E_{j-1}) \neq M_{i-1}^p.clk(E_j)$ **alors** *% modèle M_{i-1}^p dans l'état E_j de Γ_i %*
        $CLK(\Gamma_i) \leftarrow CLK(\Gamma_i) - (M_{i-1}^p.clk(E_j) - M_{i-1}^p.clk(E_{j-1}))$
      Fin Si
    Fin Pour
    $j \leftarrow j + 1$
  Fin Tant que
  $i \leftarrow i + 1$
Fin Tant que
$CLK_G \leftarrow \sum_2^N CLK(\Gamma_i)$

Dans le cadre du projet CONNEXION, des travaux complémentaires [Bouaziz et al., 2015] [Bouaziz et al., 2016] ont proposé une approche plus globale d'aide à la sélection des séquences d'actions de conduite ([Gouyon et al., 2016]). Elle se base sur les indicateurs caractéristiques des séquences (longueur et temps de parcours) tels que nous les avons défini mais aussi sur des indicateurs correspondant à l'état courant de l'installation. Ces nouveaux indicateurs peuvent

se formaliser à partir du bilan de santé des différents équipements [Abichou, 2013] et tiennent compte de la performance, des dégradations observées et du contexte d'utilisation du composant. La proposition, positionnée dans le cadre de la décision multicritère et plus précisément dans celui de la théorie de l'utilité multi-attributs, permet l'obtention de ce classement en quatre phases FIGURE 3.9 :

- quantification des critères de décision afin d'établir le lien entre les critères de décision, les séquences d'actions et l'état des équipements ; pour chaque critère de décision étudié, une fonction de mesurage est établie ;
- calcul des utilités pour identifier les préférences des utilisateurs sous la forme d'une relation graduelle entre les valeurs numériques des critères et un référentiel d'utilité allant de zéro (pour les valeurs rejetées) à un (pour les valeurs préférées) ; ces fonctions permettent de rendre commensurable l'ensemble des critères ; elles peuvent être obtenues par apprentissage d'un historique ou encore à partir de connaissances métier ;
- agrégation des différentes informations en une valeur globale (score) sur laquelle repose la prise de décision à partir de situations souvent contradictoires ; l'opérateur d'agrégation est basé sur l'intégrale de Choquet [Grabisch et al., 2011] en prenant en compte l'importance de chaque critère ainsi que leurs interactions ;
- classement des scores du meilleur au moins bon.

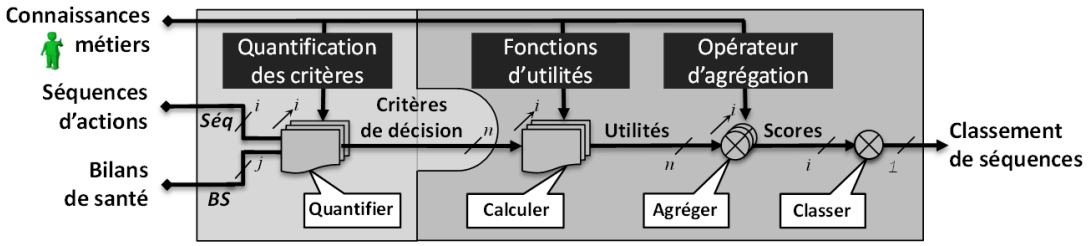


FIGURE 3.9 – Processus de classement de séquences

Ces travaux sont donc très complémentaires de l'approche de génération de séquences d'actions de conduite que nous avons développée puisqu'ils fournissent à l'expert de la conduite un ensemble d'informations consensuelles et objectives sur les différentes stratégies admissibles d'un point de vue fonctionnel mais aussi de la sécurité ainsi qu'une proposition de classement.

Conclusion

Dans ce chapitre, nous avons proposé une méthodologie complète de génération de séquences d'actions de conduite, applicable aux systèmes complexes critiques (FIGURE 3.10).

D'un côté, la méthodologie facilite et systématise le travail du modélisateur en uniformisant la structuration, la description et l'interfaçage des différentes représentations du procédé et des objectifs de conduite et en proposant des patrons de modélisation couvrant la plupart des cas rencontrés dans le cadre du projet CONNEXION.

D'un autre côté, elle permet la recherche de séquences d'actions de conduite sur des systèmes de grande taille en réduisant la taille de l'espace d'état considéré par la mise en œuvre d'une

démarche itérative de génération des séquences par paires de niveaux, en utilisant une approche itérative et en considérant des réductions de modèles par abstraction.

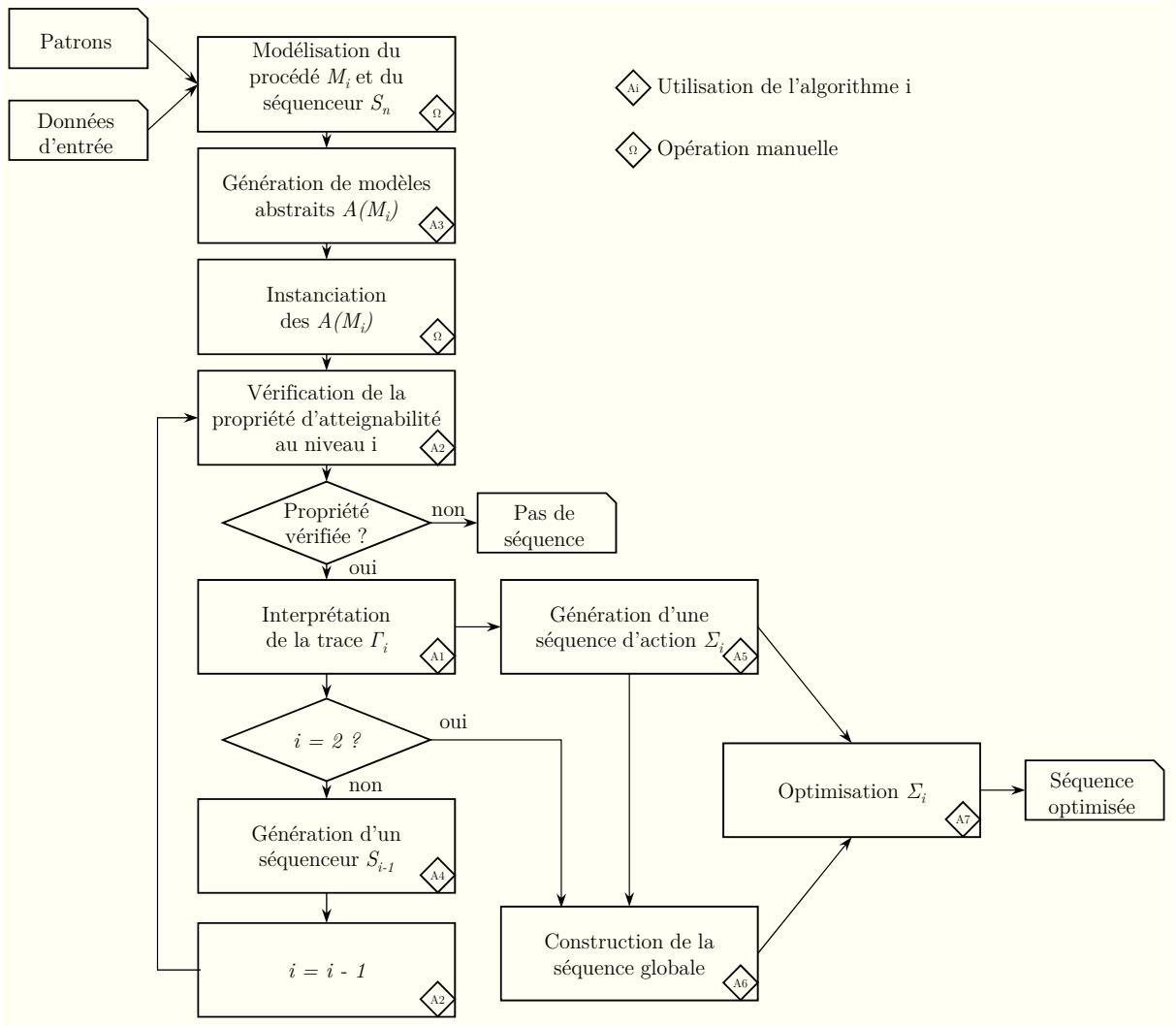


FIGURE 3.10 – Synthèse des étapes de l'approche de génération itérative de séquences d'actions de conduite

Le chapitre suivant a pour objectif, d'une part, de présenter un prototype logiciel implémentant la démarche proposée pour la génération des séquences de conduite, et d'autre part, d'illustrer les apports de cette méthodologie sur un cas d'étude académique (plate-forme expérimentale de laboratoire) et un cas d'étude à échelle industrielle proposé par le projet CONNEXION.

Chapitre 4

Mise en œuvre de l'approche de génération de séquences

Sommaire

Introduction

La mise en œuvre présentée dans ce chapitre repose sur le cadre formel présenté au Chapitre 2 et sur les algorithmes de génération itérative du Chapitre 3. Cette mise en œuvre de l'approche est expliquée sur le principe d'une structuration du procédé en trois niveaux hiérarchiques : M_1 - Équipements, M_2 - Fonctions, M_3 - Recette.

Dans un premier temps, ce chapitre présente la chaîne d'outils logiciels utilisés et/ou spécifiquement développés dans l'objectif de mettre en œuvre la méthodologie proposée.

Dans un second temps, cette approche outillée est appliquée sur deux cas d'étude, avec pour chacun un objectif différent :

- tout d'abord, la première application sur la plate-forme de laboratoire CISPI du CRAN a pour objectif d'illustrer l'approche sur un cas d'étude de dimension supérieure au cas "jouet" utilisé au chapitre précédent ;
- la seconde application, s'effectuant quant à elle dans le cadre de l'ingénierie d'un système de dimension industrielle extrait du projet CONNEXION, a pour objectif de générer de nouvelles procédures de conduite prenant en compte des modifications apportées au contrôle/commande.

Pour finir, ce chapitre évalue l'approche proposée en termes de modélisation et de passage à l'échelle, en se focalisant en particulier sur la taille de l'espace d'état parcouru et le temps nécessaire à la génération itérative de séquences d'actions de conduite.

4.1 Outillage

Comme cela a été décrit aux chapitres 2 et 3, l'approche de génération itérative de séquences d'actions de conduite proposée repose sur une modélisation du procédé sous la forme d'un réseau d'automates temporisés et sur des séquences générées formellement à l'aide de mécanismes de recherche d'atteignabilité. Parmi les outils de model-checking permettant la recherche d'atteignabilité sur des réseaux d'automates temporisés, UppAal a été choisi pour outiller cette approche. Il est complété par un ensemble de scripts et par un outil d'analyse de séquences. Ces différents éléments ont permis de créer une chaîne outillée pour la génération de séquences d'actions de conduite, représentée par la Figure 4.1. Ils sont présentés dans cette section.

4.1.1 Un outil de modélisation et de model checking : UppAal

UppAal [Behrmann et al., 2006b] est un logiciel développé en collaboration entre le département d'Information Technology de l'université d'Uppsala en Suède et le département de Computer Science de l'université D'Aalborg au Danemark. Il s'agit d'un outil de modélisation utilisant les réseaux d'automates temporisés, et disposant d'une part de fonctionnalités de model checking permettant la vérification formelle de propriétés formalisées en langage CTL* (sous-ensemble du langage CTL), et d'autre part d'une interface graphique permettant la saisie et l'instanciation des modèles. Lors de la vérification des propriétés, UppAal permet d'obtenir des traces d'exécutions issues de l'exploration des modèles, correspondant à des exemples (resp. des contres-exemples) lors de la vérification de propriétés d'atteignabilité (resp. de sécurité).

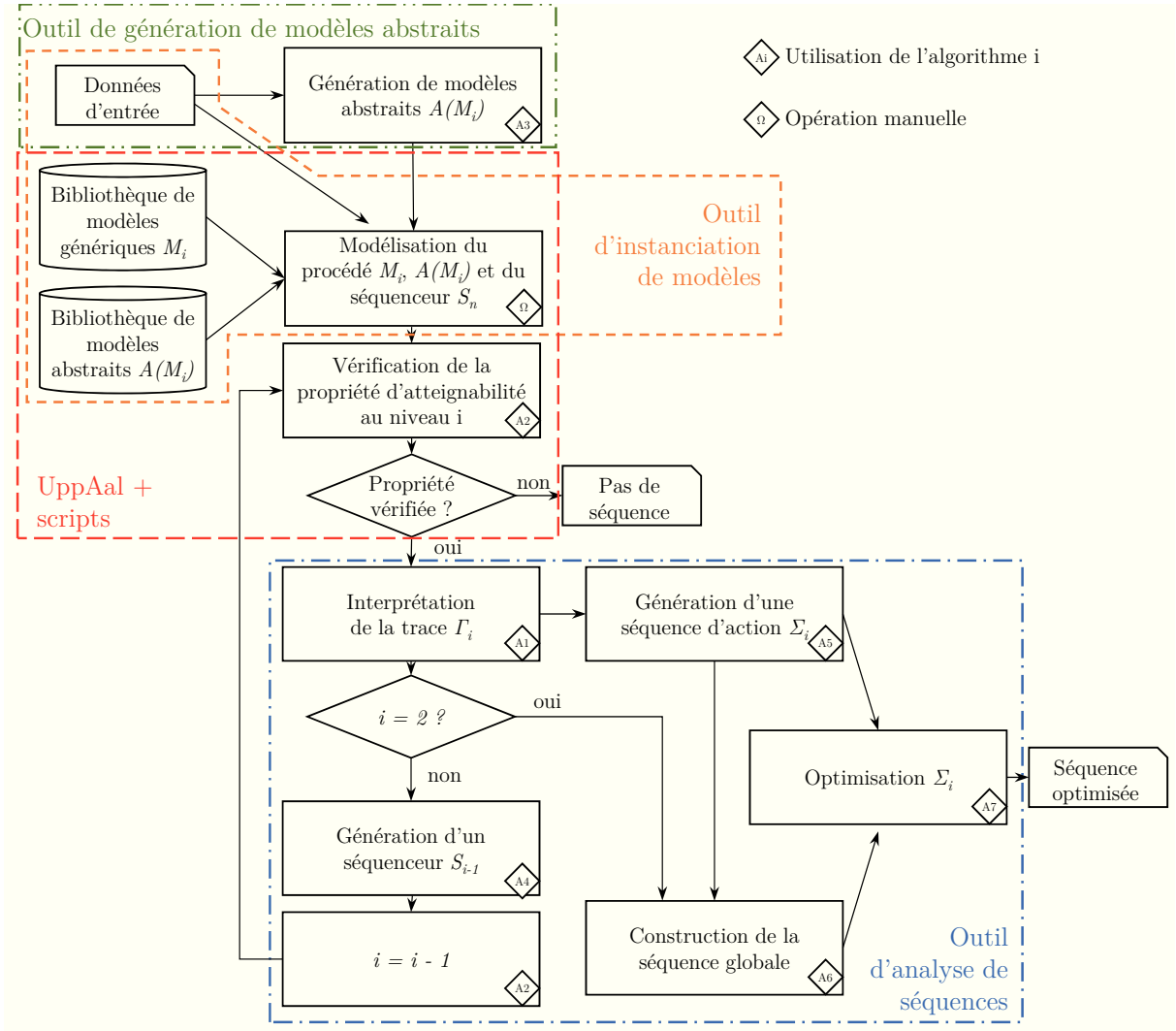


FIGURE 4.1 – Vue globale de la méthodologie proposée et de son outillage

La suite de cette section 4.1.1 fait le parallèle entre, d'une part, les modèles exprimés à l'aide de réseaux d'automates temporisés aux chapitres 2 et 3 et, d'autre part, leur implantation telle qu'elle peut être réalisée sous UppAal.

