# Expressivité, satisfiabilité et model checking d'une logique spatiale pour arbres non ordonnés

## THÈSE



présentée et soutenue publiquement le 22 juin 2006

pour l'obtention du

**Doctorat de l'Université des Sciences et Technologies de Lille**

**(spécialité informatique)**

par

Iovka BONEVA

**Composition du jury**

| | | |
|---|---|---|
| *Président :* | Pierre BOULET, Professeur | Université de Lille I |
| *Rapporteurs :* | Bruno COURCELLE, Professeur | Université de Bordeaux I |
| | Denis LUGIEZ, Professeur | Université de Provence |
| | Wolfgang THOMAS, Professeur | RWTH Aachen |
| *Examinateurs :* | Luc SEGOUFIN, Chargé de recherche | INRIA |
| *Directeurs :* | Sophie TISON, Professeur | Université de Lille I |
| | Jean-Marc TALBOT, Maître de conférences | Université de Lille I |

**UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE**
Laboratoire d'Informatique Fondamentale de Lille — UPRESA 8022
U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. : +33 (0)3 28 77 85 41 – Télécopie : +33 (0)3 28 77 85 37 – email : direction@lifl.fr

Mis en page et composé avec LaTeX.

# Remerciements

Trouver les bons mots pour dire merci est chose difficile, d'autant plus dans une langue qui n'est pas sa langue natale.

Je remercie ma directrice de thèse Sophie Tison et mon encadrant Jean-Marc Talbot, d'abord pour l'encadrement scientifique – le travail en équipe, les intuitions, les conseils et suggestions – sans lequel cette thèse ne serait pas, mais aussi pour nos rapports humains simples et chaleureux.

Je remercie Jean-Marc pour son investissement dans l'encadrement de ma thèse et de m'avoir guidée jour après jour dans l'apprentissage du métier de chercheur. Merci aussi pour sa patience, en particulier les jours où nos avis divergeaient.

Je remercie Sophie qui, malgré un emploi du temps chargé, a toujours trouvé du temps à me consacrer. J'ai grandement apprécié la confiance qu'elle m'a accordée.

Merci à Bruno Courcelle, Denis Lugiez et Wolfgang Thomas de m'avoir fait l'honneur de rapporter cette thèse. Merci également à Luc Ségoufin et Pierre Boulet de participer au jury de thèse.

Pendant les années de thèse, j'ai beaucoup apprécié l'ambiance amicale qui règne au LIFL. Je tiens à remercier plus particulièrement les membres de l'équipe STC pour les nombreux moments passées à discuter de travail, ou de la pluie et du beau temps, ainsi que les membres de l'équipe MOSTRARE pour les séminaires et réunions dans une ambiance détendue. Merci aux enseignants et personnels du département informatique de l'IUT A pour leur accueil pendant mes trois années de monitorat.

Merci également aux nombreux doctorants du LIFL avec qui on a pu discuter des joies et des malheurs de la vie de thésard. Merci à Mirabelle, Raphaël, Cédric et Denis pour les pauses passés ensemble et leurs encouragements lors des moments de doutes.

Je pense aujourd'hui avec beaucoup de tendresse à mes parents, à la confiance qu'ils m'accordent et qui, malgré la tristesse de voir leur fille partir à l'étranger, m'ont toujours soutenue et encouragée.

Je suis très reconnaissante à Claire et Jean, Mélina et Elsa, pour leur généreux accueil et leur amitié. Merci également à Yana pour son amitié.

J'ai aujourd'hui une pensée pour mes amis, d'ici et de là-bas, et pour les différents professeurs m'ayant donné le goût des sciences.

Merci à Damien.

# Introduction

L'objet de cette thèse est de donner différentes caractérisations formelles d'une logique d'arbres que nous appelons ici *logique spatiale*. Les motivations initiales de ce travail viennent de la possibilité d'utiliser la logique spatiale en tant que langage d'interrogation de documents semi structurés et typage de documents semi structurés. Ces motivations ont défini les problèmes auxquels nous nous sommes intéressés principalement, à savoir la complexité du model checking et la satisfiabilité de la logique, ainsi que de certaines restrictions de celle-ci. Les motivations secondaires viennent des particularités de cette logique par rapport au cadre d'application envisagé. D'une part, l'interprétation d'une logique spatiale sur des arbres est moins étudiée, en général, que sur d'autres objets tels que les processus mobiles, les tas, les graphes. D'autre part, cette logique s'interprète sur un modèle d'arbres *non ordonnés*, qui est un modèle pour les données semi structurées et documents XML relativement peu étudié. Toutes ces particularités nous ont amenés à nous intéresser plus à ce qu'on peut appeler le pouvoir d'expression de la logique (pour le modèle d'arbres considéré). Nous rassemblons sous ce terme différentes caractérisations comme la décidabilité de la logique, les types de propriétés qu'elle permet d'exprimer ou non, le coût d'une vérification automatisée de ces propriétés, ainsi qu'une comparaison avec des logiques plus « classiques » sur les arbres.