Localités et transitions

Dans les automates temporisés, tout comme avec UppAal, une localité est représentée par un disque, et les transitions sont représentées par des flèches. Une localité initiale d'un automate temporisé est représentée par une flèche pointant vers la localité ; sous UppAal, une localité initiale est représentée avec deux disques concentriques.

De même que pour les automates temporisés, les localités sont caractérisées dans UppAal par un nom, et il est possible de leur associer un invariant (FIGURE 4.2) :

– le nom d'une localité est indiqué dans le cercle la représentant dans les automates temporisés (voir Chapitre 2 et Chapitre 3), et en dessous de la localité en violet foncé sur un modèle UppAal ;

– un invariant d'horloge est indiqué entre crochets sur les automates temporisés, et en violet clair sur un modèle UppAal ;

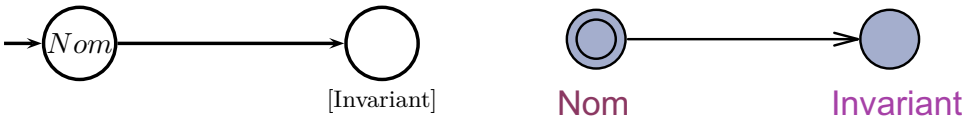


FIGURE 4.2 – Correspondance entre le modèle automate temporisé (à gauche), et son implantation UppAal (à droite)

Dans les automates temporisés, de même que dans UppAal, le franchissement d'une transition d'une localité à une autre est relatif à l'occurrence d'un label, et conditionné par une garde sur les valeurs de variables ou d'horloges. Il est également possible d'associer des actions aux franchissements des transitions, comme la mise à jour de valeurs de variables ou d'horloges. D'un point de vue de la représentation graphique (FIGURE 4.3) :

– une garde est indiquée entre crochets sur un automate temporisé ; elle est indiquée en vert sur un modèle UppAal ;
– un label est indiqué en italique sur un automate temporisé, et suivi d'un point d'exclamation dans le cas d'une émission ou d'un point d'interrogation dans le cas d'une réception ; il est indiqué en bleu clair sur un modèle UppAal ;
– une mise à jour est indiquée en italique sur un automate temporisé ; elle est indiquée en bleu foncé sur un modèle UppAal.

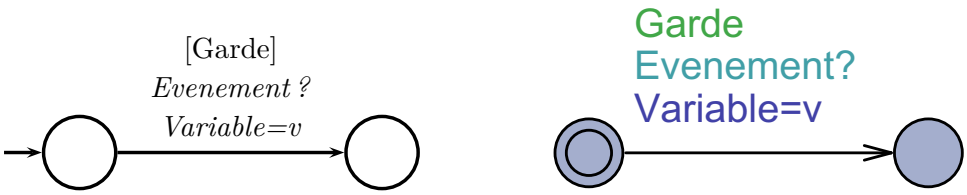


FIGURE 4.3 – Parallèle sur la représentation des transitions : à gauche, le modèle automate temporisé, à droite, l'implantation UppAal correspondante

Expression des propriétés à vérifier

Le model checker d'UppAal permet la vérification formelle de propriétés exprimées en CTL*, dans une syntaxe qui lui est propre [Behrmann et al., 2006a]. Par exemple, une propriété exprimant l'atteignabilité d'une situation dans laquelle la localité *FinSequence* du modèle *Sequenceur* est active est exprimée $EF Sequenceur.FinSequence$ en CTL*, et *E <> Sequenceur.FinSequence* sous UppAal.

4.1.2 Création de modèles génériques avec les templates UppAal

L'outil UppAal permet la création de « templates » instanciables, qui peuvent être utilisés comme modèles génériques. Leur utilisation nécessite de manière complémentaire la déclaration de variables et de fonctions qui seront utilisées lors de l'instanciation.

4.1.2.1 Modèle générique des « Équipements » sous UppAal

Le premier niveau du procédé est constitué des équipements, dont la structure générique des modèles M_1 est donnée en FIGURE 2.12. L'implantation de cette structure avec UppAal est présentée dans la FIGURE 4.4.

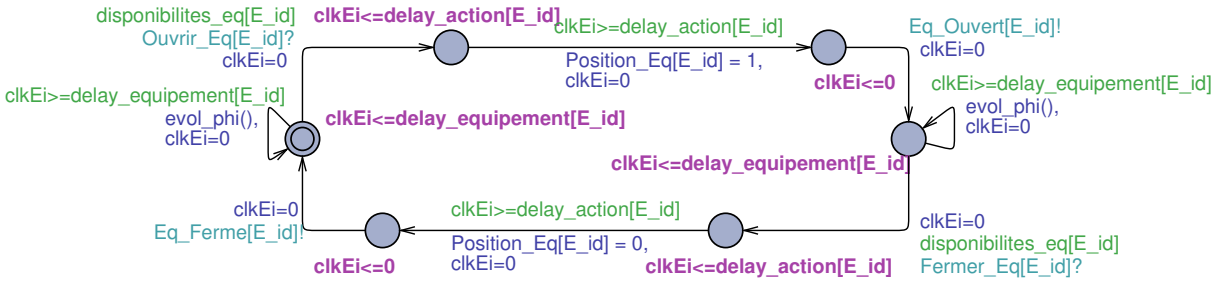


FIGURE 4.4 – Template UppAal M_1 pour la modélisation des équipements

Ce template étant par définition un modèle générique instanciable, le nombre entier E_id est utilisé comme identifiant de l'équipement considéré. Il est utilisé dans les fonctions UppAal pour instancier les valeurs relatives à cet équipement, précisées dans les données d'entrée (voir SECTION 4.1.3.1).

4.1.2.2 Modèle générique des « Fonctions » sous UppAal

Le second niveau du procédé M_2 est constitué des fonctions, qui seront considérées avec deux états stables ($configurée$ et $nonconfigurée$). La structure générique des modèles M_2 est donc dérivée de celle donnée en FIGURE 2.14 pour le niveau M_n. L'implantation de cette structure avec UppAal est présentée dans la FIGURE 4.5. Le nombre entier F_id est utilisé comme identifiant de la fonction considérée.

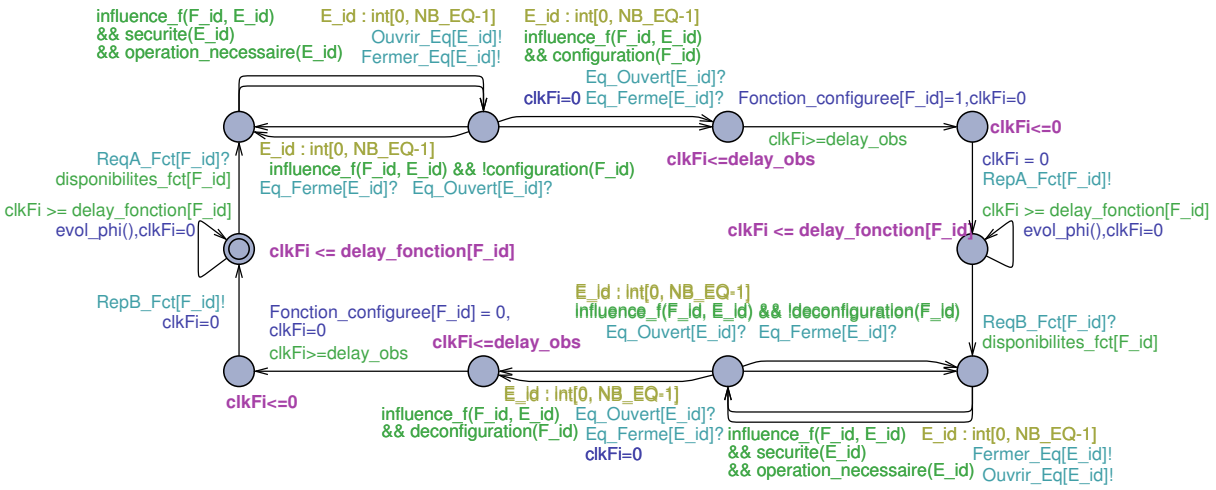


FIGURE 4.5 – Template UppAal M_2 pour la modélisation des fonctions

La réception d'un label de synchronisation Req_Fct permet d'enclencher le processus de configuration de la fonction considérée. L'émission du label de synchronisation Rep_Fct permet de signaler que la fonction d'identifiant F_id est configurée.

En complément de cette structure, différents éléments sont déclarés dans le template :

– la fonction $influence\_f$ est une fonction dont le rôle est de déterminer si un équipement E_id appartient au cône d'influence $\mathcal{C}(M_i^k)$ d'une fonction F_id. Cette fonction renvoie la valeur *true* lorsque c'est le cas ;

Code

```
//Fonction influence_f
bool influence_f(int F_id, int E_id) {
   return cone_influence_fonction_equipements[F_id][E_id];
}
```

– la fonction *securite* est une fonction dont le rôle est d'assurer le respect de différentes conditions de sécurité, comme des conditions de type « précédence » (actionnement d'un équipement en fonction de l'état d'un autre équipement ou d'une variable physique). Cette fonction renvoie la valeur *true* lorsque les conditions de sécurité G_s^p sont réunies ;

Code

```
//Fonction securite
bool securite(int E_id) {
   int i = 0, j = 0;
   for (i = 0; i < NB_COND_OUV; i++) {
      if (conditions_securite_ouverture[i][0] == E_id) {
         for (j = 1; j < NB_EQ+1; j++) {
            if (conditions_securite_ouverture[i][j] == 1 &&
Position_Eq[j-1] != 1) {
               return false;
            }
         }
         return true;
      }
   }
   return true;
}
```

– la fonction $configuration$ a pour rôle de déterminer si la fonction est configurée ou non (G_f^p). Pour cela, elle fait appel à la matrice d'entrée $configurations\_fonctions$. Elle renvoie la valeur *true* lorsque la fonction est configurée.

Code

```
//Configurations des équipements pour la fonction
bool configuration(int id) {
   int i = 0;
   for (i = 0; i < NB_EQ; i++)
      if (cone_influence_fonction_equipements[F_id][i])
```

```
         if (configurations_fonctions[id][i] != Position_Eq[i])
            return false;
   return true;
}
```

4.1.2.3 Modélisation de la « Recette » sous UppAal

Le plus haut niveau du procédé (M_3) est constitué de la recette, dont la structure générique est dérivée de celle donnée en FIGURE 2.14 pour le niveau M_n. Une implantation partielle de cette structure avec UppAal est présentée dans la FIGURE 4.5, dans laquelle deux situations stables (Sit_A et Sit_B) sont considérées.

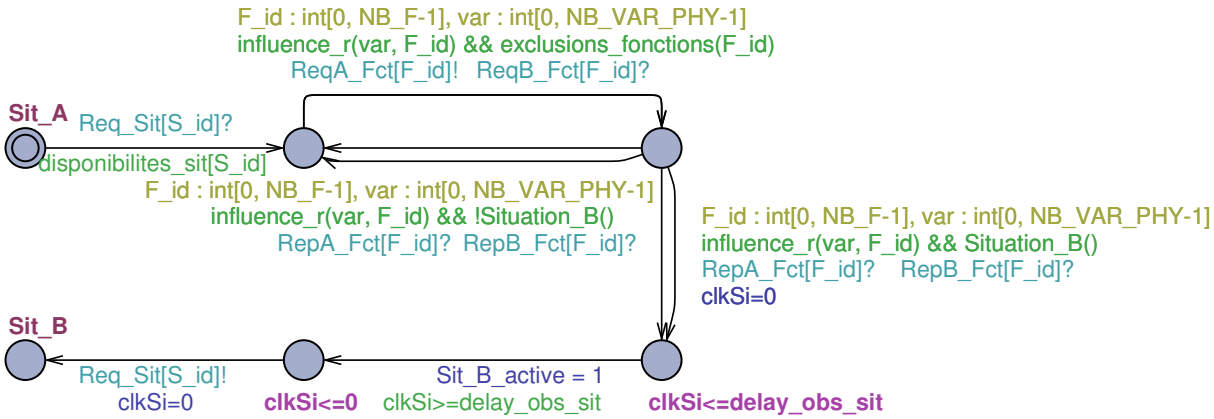


FIGURE 4.6 – Extrait du template UppAal M_3 pour la modélisation d'une recette

Le label de synchronisation *Req_Sit_B* correspond à une demande de changement de situation courante émise par le séquenceur. Le label de synchronisation *Rep_Sit_B* signale l'atteinte de la situation B au modèle de séquenceur.