### Les arbres non ordonnés

Nous considérons dans cette thèse des arbres finis étiquetés sur les arêtes par les symboles d'un alphabet dénombrable. Il n'y a pas de borne *a priori* du nombre de fils qu'un nœud de l'arbre peut avoir, nous parlons alors d'arbre *d'arité non bornée*. De plus, il n'y a pas d'ordre entre les fils d'un nœud de l'arbre : la seule relation entre nœuds qui est prise en compte étant la relation père-fils ; nous parlons alors d'arbres *non ordonnés*.

Peu de travaux existent sur des formalismes d'arbres non ordonnés (par rapport aux arbres ordonnés) ; on peut citer la logique CMSO [Courcelle, 1990b] qui capture la notion de reconnaissabilité algébrique pour les arbres non ordonnés ; la logique à traits et les automates à traits[1] [Niehren and Podelski, 1993] qui mettent également en correspondance une notion de définissabilité et de reconnaissabilité sur les arbres non ordonnés ; dans [Colcombet, 2002], une notion d'automates à multiensembles[2] a été introduite pour capturer la notion d'ensembles de termes rationnels modulo l'associativité et la commutativité ; les travaux [Ohsaki, 2001] et [Ohsaki and Takai, 2002] introduisent les automates d'arbres équationnels qui reconnaissent des termes modulo des théories équationnelles et en particulier modulo l'associativité et la commutativité ; les automates à multiarbres [Lugiez, 2003] sont des automates d'arbres qui utilisent

[1] features logic, features automata

[2] multiset automata

des contraintes numériques. A ces formalismes nous pouvons ajouter la logique monadique du second ordre (MSO) sur les graphes qui peut également être interprétée sur des arbres.

L'intérêt grandissant sur la modélisation et l'exploitation des données semi structurées a suscité quelques autres travaux sur les arbres non ordonnés, bien que le modèle prédominant soit celui des arbres ordonnés. Dans ce sens, on peut identifier deux courants. Le premier, qui contient un plus grand nombre de travaux, est celui qui s'est intéressé à la logique des ambients[Cardelli and Gordon, 2000a], ou plutôt à son fragment statique. Le calcul des ambients [Cardelli and Gordon, 2000b] est une algèbre de processus. Les termes de cette algèbre, appelés processus, ont une structure d'arbre qui peut changer dans le temps au fur et à mesure que le processus effectue des calculs. La logique des ambients permet de décrire des propriétés sur ces processus et leur évolution ; le fragment statique de la logique ne s'intéresse qu'à la structure des processus, c'est donc une logique (spatiale) pour arbres. Divers fragments et variantes du fragment statique de la logique des ambients ont été étudiés en vue de la modélisation, l'interrogation et le typage des documents structurés [Cardelli and Ghelli, 2004], [Calcagno et al., 2003], [Dal Zilio and Lugiez, 2003], [Dal Zilio et al., 2004]. Le second groupe de travaux considère une extension de la logique monadique du second ordre avec des contraintes de comptage, représentées par des formules de l'arithmétique de Presburger [Seidl et al., 2003], [Seidl et al., 2004] ; il s'agit de la logique PMSO. On peut également citer, en dehors de ces deux groupes de travaux, l'existence de quelques résultats sur l'utilisation des logiques temporelles pour des arbres non ordonnés [Barcelo and Libkin, 2005]. Une étude de différentes logique pour arbres d'arité non bornée a été effectuée récemment dans [Libkin, 2005].

### La logique spatiale

Une logique est dite *spatiale* lorsqu'elle possède un opérateur de séparation ou de composition. Cet opérateur permet d'exprimer qu'une structure peut être séparée en deux parties, chacune de ces deux parties pouvant être caractérisée indépendamment de l'autre (par une formule logique). Dans les processus du calcul des ambients par exemple, cette composition est utilisée pour exprimer l'exécution en parallèle de deux processus [Cardelli and Gordon, 2000a]. Sur les graphes, la séparation consiste à partitionner l'ensemble des arêtes du graphe en deux et de considérer les deux graphes définis par chacun des sous ensembles d'arêtes [Cardelli et al., 2002], [Dawar et al., 2004].