En complément de cette structure, différentes fonctions sont déclarées dans le template :

- la fonction $influence\_r$ a pour rôle de déterminer si une fonction F_id est utile à la variation d'une variable physique var. Cette fonction renvoie la valeur $true$ lorsque c'est le cas. Sinon, la garde associée est $false$, ce qui empêche l'émission d'une requête Req_Fct correspondant à une fonction F_id qui n'est pas dans le cône d'influence $C(M_3^k)$;

Code

```
//Fonction influence
bool influence_r(int var, int F_id) {
   bool g = false;
   if (VAR_PHY_INT[var])
      if (cone_influence_variables_fonctions[var][F_id] != 0)
         g = true;
   return g;
}
```

– la fonction *exclusions_fonctions* a pour rôle de déterminer si une fonction *F_id* est compatible avec les fonctions actuellement configurées, afin d'empêcher de demander la configuration de deux fonctions qui ne peuvent pas s'exécuter en même temps (garde G_s^p). Elle se base pour cela sur la matrice *mat_exclusions_fonctions*. La fonction renvoie la valeur *true* lorsque qu'il n'y a pas d'incompatibilité ;

Code

```
//Fonction exclusions
bool exclusions_fonctions(int F_id) {
   int i;
   for (i = 0; i < NB_F-1; i++) {
      if (mat_exclusions_fonctions[F_id][i] == 1)
         return !Fonction_configuree[i];
   }
   return true;
}
```

– la fonction *Situation_B* a pour rôle de déterminer si la situation objectif est atteinte. La fonction renvoie la valeur *true* lorsque la situation souhaitée est atteinte.

Code

```
//Fonction situation_B : détermine si la situation est atteinte
bool Situation_B() {
   int i;
   for (i = 0; i < NB_F-1; i++)
      if (Fonction_configuree[i]) return true;
   return false
}
```

4.1.2.4 Séquenceur initial

La structure du modèle S_n du séquenceur de niveau n est dérivée de celle donnée en Figure 2.8. L'implantation de cette structure avec UppAal est présentée dans la Figure 4.7. Les séquenceurs de niveau i $\neq n$ sont générés automatiquement selon de principe décrit dans la Section 3.1.5. Pour des soucis de généricité, la localité de l'automate représentant la situation que l'on souhaite atteindre est nommée « FinSequence », comme elle l'est pour les automates générés. Cela permettra d'utiliser une propriété unique à vérifier à chaque niveau d'exploration.

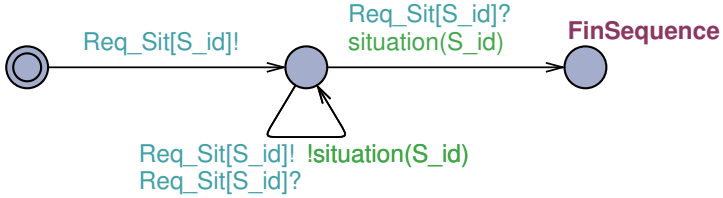


Figure 4.7 – Template UppAal pour le séquenceur initial S_n

4.1.3 Utilisation des templates pour la création de modèles UppAal

Cette section présente les déclarations relatives aux données d'entrée de la FIGURE 4.1, et leur utilisation pour la création de modèles UppAal par instanciation de templates.

4.1.3.1 Déclarations des données d'entrée utilisées pour l'instanciation des templates

La rubrique « Declaration » d'UppAal permet la déclaration des variables globales partagées par différents modèles et utilisées pour les calculs de gardes et la mise à jour de données, des canaux de synchronisation *chan*, et des fonctions. La déclaration des variables et des fonctions dans l'outil UppAal est fortement similaire à la déclaration de variables en langage C ; les différents types de variables qu'il est possible d'utiliser sont des nombres entiers, des booléens, des matrices et des structures.

Déclaration des variables V, des horloges X et du nombre de modèles M_i^k.

Des entiers constants, respectivement notées N_e, N_f, N_ϕ, N_{co}, N_{cf} au long du chapitre, sont définies. Ils représentent respectivement le nombre d'équipements, de fonctions, de variables physiques, de conditions d'ouverture et de fermeture considérées dans les modèles. Ces constantes seront respectivement notées NB_EQ, NB_F, NB_VAR_PHY, NB_COND_OUV et NB_COND_FER sous UppAal. Elles permettent d'indiquer les dimensions des matrices utilisées (présentées ci-après).

Un tableau VAR_PHY_INT est également créé afin de représenter les cônes d'influence. Il permet d'indiquer par une valeur booléenne l'intérêt ou non de considérer une variable pour l'atteinte d'un objectif de conduite.

Code

```
//Déclaration des constantes
const int NB_EQ = 23; //Nombre d'équipements
const int NB_F = 24; //Nombre de fonctions
const int NB_VAR_PHY = 3; //Nombre de variables physiques
const bool VAR_PHY_INT[NB_VAR_PHY] = {-1, 0, 1}; //Cône d'influence des
variables physiques
const int NB_COND_OUV = 10; //Nombre de conditions d'ouverture
const int NB_COND_FER = 4; //Nombre de conditions de fermeture
```

La matrice *variables_physiques*, de taille N_ϕ, contient les valeurs des différentes variables physiques. Celle-ci stocke à chaque instant t la valeur courante des variables physiques et est amenée à évoluer tout au long de l'exploration du modèle. Sa déclaration contient les valeurs initiales des variables physiques.

Code

```
//Déclaration de la matrice contenant les valeurs des variables physiques
int variables_physiques[NB_VAR_PHY] = {50, 50, 50};
```

Configuration des fonctions G_f.

La matrice $configurations\_fonctions$, de taille $N_f \times N_e$, associe à chaque fonction sa configuration, c'est-à-dire l'état attendu des équipements pour que la fonction fasse évoluer les variables physiques. Les valeurs attendues (vanne ouverte/fermée, pompe enclenchée, déclenchée) sont codées par des valeurs booléennes ($false$ = vanne fermée ou pompe déclenchée, $true$ = vanne ouverte ou pompe enclenchée).

Code

```
//Déclaration des configurations des équipements pour les fonctions
bool configurations_fonctions[NB_F][NB_EQ] = {
{0, 1, ..., 0},
...
{0, 0, ..., 1} };
```

Cônes d'influence $\mathcal{C}(M_i^k)$.

La matrice $cone\_influence\_fonction\_equipements$, de taille $N_f \times N_e$, associe à chaque fonction un ensemble d'équipements d'intérêt pour sa configuration, *via* des valeurs booléennes ($true$ ou $false$ détermine l'intérêt ou non d'un équipement pour une fonction).

Code

```
// Déclaration des cônes d'influence pour les fonctions
bool cone_influence_fonction_equipements[NB_F][NB_EQ] = {
{1, 1, ..., 0},
...
{0, 0, ..., 1} };
```

La matrice $cone\_influence\_variables\_fonctions$, de taille $N_\phi \times N_f$, associe à chaque variable physique un ensemble de fonctions d'intérêt pour son évolution ainsi que l'incrément relatif. En d'autres termes, une valeur 0 montre que la fonction n'a pas d'influence sur la valeur de la variable physique, une valeur négative (resp. positive) fait diminuer (resp.augmenter) la valeur de la variable. Plus la valeur est grande, plus la variation est importante.

Code

```
// Déclaration des cônes d'influence pour les variables physiques
int cone_influence_variables_fonctions[NB_VAR_PHY][NB_F] = {
{-1, -2, ..., 1},
...
{0, 1, ..., 1} };
```

Contraintes de sécurité G_s.

Les matrices $conditions\_securite\_ouverture$ et $conditions\_securite\_fermeture$, de taille $N_e \times N_e$, donnent les informations de sécurité à respecter, en terme de précédence (actionnement d'un équipement selon l'état d'un autre équipement).

Code

```
//Déclaration des contraintes de sécurité
int conditions_securite_ouverture[NB_COND_OUV][NB_EQ+1] = {
{14, 1, ..., 0},
...
{32, 0, ..., 0} };

int conditions_securite_fermeture[NB_COND_OUV][NB_EQ+1] = {
{14, 1, ..., 0},
...
{32, 0, ..., 0} };
```

Temps d'évolution des variables physiques Φ.

La matrice $delay\_fonction$, de taille N_f, contient les délais relatifs à l'évolution des variables physiques des différentes fonctions.

Code

```
//Déclaration des temps d'évolution des variables physiques
int delay_fonction[NB_F] = {10, 5, ..., 10};
```

Disponibilités G_d des Equipements M_i^k.

La matrice $disponibilites\_eq$ est une matrice de taille N_e, contenant l'information booléenne de disponibilité des équipements. Les éléments de cette matrice sont utilisés dans les gardes G_d, pour empêcher l'actionnement d'un équipement si celui-ci est condamné, afin de respecter la contrainte de sécurité induite.

Code

```
//Déclaration des disponibilités des équipements
bool disponibilites_eq[NB_EQ] = {1, 1, ..., 1};
```

Etat courant des équipements M_i^k La matrice $Position\_Eq$ est une matrice de taille N_e, contenant l'information d'état courant des équipements, avec E_id le nombre entier permettant l'association à un équipement E_id particulier.

Code

```
//Déclaration des états courants des équipements
bool Position_Eq[NB_EQ] = {0, 0, ..., 0};
```

Canaux de synchronisations Req_M_i et Req_M_i

La matrice Req_Eq (respectivement Rep_Eq) est une matrice de taille N_e. La réception d'un label de synchronisation Req_Eq (respectivement Rep_Eq) permet l'actionnement d'un

équipement *E_id* afin d'enclencher le processus d'ouverture (respectivement de fermeture) de cet équipement.

4.1.3.2 Création du modèle UppAal par instanciation des templates

Les différents modèles de procédé utilisés dans l'outil UppAal peuvent être créés par instanciation de templates génériques (M_i ou $\mathcal{A}(M_i)$) présentés dans la Section 4.1.2. La structure de ces modèles peut correspondre aux modèles génériques décrits au chapitre 3, ou être faite à la main par un modélisateur.

La rubrique « System declaration » de l'outil UppAal permet, d'une part, de faire l'instanciation des modèles génériques *via* la déclaration des instances, et d'autre part de déclarer quelles seront les instances considérées dans le système. Un exemple d'instanciation est donné en Section 4.2.

4.1.4 Outils de création de modèles et d'analyse de séquences

Cette section présente les différents outils qui ont été développés dans le cadre de cette thèse, en complément de l'outil de vérification formelle UppAal (Figure 4.1). Leur objectif est de faciliter d'une part la création des modèles UppAal, et d'autre part l'analyse des séquences produites.

4.1.4.1 Outil d'instanciation de modèles

L'objectif de cet outil, développé en C#, est de permettre la génération automatique des modèles M_i du procédé à conduire, dans un format UppAal. Pour cela, il utilise les données d'entrées relatives au procédé à modéliser, et instancie automatiquement les modèles génériques et abstraits disponibles sous la forme de templates UppAal en bibliothèque.

D'un point de vue technique, l'outil UppAal stocke ces modèles dans un fichier au format .xml. Ce fichier comprend la description des différents templates, fonctions et déclarations, nécessaires. Ainsi, chaque template est décrit en termes de localités, transitions et déclarations (canaux, paramètres...). A titre d'exemple, chaque localité possède :

- Obligatoirement
  - Un ID unique
  - Des coordonnées X et Y (utiles à la représentation graphique)
- Eventuellement :
  - Un nom
  - Un invariant

De même, chaque transition possède :

- Obligatoirement
  - L'identifiant d'une place en qualité de source de la transition
  - L'identifiant d'une place en qualité de destination de la transition
- Eventuellement :
  - Une garde
  - Un canal de synchronisation
  - Des mises à jour sur une/des variable(s)

L'outil d'instanciation de modèles instancie les templates et crée un nouveau fichier .xml contenant le modèle du procédé utilisable pour la génération de séquences.

4.1.4.2 Outil de génération de modèles abstraits

L'approche itérative de génération de séquences présentée au Chapitre 3 implique la création et l'utilisation de modèles abstraits. Cette abstraction peut se faire, selon que l'on utilise des modèles génériques ou non, sur un élément en bibliothèque ou sur un modèle fait à la main par le modélisateur.

L'objectif de l'outil de génération de modèles abstraits, développé en C#, est de faciliter la création d'un modèle abstrait au format UppAal. Il est basé sur l'implantation de l'algorithme 2 présenté au Chapitre 3. Il utilise en entrée le fichier .xml d'UppAal contenant le modèle à abstraire, et génère un fichier .xml contenant l'abstraction.

A titre d'exemple, l'abstraction du modèle générique de fonction présenté à la FIGURE 4.5 est donné ci-dessous à la FIGURE 4.8.

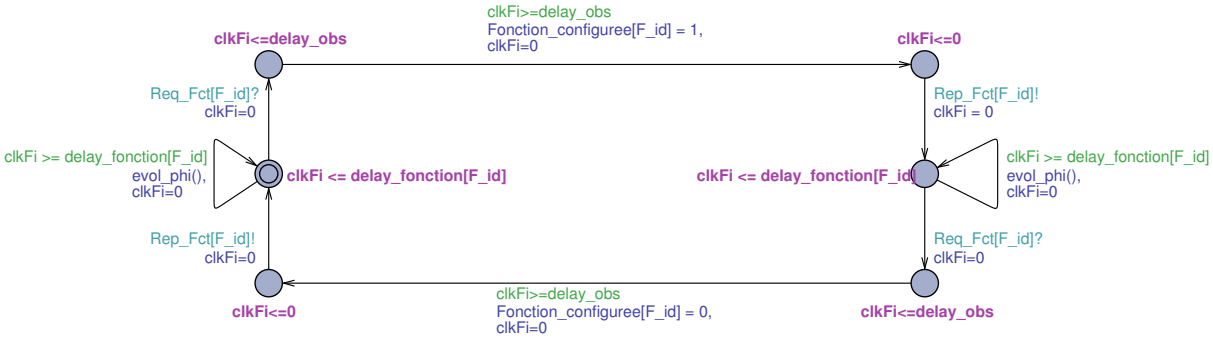


FIGURE 4.8 – Template UppAal $\mathcal{A}(M_2^k)$ pour la modélisation abstraite d'une fonction

4.1.4.3 Outil d'analyse de séquences

Sous UppAal il est possible de lancer des vérifications de plusieurs manières :

- Soit en passant par l'interface graphique et en demandant la vérification d'une propriété ;
- Soit en appelant le model checker directement en ligne de commande avec des instructions Dos (sous Windows).

Lorsque la propriété est vérifiée, UppAal est capable de retourner une trace prouvant le respect de la propriété.

- Dans le premier cas, avec l'utilisation de l'interface graphique d'UppAal, la trace peut être simulée pas à pas, afin de visualiser les différentes actions de conduite à nécessaires. Il est également possible de sauvegarder cette trace dans un fichier au format .xtr. Cependant, ce fichier n'est pas facilement exploitable car il ne comprend pas d'information "en clair", mais uniquement des suites de nombres ;
- Dans le second cas, en passant par un appel direct du model checker en ligne de commande, le fichier donnant la trace est dans un format .txt qui est plus facilement interprétable. Il se compose en effet d'une séquence d'états (donnant en clair les localités actives de tous les automates et les valeurs de toutes les variables) et de transitions (précisant en clair quelles transitions sont franchies, avec les canaux et actions associés).

L'utilisation en ligne de commande d'UppAal permettant de générer des traces plus facilement interprétables, elle a été choisie. Néanmoins, les fichiers étant de taille importante, un outil d'analyse automatique de séquences a été développé. Il s'agit d'une application c# capable d'extraire du fichier de trace les informations nécessaires à la construction d'une séquence d'actions de conduite utilisable, sous la forme d'une succession de dates associées à des actions à effectuer sur des équipements. Cette séquence est donnée sous la forme d'un fichier .csv.

4.2 Cas d'étude #1 : Plate-forme CISPI du CRAN

L'objectif, sur ce cas d'étude, est d'éprouver l'applicabilité de la méthodologie de génération itérative de séquences d'actions de conduite proposée au chapitre précédent, sur un exemple de taille plus importante que le cas "jouet" des chapitres 2 et 3.

4.2.1 Présentation de la plateforme CISPI

Le CRAN développe depuis plusieurs années au sein de son service plates-formes expérimentales un centre d'innovation et de démonstration des technologies sûres de fonctionnement nommé SAFETECH. Celui-ci est composé de plusieurs plates-formes et plateaux techniques permettant d'illustrer des résultats scientifiques. La plate-forme CISPI (Conduite Interactive et Sûre des Procédés Industriels), représentative de la variété des modes opératoires d'une tranche de centrale de production d'électricité, est une partie intégrante de SAFETECH [Clanché et al., 2010].

La plate-forme CISPI s'intéresse à la conduite des procédés industriels, qui met en jeu un ensemble de processus complexes couvrant des modes opératoires variés (en production, en arrêt, en démarrage, ...), adaptés à la criticité des modes d'exploitation rencontrés (conduite normale, conduite incidentelle et accidentelle), selon des constantes de temps différentes (conduite en temps réel, maintenance hors ligne, ...). Aujourd'hui, ces processus reposent sur des interactions entre les différents métiers des opérateurs et des systèmes propriétaires, hétérogènes et généralement limités à la phase de production normale.

L'objectif du démonstrateur CISPI est, d'une part, d'illustrer les travaux de recherche en IAM (Intelligent Actuation and Measurement) [Pétin et al., 1998, Dobre, 2010] et, d'autre part, d'expérimenter de nouvelles formes d'organisation de la commande et de la conduite de procédés industriels exploitant au mieux les capacités de stockage, de traitement et de communication de l'information de nouvelles technologies ambiantes, pour favoriser une sécurité active (interactions homme/système, produits/système) des procédés [Dobre et al., 2010]. CISPI couvre également les besoins de validation expérimentale de travaux actuels portant sur l'ingénierie système basée sur les modèles [Dobre, 2010, Bouffaron, 2016].

L'un des intérêts du démonstrateur CISPI est de présenter un nombre important de redondances matérielles (Figure 4.9). Ceci permet d'offrir à un opérateur, pour un même objectif de conduite, des modes de conduite et un ensemble d'opérations variés. Les lignages possibles reposent sur des combinaisons de configurations des 22 équipements composant le système.

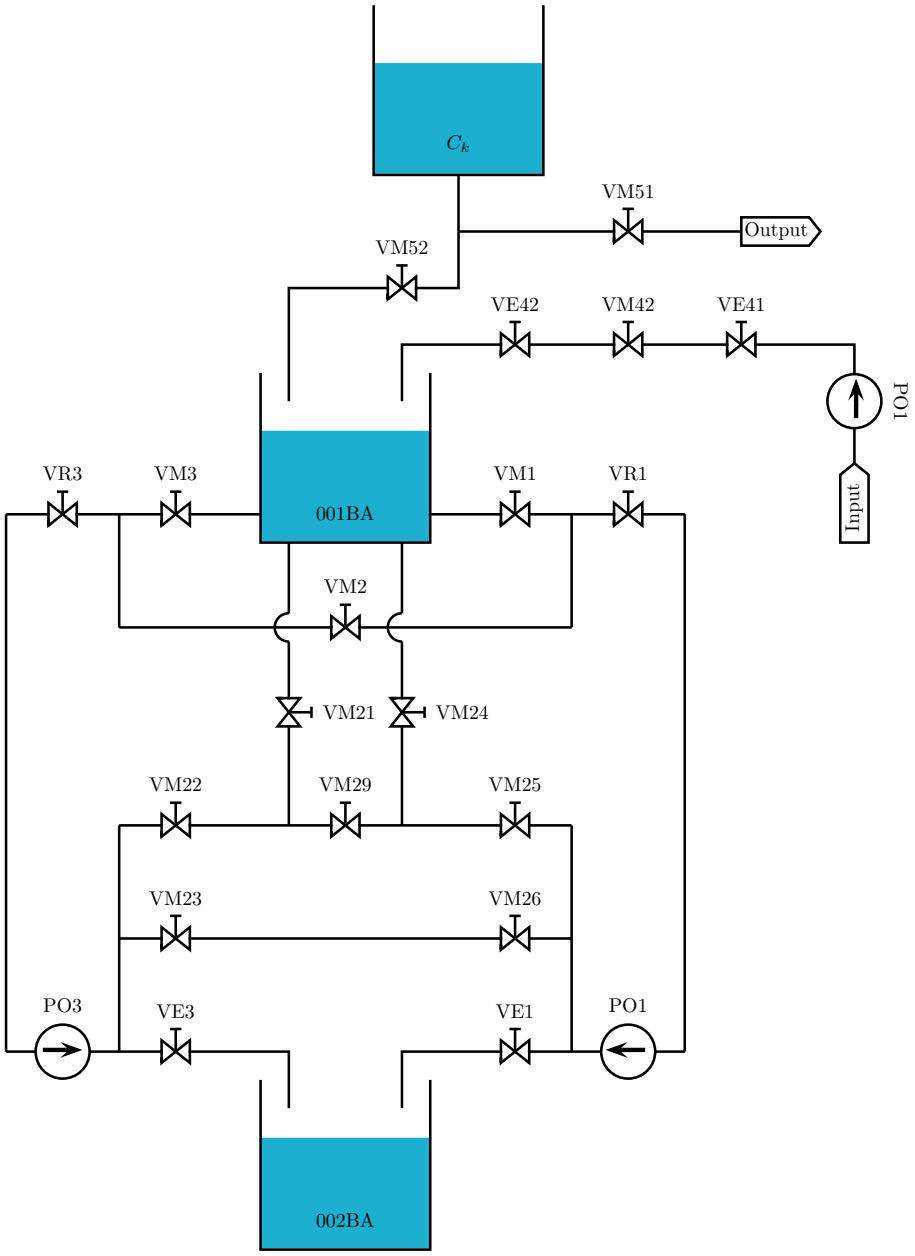


FIGURE 4.9 – Plan de circulation des fluides partiel de la plate-forme CISPI

4.2.2 Génération d'une séquence d'actions : alimentation de la bâche 002BA

4.2.2.1 Présentation du procédé CISPI d'alimentation en eau de la bâche 002BA

Initialement, la plate-forme est considérée dans un état dans lequel tous les équipements sont en position *fermé* (pour les vannes) ou *arrêté* (pour les pompes). De plus, on considère les bâches C_k et $001BA$ remplies, et la bâche $002BA$ vide. On souhaite conduire la plate-forme CISPI de manière à alimenter en eau la bâche $002BA$ à partir de la bâche $001BA$. Une analyse des circulations de fluide possibles a permis de déterminer 24 configurations différentes, dont 20

peuvent être utiles à remplir la bâche 002BA [10] :

- Vidange de la bâche C_k
  - CkVersCP : *via* VM52
  - CkVersOut1 : *via* VM51
  - CkVersOut1EtCp : *via* VM51 et VM52
- Remplissage de la bâche 001BA
  - InVersCp : *via* PO1, VE41, VM42, VE42
- Remplissage de la bâche 002BA
  - CpCsExt1G : *via* VM3, VR3, PO3 et VE3
  - CpCsExt1D : *via* VM1, VR1, PO1 et VE1
  - CpCsExt1G_Vm2 : *via* VM3, VM2, VR1, PO1 et VE1
  - CpCsExt1D_Vm2 : *via* VM1, VM2, VR3, PO3 et VE3
  - CpCsExt2G : *via* VM3, VR3, PO3, VM23, VM26 et VE1
  - CpCsExt2D : *via* VM1, VR1, PO1, VM26, VM23 et VE1
  - CpCsExt2G_Vm2 : *via* VM3, VM2, VR1, PO1, VM26, VM23 et VE1
  - CpCsExt2D_Vm2 : *via* VM1, VM2, VR3, PO3, VM23, VM26 et VE1
  - CpCsExt3G : *via* VM3, VR3, PO3, VM22, VM29, VM25 et VE1
  - CpCsExt3D : *via* VM1, VR1, PO1, VM25, VM29, VM22 et VE1
  - CpCsExt3G_Vm2 : *via* VM3, VM2, VR1, PO1, VM25, VM29, VM22 et VE1
  - CpCsExt3D_Vm2 : *via* VM1, VM2, VR3, PO3, VM22, VM29, VM25 et VE1
  - CpCsInt1D : *via* VM24, VM25 et VE1
  - CpCsInt1G : *via* VM21, VM22 et VE3
  - CpCsInt2G : *via* VM21, VM29, VM25 et VE1
  - CpCsInt2D : *via* VM24, VM29, VM22 et VE3
  - CpCsInt3G : *via* VM21, VM22, VM23, VM26 et VE1
  - CpCsInt3D : *via* VM24, VM25, VM26, VM23 et VE3
  - CpCsInt4D : *via* VM21, VM29, VM25, VM26, VM23 et VE3
  - CpCsInt4G : *via* VM24, VM29, VM22, VM23, VM26 et VE1

4.2.2.2 Modélisation du procédé et du séquenceur

Les modèles génériques présentés en Section 2.2.2 sont instanciés afin de créer 22 modèles M_1 d'équipements (un exemple de modèle d'équipement est donné en Figure 4.10), 24 modèles M_2 de fonctions (un exemple de fonction est donné en Figure 4.11), 1 modèle de recette M_3 et un séquenceur initial S_2.

De manière complémentaire, 24 modèles abstraits de fonctions $\mathcal{A}(M_2^k)$ sont instanciés à partir du template présenté en Figure 4.8. Un exemple d'instance de modèle abstrait de fonction est donné en Figure 4.12.

4.2.2.3 Génération d'une trace admissible aux niveaux recette M_3 / fonctions M_2

Une fois les modèles de séquenceur S_3, de recette M_3 et d'abstraction de fonctions $\mathcal{A}(M_2^k)$ instanciés, une vérification de la propriété UppAal $E <> Sequenceur.FinSequence$ est lancée. La propriété étant vérifiée, une trace est retournée par UppAal sous la forme d'un fichier texte.

10. Ne sont mentionnés que les équipements traversés par le fluide

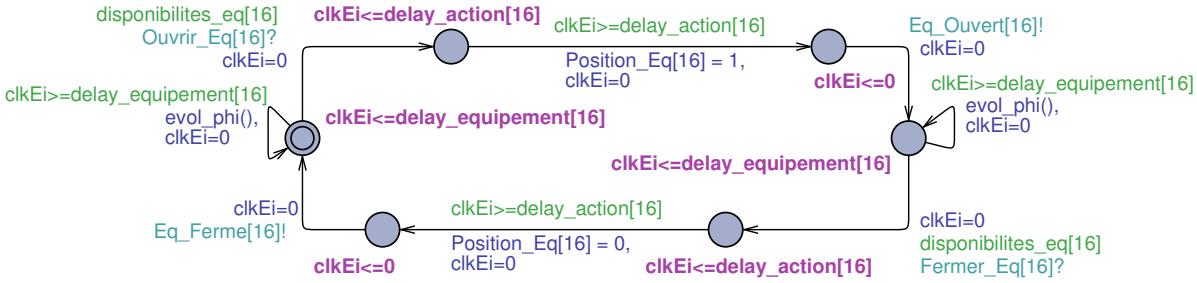


FIGURE 4.10 – Modèle M_1 de la vanne $VM21$

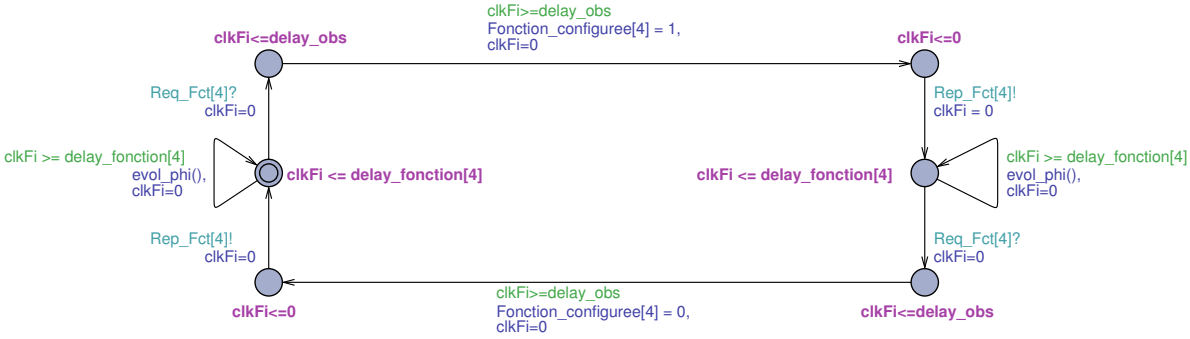


FIGURE 4.11 – Modèle M_2 de la fonction $CpCsExt1G$

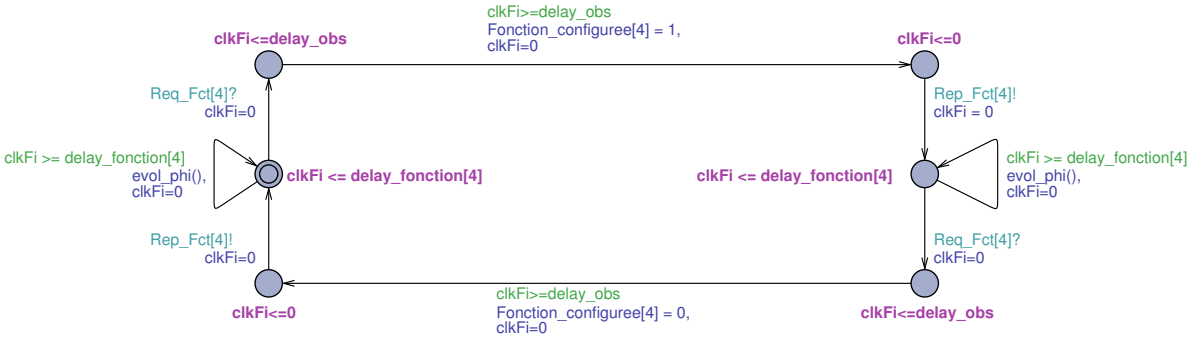


FIGURE 4.12 – Modèle abstrait $\mathcal{A}(M_2^k)$ de la fonction $CpCsExt1G$

4.2.2.4 Interprétation de la trace d'exécution

L'outil d'analyse de séquences interprète la trace générée, pour en extraire la suite des canaux de synchronisation utilisés afin d'atteindre la situation objectif :

1. Req_Sit_B ! → Req_Sit_B ?
2. Req_Fct[4] ! → Req_Fct[F_id] ?
3. Rep_Fct[F_id] ! → Rep_Fct[4] ?
4. Delay : 10
5. Rep_Sit_B ! → Rep_Sit_B ?