La logique que nous étudions ici s'inscrit dans le premier groupe de logiques mentionné ci-dessus, càd les logiques issues de la logique des ambients et appliquées aux documents semi-structurés. Plus précisément, nous étudions la logique introduite dans [Cardelli and Ghelli, 2004] et nous l'appelons *logique spatiale*. Cette logique a été initialement introduite comme la base d'un langage de requêtes pour données semi structurées, le langage TQL [Conforti et al., 2002b]. Elle représente essentiellement le fragment statique de la logique des ambients enrichi avec un opérateur de point fixe et un opérateur de quantification sur les arbres. En plus des connecteurs booléens classiques (disjonction, négation, vérité), cette logique comprend des opérateurs *spatiaux*, de la *quantification* et un mécanisme de *récursion* par l'intermédiaire d'un opérateur de point fixe. Les opérateurs spatiaux permettent de parler de la structure de l'arbre : $a[\phi]$ exprime le fait que l'arbre a une seule arête partant de la racines et qui mène vers un sous arbre satisfaisant $\phi$ ; $\phi_1 \mid \phi_2$ exprime que l'arbre peut être obtenu en fusionnant les racine de deux arbres, l'un satisfaisant $\phi_1$ et l'autre satisfaisant $\phi_2$. Ce dernier opérateur est appelé *composition* et représente une des particularité des logiques spatiales. La quantification dans la logique permet de quantifier sur des étiquettes et sur des arbres. Quant à l'opérateur de point fixe, il ajoute un

mécanisme de récursion permettant, par exemple, d'exprimer des régularités dans les chemins de l'arbre, des répétitions d'un motif exact ou approché dans l'arbre, d'exprimer que « quelque part dans l'arbre » $\phi$ est vérifié, ou bien « partout où $\phi_1$ est vérifié, $\phi_2$ l'est aussi » et beaucoup d'autres.

Grâce à la multitude d'opérateurs, cette logique spatiale est très expressive. On peut considérer qu'elle englobe tous les fragments et variantes de la logique des ambients qui ont été cités plus haut, et qui ont été étudiés pour une application aux données semi-structurés. Par ailleurs, certains résultats connus pour la logique des ambients s'appliquent directement à la logique spatiale. Par exemple, nous savons que la satisfiabilité de cette logique n'est pas décidable en présence de quantification [Charatonik and Talbot, 2001, Charatonik et al., 2003]. De plus, le langage TQL dispose d'une implémentation [Conforti et al., 2002a], ce qui laisse croire que le problème de model checking de la logique est décidable, au moins pour le fragment utilisé dans le langage de requêtes TQL.

Nous nous proposons ici de faire une étude de la logique spatiale qui se veut systématique en ce qui concerne deux problèmes particuliers : le problème de satisfiabilité et le problème de model checking. Cette étude est systématique parce que nous considérons divers fragments de la logique qui ont parfois été introduits par ailleurs, et pour chacun de ses fragment nous donnons des résultats sur la satisfiabilité et le model checking. Nous faisons également le rapprochement avec la logique monadique du second ordre et la logique PMSO.

**Les problèmes de satisfiabilité et de model checking**

Une des premières question qu'on se pose lorsqu'on étudie une logique est le problème de satisfiabilité. Le problème de satisfiabilité pour la logique spatiale est le problème de décision suivant : étant donné une formule logique $\phi$, décider s'il existe un arbre qui satisfait cette formule. Lorsque ce problème est décidable, nous disons que la logique est décidable.

D'un point de vue théorique, la décidabilité est en quelque sorte une mesure pour l'expressivité d'une logique : les logiques pouvant exprimer des propriétés complexes ont plus de chances d'être indécidables. D'un point de vue plus pragmatique, la satisfiabilité d'une logique permet de résoudre le problème d'implication d'une formule par une autre et l'équivalence de deux formules.

Le problème de *model checking* pour la logique spatiale est le problème suivant : étant donné une formule logique $\phi$ et un arbre $t$, décider si $t$ satisfait la formule $\phi$. Nous nous intéressons d'une part à la décidabilité de ce problème, et d'autre part à sa complexité lorsqu'il est décidable.

Dans le contexte des bases de données, le model checking peut être utilisé pour l'évaluation de ce qu'on appelle requêtes booléennes, qui reviennent à vérifier si une base de donnée, ou un document semi structuré, satisfait une contrainte. On peut également utiliser le model checking pour tester si un ou plusieurs objets (arbres, étiquettes) sont le résultat d'une requête.