4.2.2.5 Génération d'un nouveau séquenceur S_2

Á l'aide de la suite récupérée ci-dessus, l'outil d'analyse de séquences génère, conformément à l'algorithme 3, un automate pouvant être utilisé comme séquenceur S_2 au niveau fonctions/ équipements. Cet automate est présenté en FIGURE 4.13.

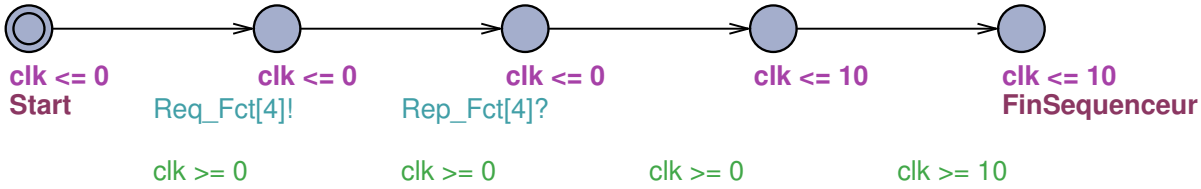


FIGURE 4.13 – Exemple d'un séquenceur S_2 généré automatiquement

4.2.2.6 Génération d'une trace admissible aux niveaux fonctions M_2 / équipements M_1

En considérant maintenant les modèles de fonctions M_2, les modèles d'équipements M_1, et le modèle de séquenceur S_2 généré précédemment, une nouvelle vérification de la propriété UppAal $E <> Sequenceur.FinSequence$ est lancée. Cette itération permet d'obtenir une trace qui est interprétée de la manière suivante :

1. Req_Fct[4] ! → Req_Fct[F_id] ?
2. Req_Eq[8] ! → Req_Eq[E_id] ?
3. Eq_Ouvert[E_id] ! → Eq_Ouvert[8] ?
4. Req_Eq[7] ! → Req_Eq[E_id] ?
5. Eq_Ouvert[E_id] ! → Eq_Ouvert[7] ?
6. Req_Eq[9] ! → Req_Eq[E_id] ?
7. Eq_Ouvert[E_id] ! → Eq_Ouvert[9] ?
8. Req_Eq[10] ! → Req_Eq[E_id] ?
9. Eq_Ouvert[E_id] ! → Eq_Ouvert[10] ?
10. Rep_Fct[F_id] ! → Rep_Fct[4] ?
11. Delay : 10
12. Rep_Sit_B ! → Rep_Sit_B ?

4.2.2.7 Construction de la séquence globale

Cette seconde itération de recherche d'atteignabilité ayant généré une trace incluant le niveau M_1, il s'agit de la dernière itération possible (FIGURE 4.1). L'outil d'analyse de séquences permet alors la construction de la séquence globale des actions à effectuer.

Après conversion des identifiants en noms d'équipements, l'outil retourne la séquence suivante, exprimée dans un langage compréhensible par un opérateur humain, et permettant de conduire la plate forme CISPI de manière à pouvoir remplir la bâche $001BA$:

1. Ligner la fonction « CpCsExt1G »
   (a) Ouvrir l'équipement VR3
   (b) Ouvrir l'équipement VM3
   (c) Ouvrir l'équipement PO3
   (d) Ouvrir l'équipement VE3
2. Attendre 10 unités de temps
3. *La situation souhaitée est atteinte.*

4.3 Cas d'étude #2 : Plate-forme expérimentale issue du projet CONNEXION

L'objectif sur ce second cas d'étude est double. Il s'agit d'une part d'évaluer l'approche proposée sur un cas d'étude de taille plus importante, afin de déterminer son intérêt pour le traitement de cas de dimension industrielle. Il s'agit d'autre part d'appliquer la génération de séquences d'actions de conduite dans le cadre de l'ingénierie du contrôle/commande, afin de vérifier, suite à des modifications apportées à l'installation, qu'il existe toujours une séquence d'actions permettant de conduire cette dernière vers les situations souhaitées.

4.3.1 Présentation du cas d'étude

Dans le cadre du projet CONNEXION, la validation des travaux relatifs à l'ingénierie du contrôle/commande et à la conduite s'est effectuée sur une cas d'étude concret, proposé par des industriels de la filière nucléaire. Les types d'équipements, les redondances, la complexité et les modes de marche sont représentatifs de ce qui se rencontre dans les centrales nucléaires de production d'électricité (CNPE). Ce cas d'étude de taille réduite est basé sur les principes d'un sous-ensemble central dans la constitution d'une centrale nucléaire, composé de six "Systèmes Élémentaires" (parmi plus de 250 que compte une tranche de CNPE), interconnectés comme le montre la Figure 4.14 [Piriou, 2015], et représentatif de divers aspects : besoins en régulation, règles de sécurité, diversité et redondance matérielles ....

Pour des raisons de confidentialité, seuls les principes de l'architecture du cas d'étude sont présentés ici, et non les modèles instanciés. Les données quantitatives sur les résultats sont cependant discutées dans la Section 4.4.

Ces six systèmes élémentaires, dont les acronymes sont utilisés dans la Figure 4.14, correspondent à :

RCP : Réacteur Circuit Primaire

RRA : Refroidissement du Réacteur à l'Arrêt

PTR : Traitement et Refroidissement de la Piscine de refroidissement

RCV : Réacteur - Contrôle Chimique et Volumétrique

TEP : Traitement des Effluents Primaires

REA : Réacteur - Eau d'Appoint

En particulier, on s'intéresse ici au cas du basculement entre les systèmes élémentaires RRA et PTR, qui permettent tous deux de refroidir le cœur du réacteur.

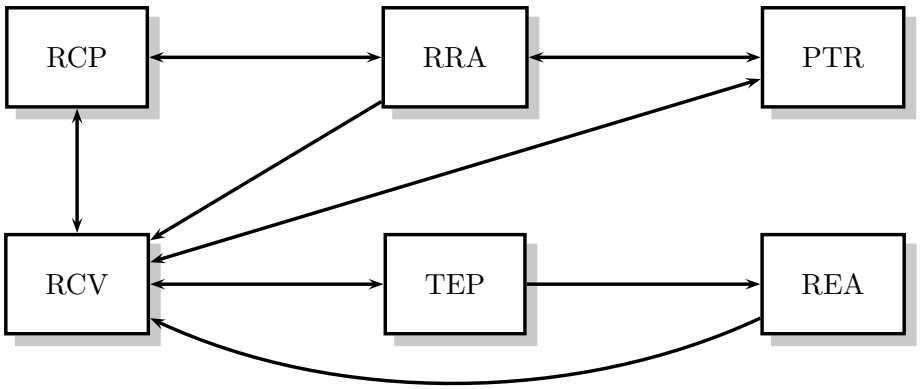


FIGURE 4.14 – Principes de structuration du cas d'étude CONNEXION

Le système élémentaire RRA contrôle la température primaire dans les plages où la température est inférieure à 180˚C et la pression inférieure à 28 bars, en permettant l'évacuation de la puissance résiduelle du réacteur à l'arrêt par deux files de refroidissement indépendantes. Il participe également via une liaison avec le système élémentaire RCV [11] au contrôle chimique et volumétrique. Une représentation graphique des principes de l'architecture de base du système élémentaire est donné par la FIGURE 4.15.

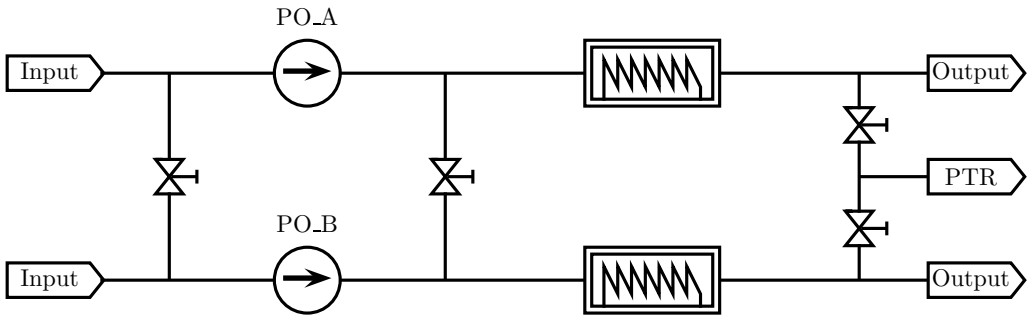


FIGURE 4.15 – Schéma simplifié de circulation des fluides du système élémentaire RRA

Pour ce système élémentaire RRA, trois configurations seront envisagées, chacune permettant de refroidir le fluide circulant :

- Utilisation d'une voie A (voie « du haut » sur la FIGURE 4.15)
- Utilisation d'une voie B (voie « du bas » sur la FIGURE 4.15)
- Utilisation simultanée de ces deux voies

Essentiel pour la sécurité, le système élémentaire PTR a pour mission principale le maintien en conditions opérationnelles des masses d'eau considérables assurant la barrière neutronique et le refroidissement du combustible hors cuve et cuve ouverte. Il assure par exemple des fonctions de refroidissement, filtration, purification et vidange des piscines, ainsi qu'un secours pour le circuit RRA. Une représentation graphique des principes de l'architecture de base du système élémentaire est donné par la FIGURE 4.16.

11. Le circuit RCV contrôle la volumétrie, la pression et la chimie de l'eau primaire

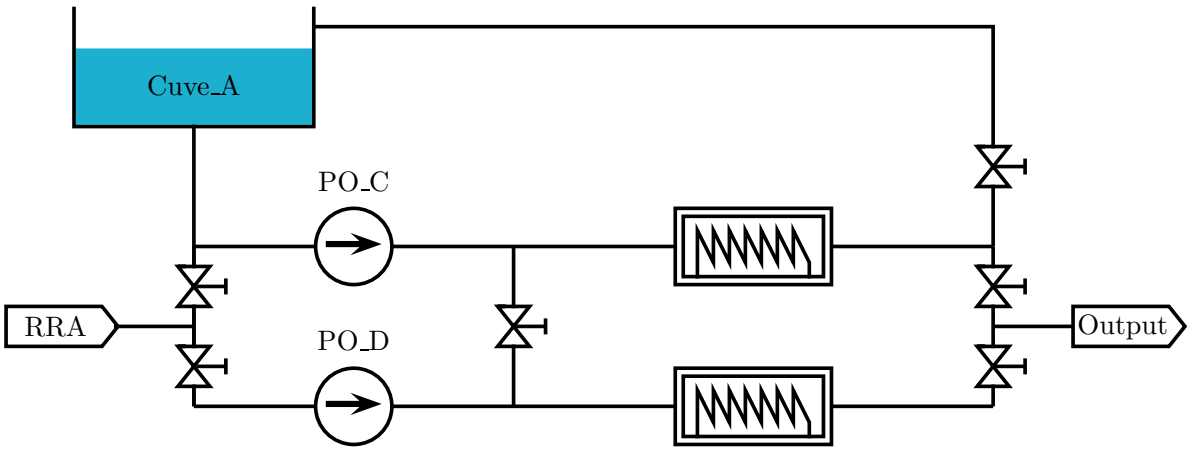


FIGURE 4.16 – Schéma simplifié de circulation des fluides du système élémentaire PTR

Pour ce système élémentaire, huit configurations seront envisagées :
- Quatre configurations permettant de refroidir un fluide circulant :
  - Utilisation de la pompe C et de l'échangeur C
  - Utilisation de la pompe D et de l'échangeur C
  - Utilisation de la pompe C et de l'échangeur D
  - Utilisation de la pompe D et de l'échangeur D
- Quatre configurations permettant de refroidir une autre piscine :
  - Utilisation de la pompe C et de l'échangeur C
  - Utilisation de la pompe D et de l'échangeur C
  - Utilisation de la pompe C et de l'échangeur D
  - Utilisation de la pompe D et de l'échangeur D

4.3.2 Génération de séquences pour la validation du contrôle-commande

Si l'approche de génération de séquences proposée dans ce manuscrit peut être utilisée pour générer des séquences d'actions de conduite, elle peut également être utilisée dans un contexte plus global d'ingénierie (SECTION 1.1.2.2). En effet, il est probable d'arriver dans les premières phases de conception, suite à des modifications du procédé (par exemple ajout ou suppression d'équipements) et de son contrôle/commande, ou en cas d'insponibilité d'équipements, dans des situations bloquantes pour le contrôle du procédé. Autrement dit, il est possible qu'il n'existe pas de séquence d'actions de conduite permettant d'atteindre une situation objectif donnée. Les problèmes de ce type peuvent être liés, par exemple, à un manque de redondance ou une exploration (manuelle) insuffisante des possibilités offertes par les redondances du procédé. Dans ce cas, la méthode de génération de séquences permet d'assurer, ou d'infirmer, qu'il existe une manière de conduire le procédé jusqu'à une situation objectif. S'il s'avère que cela est impossible, cela peut mener à une refonte de l'architecture fonctionnelle et/ou organique du système tel qu'il avait été pensé.

Sur le cas d'étude #2, les systèmes élémentaires RRA et PTR, qui se composent de 65 équipements pour lesquels 11 fonctions sont configurables, ont été modélisés sous UppAal, en 4 niveaux hiérarchiques (*équipements*, *fonctions*, *systèmes* et niveau « *inter-systèmes* ». Différents

scénarios ont été testés : suppressions de vannes, indisponibilités d'équipements (pompe PO_A par exemple) suite à une défaillance... Dans chacun des cas, des recherches d'atteignabilité ont été conduites, et, lorsque des séquences ont pu être générées, cela signifiait que les modifications (ou les pannes), permettaient d'atteindre la situation objectif, en suivant la séquence d'action générée. Dans le cas contraire, lorsque la situation cible n'était pas atteignable, cela signifiait, respectivement, que la modification n'est pas envisageable sous cette forme, ou que le système ne peut être conduit avec ces indisponibilités. Une solution possible à ce type de problème est une modification de l'architecture de l'installation, pour pallier au manque de redondance des équipements et circuits.

Ce principe d'évaluation d'atteignabilité de situation suite à une modification du procédé a été exploré dans le cadre du projet CONNEXION, en coopération avec les sociétés Corys et EDF d'une part, qui ont fourni un modèle simulable du procédé et de son contrôle commande, et le CEA d'autre part, qui a fourni un outil de surveillance de l'exécution d'un scénario de simulation (ARTIMON [Rapin, 2014]). Les séquences d'actions de conduite générées ont été exécutées sur le simulateur, et au cours de cette simulation, des paramètres de sécurité, liés notamment à des valeurs de variables physiques sur des intervalles temporels, ont été surveillés. L'intérêt de coupler la génération de séquences, basée sur des modèles à états discrets, à une simulation basée sur des lois de la physique, permet une validation temporelle des séquences, et de la bonne prise en compte des règles métiers liées à la sécurité dans les modèles utilisés pour la génération de séquences [Cochard et al., 2015c].

4.4 Evaluation de l'approche de génération de séquences

Cette section a pour objectif d'évaluer les apports de l'approche de génération de séquences d'actions de conduite proposée, en termes de modélisation et de passage à l'échelle. Pour cela elle présente le résultat d'un ensemble d'expérimentations qui ont été menées, sur la base des deux cas d'étude présentés dans ce chapitre.

Initialement, notre approche de modélisation était relativement « instinctive », c'est-à-dire qu'elle n'était guidée par aucun cadre formel de modélisation, et les générations de séquences s'effectuaient sur le modèle complet. Les modèles de la plateforme CISPI ont ainsi été faits au départ sans méthode, puis à l'aide du cadre de modélisation. Ceux concernant le second cas d'étude, faits postérieurement à ceux de CISPI, ont été faits suivant le cadre proposé. Sur la base de ces modèles, des générations de séquences ont été effectuées, seules, puis par lots, selon une approche « classique », puis en suivant l'approche itérative proposée dans le Chapitre 3.

4.4.1 Apports de l'approche proposée en termes de modélisation

Les premiers travaux de modélisation de la plateforme CISPI et du cas d'étude #2 que nous avons faits [Cochard et al., 2015a], reposaient sur une approche « instinctive » de modélisation. Pour élaborer ces premiers modèles, plusieurs itérations ont été faites, nécessitant plusieurs semaines avant qu'une version utilisable pour la génération de séquences par recherche d'atteignabilité soit produite.

La proposition, puis l'utilisation du cadre de modélisation proposé au Chapitre 2 et raffiné au Chapitre 3 a permis un gain de temps certain, grâce à l'utilisation de modèles génériques

instanciés. Si la création des modèles génériques demande du temps, leur utilisation est ensuite assez rapide. Les modèles de la plateforme CISPI utilisés pour les lots de génération de séquences du cas d'étude #1 analysés ci-dessous, ont nécessité moins de deux jours de travail. Il en a été de même pour la modélisation du cas d'étude #2, qui a pris moins de 5 jours de travail, alors que sa dimension est bien supérieure à celle du cas d'étude #1.

On peut donc en conclure que l'approche proposée permet, grâce à la réutilisation et l'instanciation de modèles génériques, un gain de temps appréciable.

4.4.2 Apports de l'approche proposée pour le passage à l'échelle

4.4.2.1 Apports concernant la taille de l'espace d'état

L'utilisation d'UppAal en lignes de commandes permet de connaître la taille de l'espace d'état parcouru. Nous avons utilisé cette option afin de mesurer l'apport de l'approche de génération de séquences d'actions de conduite en termes de réduction de la taille de l'espace d'état.

En ce qui concerne le cas d'étude #1, pour la génération d'une séquence d'actions couvrant l'ensemble des 3 niveaux (recette, fonctions, équipements), nous n'avons pas noté d'apport significatif au niveau de la réduction de l'espace d'état. En effet, la taille de l'espace d'état parcouru lors de la seconde itération de recherche d'atteignabilité concernant les fonctions et les équipements n'est que très légèrement inférieure à celle de l'espace d'état utilisé sur le modèle complet. Cette faible gain s'explique d'une part par le fait que les modèles de la plateforme CISPI, s'ils sont plus importants que ceux du cas d'étude "jouet" utilisé aux chapitres précédents, restent de taille modeste, et d'autre part par le fait que seuls trois niveaux hiérarchiques ont été utilisés. En effet, le niveau hiérarchique le plus haut n'étant composé que d'un seul modèle de recette, les deux niveaux *fonctions* et *équipements* utilisés pour la seconde itération comportent donc presque le même nombre de modèles que l'ensemble utilisé pour l'approche « classique ».

On note cependant, sur le cas d'étude #2, que la différence de taille de l'espace d'état est plus sensible entre les deux approches « classique » et itérative. Suivant les cas, elle oscille entre 15 % et 20%. A titre d'exemple, dans le cas d'une recherche de la séquence la plus courte, avec un parcours aléatoire de l'espace d'état, on obtient un gain de l'ordre de 15 % :

- Dans le cas de l'approche « classique » : 3 552 124 états sont parcourus ;
- Dans le cas de la seconde itération : 3 039 278 états sont parcourus.

Cette différence dans le cas du second cas d'étude s'explique en partie par le fait que le niveau hiérarchique n'a que deux systèmes à coordonner. Le gain aurait été plus important si le nombre de systèmes à coordonner avait été lui-même plus important. Afin de mettre en évidence l'intérêt de l'approche itérative, nous avons donc été amené à l'expérimenter sur un cas d'étude "virtuel", impliquant 4 niveaux hiérarchiques et un grand nombre de fonctions et d'équipements, mais construit de manière artificielle par une génération aléatoire des relations entre :

- 11 systèmes élémentaires ;
- de l'ordre de 120 fonctions ;
- de l'ordre de 800 équipements.

L'objectif n'était pas d'être représentatif d'une installation réelle mais de s'approcher des facteurs d'échelle auxquels pourrait être confrontée notre approche dans un contexte industriel. Sur ce cas, l'utilisation de l'approche« classique » basée sur une exploration unique de l'ensemble des modèles n'a pas permis la génération d'une séquence, alors que cela a été possible avec l'approche itérative.

Par ailleurs, un des intérêts de l'approche itérative est que, dans le cas où l'on souhaite générer une séquence qui ne porte que sur une partie des niveaux hiérarchiques, (par exemple un enchaînement de fonctions, sans se préoccuper dans un premier temps des équipements qui seront à manipuler), il est possible d'avoir un gain très important au niveau de la taille de l'espace d'état. En effet, pour générer une séquence de fonctions, il est nécessaire avec l'approche « classique » de parcourir l'ensemble des niveaux, alors qu'une ou deux itérations peuvent être suffisantes avec l'approche itérative, en fonction du nombre de niveau hiérarchiques considérés. Ceci a été vérifié avec le cas d'étude #2, pour lequel il a été possible de générer une séquence d'actions de conduite du niveau *fonctions* en parcourant des espaces d'état dont les tailles sont :

- dans le cas de l'approche « classique » : 3 552 124 états ;
- dans le cas de l'approche itérative : 80 états.

Ce gain très important s'explique par le fait que seuls les niveaux nécessaires sont parcourus dans le cas de l'approche itérative, et que les niveaux les plus bas sont ceux qui contiennent le plus de modèles à parcourir.

4.4.2.2 Apports concernant le temps de génération de séquence

Pour analyser les gains potentiels en termes de temps nécessaires à la génération de séquences, des lots de 2 000 générations ont été effectués sur chacun des deux cas d'étude, en suivant d'une part une approche « classique » telle que celle présentée au Chapitre 2, et d'autre part l'approche itérative présentée au Chapitre 3. Ces lots sont obtenus en modifiant la manière dont UppAal explore l'espace d'état (utilisation de *seeds* différents), afin d'obtenir des séquences potentiellement différentes. Afin que la comparaison des valeurs temporelles soit possible entre les différents lots de génération de séquences, ils ont été effectués sur une même machine.

En ce qui concerne le cas d'étude #2, si l'on considère l'ensemble du modèle selon une approche « classique », le temps moyen d'exploration est de 1,976 s. Les temps mis lors des 2 000 explorations sont synthétisés dans l'histogramme ci-dessous (FIGURE 4.17), représentant le nombre d'occurrences en fonction du temps nécessaire à l'exploration.

Dans le cas où l'on utilise l'approche de génération itérative présentée au Chapitre 3, il y a plusieurs itérations à considérer. Le temps moyen pour l'itération aux niveaux *Recette* M_3 / *Fonctions* M_2 est de 0,552 s, et le temps moyen pour les niveaux *Fonctions* M_2 / *Equipements* M_1 est de 1,721 s. Les deux ensembles de valeurs des temps d'exploration, relatifs à chacune de ces itérations, sont synthétisés dans les histogrammes ci-dessous (FIGURE 4.18), représentant le nombre d'occurrences en fonction du temps nécessaire à l'exploration.

On constate ainsi que le temps moyen de l'itération la plus longue, concernant les niveaux *Fonctions* M_2 / *Equipements* M_1, est légèrement inférieur de celui de l'approche « classique ». Sur les différents cas d'étude traités, les temps nécessaires à la génération de séquences restent

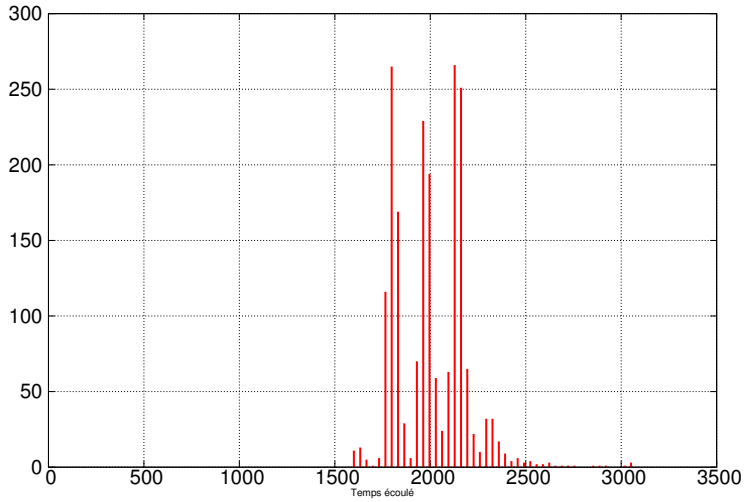
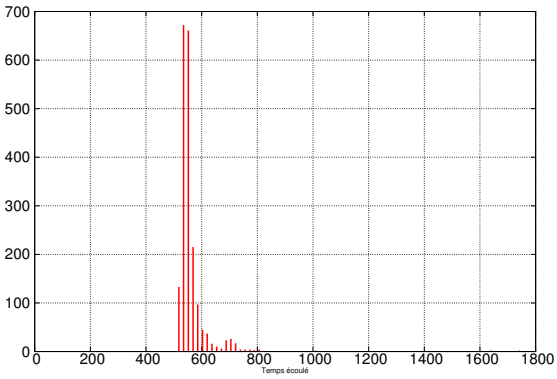
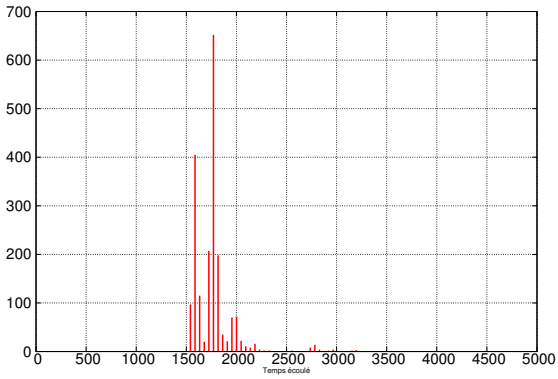


FIGURE 4.17 – Temps d'exploration du modèle complet du cas d'étude CONNEXION



(a) Temps d'exploration des niveaux *Recette* M_3 / *Fonctions* M_2



(b) Temps d'exploration des niveaux *Fonctions* M_2 / *Equipements* M_1

FIGURE 4.18 – Temps d'exploration de deux niveaux du cas d'étude CONNEXION

tout à fait acceptables dans le cadre de l'ingénierie. Il en a été de même avec les temps utilisés pour des recherches de séquences optimales, pour lesquelles les temps de parcours de l'espace d'état les plus longs ont été de l'ordre d'une dizaine de minutes.

Si la comparaison des temps nécessaires donne une première indication, l'analyse de la valeur absolue de ces temps ne peut cependant pas être concluante, puisque qu'elle repose sur les capacités de la machine supportant les simulations. On peut donc imaginer qu'un ordinateur plus puissant obtiendrait des résultats plus rapidement.

Conclusion

Ce chapitre présente un ensemble d'outils permettant l'application de l'approche de génération itérative de séquences présentée au Chapitre 3. Ces outils ont permis de générer des séquences d'actions de conduite sur deux cas d'étude, l'un de laboratoire fortement redondant,

et l'autre industriel de taille supérieure. Les propositions méthodologiques du Chapitre 2, utilisant une structuration hiérarchiquement et des patrons, ont montré leur intérêt en termes de modélisation sur les cas d'étude présentés.

Pour déterminer les apports pour la passage à l'échelle, ce chapitre compare l'approche itérative avec une approche plus « classique » utilisant l'ensemble du modèle de l'installation à conduire. Concernant la taille de l'espace d'état parcouru, ainsi que les temps nécessaires à la génération de séquence, les modèles structurés selon 3 niveaux ne permettent pas de bien mettre en évidence le gain apporté par l'approche itérative. Ce gain est cependant vérifié sur des modèles de grande taille, structurés sur un nombre plus important de niveaux hiérarchiques. Cela peut s'expliquer par le fait que :

- Le découpage par paires de niveaux implique la génération de séquenceurs restrictifs aux différents niveaux. Ceci implique une limitation de l'espace d'état parcouru ;
- Le cône d'influence joue également un rôle important dans la restriction de l'exploration de l'espace d'état, impliquant une réduction du nombre d'éléments d'intérêt à explorer.

La minimisation du temps nécessaire à l'obtention d'une séquence n'était pas la priorité donnée dans ces travaux, car il est fortement conditionné par la puissance de calcul de la machine utilisée. Cependant, les résultats conduisent à observer que le temps maximal relevé sur les deux cas d'études est de l'ordre de 30 s, ce qui reste un temps très faible pour traiter, en ingénierie, un système volontairement très redondant.

Les lots de génération de séquences ont permis de montrer que de nombreuses séquences différentes peuvent être générées. En effet, en 2 000 générations, un peu plus de 1 000 séquences différentes ont été générées.

Conclusion

Les travaux présentés dans ce manuscrit se positionnent dans le contexte de la préparation et de l'ingénierie de la conduite de systèmes complexes critiques. Ils montrent l'intérêt d'utiliser une approche de génération de séquences d'actions de conduite par recherche d'atteignabilité dans des modèles formalisés en automates à états temporisés.

L'approche proposée repose, d'une part, sur une modélisation structurée de l'installation à conduire, formalisée à l'aide d'automates à états temporisés. Cette modélisation est guidée par l'utilisation de patrons génériques, qui permettent de systématiser le travail du modélisateur. De plus, la modularité induite par l'approche proposée, couplée à la généricité des patrons, a comme avantage de faciliter la réutilisation des modèles. D'un point de vue industriel, ces deux axes de proposition permettent de réduire le temps d'élaboration des modèles à des échelles compatibles avec les processus d'ingénierie actuels.

La proposition repose, d'autre part, sur la mise en œuvre itérative de mécanismes de recherche d'atteignabilité. L'explosion combinatoire étant l'une des principales limites à l'utilisation industrielle des approches de model-checking, les itérations et raffinements successifs sur lesquels repose l'approche proposée permettent de réduire la taille de l'espace d'état parcouru. Ces itérations ne considèrent, en effet, lors des recherches d'atteignabilité, qu'un sous-ensemble des modèles de l'installation à conduire. L'approche met également en œuvre le principe du cône d'influence afin de limiter l'exploration de l'espace d'état à un sous-ensemble en lien avec la situation souhaitée.

Différents cas d'étude ont été utilisés dans nos travaux afin d'éprouver la faisabilité de l'approche. Dans un premier temps, la plateforme CISPI du CRAN a permis d'illustrer, sur un exemple de laboratoire, l'aide à la préparation de la conduite que pouvait offrir la génération d'une séquence admissible, d'un ensemble de séquences ou bien d'une séquence optimisée en terme de longueur ou de durée.

Le second cas d'étude issu du projet CONNEXION, a permis, d'une part, d'évaluer les capacités de passage à l'échelle de notre approche, et, d'autre part, de montrer l'intérêt de la génération de séquence d'actions de conduite pour la vérification d'architectures en ingénierie du contrôle/commande.

Ces travaux offrent de nombreuses perspectives de recherche selon trois axes : niveau de maturité, modélisation du procédé et prise en compte d'un contexte incertain dans la préparation aux actions de conduite.

Le prototype de génération, développé dans le cadre de ce travail de thèse a permis d'automatiser la création et l'instanciation de modèles automates, ainsi que le lancement des itérations de recherche d'atteignabilité dans UppAal par lots. Le transfert de cette approche dans un contexte industriel repose sur deux éléments :

- stabilisation et industrialisation du prototype, que l'on peut estimer en l'état actuel à un niveau de TRL 3 (Technology Readiness Level) ; cela passe notamment par la création d'une interface graphique en remplacement des lignes de commande actuellement utilisées, par la fiabilisation de l'ensemble et la vérification du code des applications,
- la mise en œuvre de la démarche dans un environnement industriel réel sur la base de situations réelles (installations et scénarios de conduite).

Le deuxième axe de recherche concerne la niveau de représentation de l'installation à conduire, et notamment la dynamique du procédé (évolution des niveaux, pressions, températures, ...). Les modèles utilisés actuellement se limitent à une représentation de cette dernière sous une forme abstraite et discrétisée. Deux axes sont envisageables :

- remplacer le formalisme de modélisation utilisé actuellement (automates temporisés) par un formalisme apte à supporter la modélisation du procédé sous la forme d'équations algébro-différentielles, comme par exemple les automates hybrides [Henzinger, 2000] ; cette approche, séduisante sur le papier, peut néanmoins engendrer quelques difficultés lors de l'exploration de l'espace d'état,
- coupler l'algorithme actuel d'exploration de l'espace d'état avec un modèle exécutable du procédé, tels que les simulateurs de procédé utilisés pour la formation des opérateurs [Blanchon and Bruneau, 2016] ; cela revient à abandonner la définition et le parcours de l'espace d'état au profit d'une exécution synchronisée (simulation ou co-simulation) des différents modèles ; la difficulté principale réside dans l'algorithme de recherche de la solution conduisant à l'état recherché, qui devra vraisemblablement intégrer l'exécution de multiples scénarios jusqu'à obtention d'un scénario admissible.

Enfin, le troisième axe de recherche concerne la prise en compte dans nos modèles des aléas de fonctionnement (pannes des équipements, événements perturbateurs externes, ...) et des incertitudes sur les durées opératoires. Une piste envisageable est de considérer l'utilisation d'automates stochastiques pour la génération de séquences de telle sorte à pouvoir évaluer leur robustesse vis à vis des aléas (par exemple sous la forme d'une probabilité d'atteindre l'état recherché avec une séquence donnée). Cette prise en compte soulèvera vraisemblablement quelques questions en termes d'exploration de l'espace d'état mais permettrait d'enrichir de manière significative l'aide à la préparation des actions de conduite en proposant aux opérateurs une évaluation de la robustesse des différents scénarios de conduite admissibles.

Bibliographie

[Abichou, 2013] Abichou, B. (2013). *Contribution à la formalisation de bilans/états de santé multi-niveaux d'un système pour aider à la prise de décision en maintenance : agrégation d'indicateurs par l'intégrale de Choquet*. PhD thesis, Université de Lorraine.

[Alengry, 1989a] Alengry, P. (1989a). Analyse du travail de pilotage d'une centrale nucleaire (i) : le systeme socio-technique.

[Alengry, 1989b] Alengry, P. (1989b). Analyse du travail de pilotage d'une centrale nucleaire (ii) : les classes de situation.

[Alexander et al., 1977] Alexander, C., Ishikawa, S., Silverstein, M., i Ramió, J. R., Jacobson, M., and Fiksdahl-King, I. (1977). *A pattern language*. Gustavo Gili.

[Alur and Dill, 1994] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical computer science*, 126(2) :183–235.

[Arzen and Johnsson, 1996] Arzen, K.-E. and Johnsson, C. (1996). Object-oriented sfc and isa-s88. 01 recipes presented at the world batch forum. *ISA transactions*, 35(3) :237–244.

[Behrmann et al., 2006a] Behrmann, G., David, A., and Larsen, K. G. (2006a). A tutorial on uppaal 4.0. *Department of computer science, Aalborg university*.

[Behrmann et al., 2006b] Behrmann, G., David, A., Larsen, K. G., Hakansson, J., Petterson, P., Yi, W., and Hendriks, M. (2006b). Uppaal 4.0. In *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*, pages 125–126. IEEE.

[Behrmann et al., 2001] Behrmann, G., Fehnker, A., Hune, T., Larsen, K., Pettersson, P., and Romijn, J. (2001). *Efficient guiding towards cost-optimality in uppaal*. Springer.

[Bigot et al., 2003] Bigot, C., Faivre, A., Gallois, J.-P., Lapitre, A., Lugato, D., Pierron, J.-Y., and Rapin, N. (2003). Automatic test generation with agatha. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 591–596. Springer.

[Blanchon and Bruneau, 2016] Blanchon, J.-C. and Bruneau, O. (2016). Simulez, partagez, opérez ! In *L'avenir se construit aujourd'hui*. Edition Multimedia.

[Bouajjani et al., 2004] Bouajjani, A., Habermehl, P., and Vojnar, T. (2004). Abstract regular model checking. In *Computer Aided Verification*, pages 372–386. Springer.

[Bouaziz et al., 2016] Bouaziz, M.-F., Marangé, P., Voisin, A., and Jean-François, P. (2016). Prise en compte de l'état réel d'un système complexe pour l'aide à la décision en phase d'exploitation. *Journal Européen des Systèmes Automatisés (JESA)*.

[Bouaziz et al., 2015] Bouaziz, M.-F., Marangé, P., Voisin, A., and Pétin, J.-F. (2015). Health checkup indicators-based safety criteria for operating sequences ranking of critical systems.

In *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'2015*, Paris, France.

[Bouffaron, 2016] Bouffaron, F. (2016). *Executable system co-specification based on models : Application to interactive conduct of critical industrial process*. Theses, Université de Lorraine.

[Bouffaron et al., 2014] Bouffaron, F., Dupont, J.-M., Mayer, F., and Morel, G. (2014). Integrative construct for model-based human-system integration : a case study. In *19th IFAC World Congress, IFAC'14*.

[Cassandras and Lafortune, 2008] Cassandras, C. G. and Lafortune, S. (2008). Introduction to discrete event systems.

[Castelnuovo et al., 2007] Castelnuovo, A., Ferrarini, L., and Piroddi, L. (2007). An incremental petri net-based approach to the modeling of production sequences in manufacturing systems. *Automation Science and Engineering, IEEE Transactions on*, 4(3) :424–434.

[CENELEC, 1995] CENELEC (1995). Env 12204, advanced manufacturing technology, systems architecture, constructs for enterprise modelling.

[Cimatti et al., 2002] Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., and Tacchella, A. (2002). Nusmv 2 : An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 359–364. Springer.

[Clanché et al., 2010] Clanché, F., Gouyon, D., Dobre, D., Pétin, J.-F., Morel, G., et al. (2010). Plate-forme pour la conduite interactive et sûre. *Plate-forme pour la conduite interactive et sûre*.

[Clarke et al., 2001] Clarke, E., Biere, A., Raimi, R., and Zhu, Y. (2001). Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1) :7–34.

[Clarke et al., 2000] Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2000). Counterexample-guided abstraction refinement. In *Computer aided verification*, pages 154–169. Springer.

[Clarke and Emerson, 1981] Clarke, E. M. and Emerson, E. A. (1981). Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71. Springer-Verlag.

[Clarke et al., 1994] Clarke, E. M., Grumberg, O., and Long, D. E. (1994). Model checking and abstraction. *ACM transactions on Programming Languages and Systems (TOPLAS)*, 16(5) :1512–1542.

[Clarke et al., 1999] Clarke, E. M., Grumberg, O., and Peled, D. (1999). Model checking.

[Clarke and Wing, 1996] Clarke, E. M. and Wing, J. M. (1996). Formal methods : State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4) :626–643.

[Cochard et al., 2015a] Cochard, T., Gouyon, D., and Pétin, J.-F. (2015a). Génération de séquences d'actions sûres par recherche d'atteignabilité. *Génie logiciel*, (112) :43–50.

[Cochard et al., 2015b] Cochard, T., Gouyon, D., and Pétin, J.-F. (2015b). Generation of safe plant operation sequences using reachability analysis. *20th IEEE International Conference on Emerging Technologies and Factory Automation*.

[Cochard et al., 2016] Cochard, T., Gouyon, D., and Pétin, J.-F. (2016). Generation of safe operation sequences using riterative refinements and abstractions of timed automata. *21th IEEE International Conference on Emerging Technologies and Factory Automation*.

[Cochard et al., 2017] Cochard, T., Gouyon, D., and Pétin, J.-F. (2017). Safe operation sequences : A generation approach based on iterative refinements and abstractions of timed automata. In *20th IFAC World Congress, IFAC 2017*, Toulouse, France.

[Cochard et al., 2015c] Cochard, T., Gouyon, D., Rapin, N., and Pétin, J.-F. (2015c). Aiding generation and verification of action sequence. In *26th International Conference on Software & Systems Engineering and their Applications, ICSSEA'15*, Paris, France.

[David and Alla, 1992] David, R. and Alla, H. (1992). Petri nets and grafcet : tools for modelling discrete event systems.

[Devic, 2014] Devic, C. (2014). Connexion - contrôle commande nucléaire numérique pour l'export et la rénovation de la schématique progressive à la centrale numérique au service de la productivité des études. *Génie Logiciel*, pages 2–11.

[Devic, 2016] Devic, C. (2016). L'innovation se construit par brique. In *L'avenir se construit aujourd'hui*. Edition Multimedia.

[Devic and Dourgnon, 2015] Devic, C. and Dourgnon, A. (2015). L'innovation à portée de tous pour la performance de nos usines : défis et possibilités offertes par les technologies capacitantes. *Génie Logiciel*, pages 16–24.

[Devic and Morilhat, 2013] Devic, C. and Morilhat, P. (2013). Connexion contrôle commande nucléaire numérique pour l'export et la rénovation - coupler génie logiciel et ingénierie système : source d'innovations. *Génie Logiciel*, 104 :2–11.

[Dobre, 2010] Dobre, D. (2010). *Contribution à la modélisation d'un système interactif d'aide à la conduite d'un prodédé industriel*. PhD thesis, Nancy 1.

[Dobre et al., 2010] Dobre, D., Morel, G., and Gouyon, D. (2010). Improving human-system digital interaction for industrial system control : some systems engineering issues. *IFAC Proceedings Volumes*, 43(4) :290–295.

[D'silva et al., 2008] D'silva, V., Kroening, D., and Weissenbacher, G. (2008). A survey of automated techniques for formal software verification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(7) :1165–1178.

[Eshuis, 2009] Eshuis, R. (2009). Reconciling statechart semantics. *Science of Computer Programming*, 74(3) :65–99.

[Faraut et al., 2009] Faraut, G., Piétrac, L., and Niel, E. (2009). Formal approach to multimodal control design : Application to mode switching. *Industrial Informatics, IEEE Transactions on*, 5(4) :443–453.

[Foulkes et al., 1988] Foulkes, N., Walton, M., Andow, P., and Galluzzo, M. (1988). Computer-aided synthesis of complex pump and valve operations. *Computers & Chemical Engineering*, 12(9) :1035–1044.

[Fusillo and Powers, 1987] Fusillo, R. and Powers, G. (1987). A synthesis method for chemical plant operating procedures. *Computers & chemical engineering*, 11(4) :369–382.

[Fusillo and Powers, 1988] Fusillo, R. and Powers, G. (1988). Operating procedure synthesis using local models and distributed goals. *Computers & Chemical Engineering*, 12(9) :1023–1034.

[Galara, 2006] Galara, D. (2006). Roadmap to master the complexity of process operation to help operators improve safety, productivity and reduce environmental impact. *Annual Reviews in Control*, 30(2) :215–222.

[Galara and Hennebicq, 1999] Galara, D. and Hennebicq, J. P. (1999). Process control engineering trends. *Annual Reviews in Control*, 23 :1–11.

[Goubali, 2017] Goubali, O. (2017). *Apport des techniques de programmation par démonstration dans une démarche de génération automatique d'applicatifs de contrôle-commande*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechnique-Poitiers.

[Gouyon et al., 2016] Gouyon, D., Cochard, T., Pétin, J.-F., Voisin, A., Marangé, P., and Bouaziz, M.-F. (2016). Conduire à l'objectif. In *L'avenir se construit aujourd'hui*. Edition Multimedia.

[Grabisch et al., 2011] Grabisch, M., Marichal, J.-L., Mesiar, R., and Pap, E. (2011). Aggregation functions : means. *Information Sciences*, 181(1) :1–22.

[Harel, 1987a] Harel, D. (1987a). On the formal semantics of statecharts. In *IEEE Symposium on Logic in Computer Science, 1987*, pages 54–64.

[Harel, 1987b] Harel, D. (1987b). Statecharts : A visual formalism for complex systems. *Science of computer programming*, 8(3) :231–274.

[Harel, 2007] Harel, D. (2007). Statecharts in the making : a personal account. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pages 5–1. ACM.

[Harel and Kahana, 1992] Harel, D. and Kahana, C.-A. (1992). On statecharts with overlapping. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 1(4) :399–421.

[Harel et al., 1990] Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., and Trakhtenbrot, M. (1990). Statemate : A working environment for the development of complex reactive systems. *IEEE transactions on software engineering*, 16(4) :403–414.

[Henzinger, 2000] Henzinger, T. A. (2000). The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer.

[Holzmann, 1997] Holzmann, G. J. (1997). The model checker spin. *IEEE Transactions on software engineering*, 23(5) :279–295.

[IEC, 1997] IEC (1997). Iec 61512-1 : Batch control – part 1 : Models and terminology. *International Electrotechnical Commission*.

[IEC, 2002] IEC (2002). Iec 60848 : Grafcet specification language for sequential function charts.

[IEC, 2013] IEC (2013). Iec 62264-1 : Enterprise-control system integration – part 1 : Models and terminology. *International Electrotechnical Commission*.

[ISA, 1998] ISA (1998). Ansi/isa-88.01-1995 : Batch control - part 1 : Models and terminology. *The Instrumentation, Systems and Automation Society*.

[ISA, 2010] ISA (2010). Ansi/isa-95.00.01-2010 (iec 62264-1 mod) enterprise-control system integration - part 1 : Models and terminology. *The Instrumentation, Systems and Automation Society*.

[Jensen et al., 2007] Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4) :213–254.

[Kumar, 1992] Kumar, R. (1992). *Supervisory synthesis techniques for discrete event dynamical systems*. PhD thesis, University of Texas at Austin.

[Kurshan, 1994] Kurshan, R. P. (1994). Automata-theoretic verification of coordinating processes. In *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*, pages 16–28. Springer.

[Lemattre, 2013] Lemattre, T. (2013). *Allocation de fonctions de commande de systèmes critiques par recherche d'atteignabilité dans un réseau d'automates communicants*. PhD thesis, École normale supérieure de Cachan-ENS Cachan.

[Li et al., 1997] Li, H. S., Lu, M. L., and Naka, Y. (1997). A two-tier methodology for synthesis of operating procedures. *Computers & chemical engineering*, 21 :S899–S903.

[Li et al., 2014] Li, J.-H., Chang, C.-T., and Jiang, D. (2014). Systematic generation of cyclic operating procedures based on timed automata. *Chemical Engineering Research and Design*, 92(1) :139–155.

[Lieberman et al., 2006] Lieberman, H., Paternò, F., Klann, M., and Wulf, V. (2006). End-user development : An emerging paradigm. In *End user development*, pages 1–8. Springer.

[Lind, 1990] Lind, M. (1990). Representing goals and functions of complex systems—an introduction to multilevel flow modeling.

[Lind, 1994] Lind, M. (1994). Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence an International Journal*, 8(2) :259–283.

[Lind, 2011a] Lind, M. (2011a). An introduction to multilevel flow modeling. *Nuclear safety and simulation*, 2(1) :22–32.

[Lind, 2011b] Lind, M. (2011b). Reasoning about causes and consequences in multilevel flow models,". ESREL.

[Lind et al., 2011] Lind, M., Yoshikawa, H., Jørgensen, S. B., Yang, M., Tamayama, K., and Okusa, K. (2011). Multilevel flow modeling of monju nuclear power plant. *Nuclear safety and simulation*, 2(3) :274–284.

[Lüttgen and Mendler, 2002] Lüttgen, G. and Mendler, M. (2002). The intuitionism behind statecharts steps. *ACM Transactions on Computational Logic (TOCL)*, 3(1) :1–41.

[Machin, 2015] Machin, M. (2015). *Synthèse de règles de sécurité pour des systèmes autonomes critiques*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Maggiolo-Schettini et al., 2003] Maggiolo-Schettini, A., Peron, A., and Tini, S. (2003). A comparison of statecharts step semantics. *Theoretical Computer Science*, 290(1) :465–498.

[Marangé et al., 2011] Marangé, P., Pétin, J.-F., Manceaux, A., Gouyon, D., et al. (2011). Contribution à la reconfiguration des systèmes de production : ordonnancement par recherche d'atteignabilité. *Journal Européen des Systèmes Automatisés*, 45(1/3) :45–60.

[McMillan, 1993] McMillan, K. L. (1993). Symbolic model checking. *Symbolic Model Checking*, pages 25–60.

[Meinadier, 2009] Meinadier, J. (2009). Découvrir et comprendre l'ingénierie système-version expérimentale-version 3. *Association Française d'Ingénierie Système (AFIS)*.

[Mesli Kesraoui, 2017] Mesli Kesraoui, S. (2017). *Intégration des techniques de vérification formelle dans une approche de conception des systèmes de contrôle-commande : Application aux architectures SCADA*. PhD thesis, Université Bretagne Loire.

[Morel, 1992] Morel, G. (1992). *Contribution à l'automatisation et à l'ingénierie des systèmes intégrés de production*. Habilitation à diriger des recherches, Université H. Poincaré-Nancy I.

[Murata, 1989] Murata, T. (1989). Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541–580.

[Niang et al., 2017] Niang, M., Philippot, A., Gellot, F., Coupat, R., Riera, B., and Lefebvre, S. (2017). Formal verification for validation of pseel's plc program. In *The 14th International Conference on Informatics in Control, Automation and Robotics*, Madrid, Spain.

[O'Hara et al., 2000] O'Hara, J., Higgins, J., Stubler, W., and Kramer, J. (2000). Computer-based procedure systems : Technical basis and human factors review guidance (technical report no. nureg/cr-6634). *Nuclear Regulatory Commission, Washington, DC*.

[Pétin et al., 2007] Pétin, J.-F., Gouyon, D., and Morel, G. (2007). Supervisory synthesis for product-driven automation and its application to a flexible assembly cell. *Control Engineering Practice*, 15(5) :595–614.

[Pétin et al., 1998] Pétin, J.-F., Iung, B., and Morel, G. (1998). Distributed intelligent actuation and measurement (iam) system within an integrated shop-floor organisation. *Computers in Industry*, 37(3) :197–211.

[Petri, 1966] Petri, C. A. (1966). Communication with automata.

[Piriou, 2015] Piriou, P.-Y. (2015). *Contribution to model Based Safety Analysis for dynamic repairable reconfigurable systems*. Theses, Université Paris-Saclay.

[Pnueli, 1981] Pnueli, A. (1981). The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13 :45 – 60.

[Pnueli and Shalev, 1991] Pnueli, A. and Shalev, M. (1991). What is in a step : On the semantics of statecharts. In *Theoretical aspects of computer Software*, pages 244–264. Springer.

[Qiu, 2005] Qiu, R. G. (2005). Virtual production line based wip control for semiconductor manufacturing systems. *International Journal of Production Economics*, 95(2) :165–178.

[Ramadge and Wonham, 1987] Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1) :206–230.

[Rapin, 2014] Rapin, N. (2014). Artimon un outil de monitoring de propriétés de logique temporisée. *Génie Logiciel*, pages 26–31.

[Rene and Hassane, 2005] Rene, D. and Hassane, A. (2005). Discrete, continuous, and hybrid petri nets. *NY : Springer-Verlag*.

[Rivas and Rudd, 1974] Rivas, J. R. and Rudd, D. F. (1974). Synthesis of failure-safe operations. *AIChE Journal*, 20(2) :320–325.

[Ruel, 2009] Ruel, S. (2009). *Évaluation des bornes des performances temporelles des Architectures d'Automatisation en Réseau par preuves itératives de propriétés logiques*. Theses, École normale supérieure de Cachan-ENS Cachan.

[Saleem and Lind, 2009] Saleem, A. and Lind, M. (2009). Reasoning about control situations in power systems. In *Intelligent System Applications to Power Systems, 2009. ISAP'09. 15th International Conference on*, pages 1–6. IEEE.

[Viswanathan et al., 1998a] Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V., and Ärzen, K. E. (1998a). Automating operating procedure synthesis for batch processes : Part i. knowledge representation and planning framework. *Computers & chemical engineering*, 22(11) :1673–1685.

[Viswanathan et al., 1998b] Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V., and Ärzen, K. E. (1998b). Automating operating procedure synthesis for batch processes : Part ii. implementation and application. *Computers & chemical engineering*, 22(11) :1687–1698.

[Vogel, 1988] Vogel, C. (1988). *Génie cognitif*. Masson.

[Walker and Stine, 2004] Walker, J. S. and Stine, J. K. (2004). *Three Mile Island*. C-SPAN Archives.

[Wang et al., 2005] Wang, Y.-F., Chou, H.-H., and Chang, C.-T. (2005). Generation of batch operating procedures for multiple material-transfer tasks with petri nets. *Computers & chemical engineering*, 29(8) :1822–1836.

[Williams, 1994] Williams, T. J. (1994). The purdue enterprise reference architecture. *Computers in industry*, 24(2-3) :141–158.

[Yeh and Chang, 2012] Yeh, M.-L. and Chang, C.-T. (2012). An automata-based approach to synthesize untimed operating procedures in batch chemical processes. *Korean Journal of Chemical Engineering*, 29(5) :583–594.

[Zaytoon and Riera, 2017] Zaytoon, J. and Riera, B. (2017). Synthesis and implementation of logic controllers–a review. *Annual Reviews in Control*.

Résumé

Les travaux présentés dans ce manuscrit portent sur la conduite de systèmes complexes critiques. Ils s'inscrivent dans le cadre du projet CONNEXION (Investissements d'Avenir, BGLE2) qui réunit les principaux acteurs de la filière nucléaire française autour de la conception des systèmes de contrôle-commande des centrales et de leur exploitation. Dans le domaine de la conduite, les actions développées par le projet concernent la phase d'ingénierie avec pour objectif d'intégrer le point de vue de l'exploitant au plus tôt dans la validation des architectures de contrôle de commande, et la phase d'exploitation avec pour objectif de fournir une aide à la préparation et à l'exécution des procédures de conduite.

Dans ce contexte, la contribution présentée dans ce mémoire porte sur la génération et la vérification de séquences d'actions de conduite répondant à un objectif donné et pouvant être opérées en toute sécurité sur le procédé. L'approche proposée repose la vérification d'une propriété d'atteignabilité sur un réseau d'automates temporisés modélisant le comportement des architectures. L'originalité réside dans la définition d'un cadre formel de modélisation sous la forme de patrons favorisant la réutilisabilité des modèles ainsi que dans la proposition d'algorithmes d'abstraction et de recherche d'atteignabilité itératifs exploitant la hiérarchisation intrinsèque des architectures afin de permettre le passage à l'échelle de l'approche proposée.

La contribution a été éprouvée sur la plate-forme d'expérimentation CISPI du CRAN puis sur un cas d'étude à échelle industrielle proposé dans le cadre du projet CONNEXION.

Mots-clés: conduite de systèmes complexes critiques, aide à la conduite, génération itérative de séquences d'actions de conduite, recherche d'atteignabilité, automates à états temporisés

Abstract

The works presented in this manuscript deals with critical complex systems operation. They are part of the CONNEXION project (Investissements d'Avenir, BGLE2), which involves the main actors in the French nuclear industry around the design of control systems for power plants and their operation. In the operation field, the actions developed by the project concern the engineering phase with the aim of integrating the operator's point of view as soon as possible in the validation of control architectures, and the operation phase with the aim of providing assistance in the preparation and execution of operation procedures.

In this context, the contribution presented in this manuscript deals with the generation and verification of action sequences that meet a given objective and that can be safely operated on the process. The proposed approach relies on verifying a reachability property on a network of timed automata modelling the behavior of architectures. The originality is in the definition of a formal modelling framework using patterns promoting the reusability of models, as well as in the proposition of abstraction and reachability iterative analysis algorithms exploiting the intrinsic hierarchization of architectures in order to scale-up of the proposed approach.

The contribution was evaluated on the CISPI experimental platform of the CRAN, and on an industrial scale case study proposed within the framework of the CONNEXION project.

Keywords: critical complex systems operation, operation aid, operation sequences iterative generation, reachability analysis, timed automata