En ce qui concerne la complexité, on a deux manières d'évaluer la complexité du model checking. La première, appelée *complexité combinée*, est la complexité du problème dans le sens classique, en considérant l'arbre et la formule comme données du problème. La seconde, appelée *complexité de données*, est la complexité du problème en considérant seul l'arbre comme donnée du problème. La complexité de données est une mesure plus pertinente pour la praticabilité d'une logique en tant que langage de requêtes. En effet, dans une utilisation pratique, la taille de l'arbre (de la donnée) est beaucoup plus grande que la taille de la formule, cette dernière est alors considérée comme une constante.

**Plan de la thèse**

*Première partie : les bases* La première partie est consacrée essentiellement à l'introduction des objets de base que nous allons considérer tout au long de la thèse, notamment le modèle d'arbres non ordonnés d'arité non bornée que nous adoptons, ainsi que deux représentations de ces arbres. Nous introduisons également la classe d'automates à contraintes numériques, qui nous serviront comme outil pour établir une grande partie des résultats de la thèse. Nous étudions certaines propriétés essentielles de ces automates : déterminisation, clôture par les opérations booléennes, test d'appartenance et test du vide. Nous introduisons finalement la logique spatiale, ainsi que la logique monadique du second ordre et son extension, la logique PMSO.

*Deuxème partie : satisfiabilité* La deuxième partie est consacrée à l'étude du problème de satisfiabilité de la logique spatiale sans quantification (étant donné que la logique avec quantification est connue pour être indécidable). Les deux principaux résultats de cette partie sont :

- montrer que la logique spatiale a un problème de satisfiabilité indécidable, même en absence de quantification ;
- définir un automate à contraintes numériques équivalent à une formule de la logique spatiale ;
- identifier deux fragments syntaxiques de la logique spatiale équivalents à la logique PMSO et à la logique MSO respectivement. Conjointement avec ces équivalences nous établissons que le problème de satisfiabilité est décidable pour ces deux fragments de la logique.

Pour établir les résultats ci-dessus, nous avons été amenés à introduire une représentation des formules de la logique spatiale comme des systèmes d'équations à point fixe.

*Troisième partie : model checking* La troisième partie est consacrée à l'étude du problème de model checking de la logique spatiale. Nous établissons que ce problème est décidable, en présentons un algorithme permettant de le résoudre. Cet algorithme requiert un espace polynomial pour être exécuté. Dans un deuxième temps, nous étudions les bornes inférieures de la complexité du model checking. Nous montrons que la complexité combinée est PSPACE-difficile, et que la complexité de données va de linéaire, à PSPACE-difficile, en fonction du fragment de la logique qu'on considère.

# Table des matières

# Première partie

# Préliminaires

# Chapitre 1

# Définitions de base

## 1.1 Multiensembles

Soient $P$ et $Q$ deux ensembles quelconques. Nous notons par $P^Q$ l'ensemble des applications de $Q$ dans $P$. Si $Q$ est l'ensemble fini $\{q_1, \ldots, q_n\}$, nous écrirons parfois $(q_1 \mapsto p_1, \ldots, q_n \mapsto p_n)$ pour désigner l'application dans $P^Q$ qui à $q_i$ associe $p_i \in P$ pour chaque $i$ dans $1..n$.

Soit $Q$ un ensemble. Un *multiensemble sur* $Q$ est une application de $Q$ dans $\mathbb{N}$, l'ensemble des entiers naturels. Donc, $\mathbb{N}^Q$ est l'ensemble des multiensembles sur $Q$. Le *multiensemble vide* sur $Q$ est l'application qui associe 0 à tout élément de $Q$ ; il sera noté $0^Q$. Le multiensemble $\omega$ sur $Q$ est dit *unitaire* s'il existe $q$ dans $Q$ tel que $\omega(q) = 1$ et pour tout $q'$ dans $Q$ différent de $q$, $\omega(q') = 0$ ; dans ce cas $\omega$ est noté $1_q$. Si $\omega$ est un multiensemble sur $Q$, pour tout $q$ dans $Q$, le nombre $\omega(q)$ est appelé *multiplicité de q dans* $\omega$.

L'inclusion de multiensembles a la définition attendue : soient $\omega$ et $\omega'$ des multiensembles sur $Q$. On dit que le multiensemble $\omega$ est *inclus* dans le multiensemble $\omega'$, noté $\omega \subseteq \omega'$, si pour tout $q$ dans $Q$, $\omega(q) \leq \omega'(q)$ ; $\omega$ est *strictement inclus* dans $\omega'$, noté $\omega \subset \omega'$, si $\omega$ est inclus dans $\omega'$ et différent de $\omega'$.

Un multiensemble $\omega$ est dit *fini* si l'ensemble $\{q \in Q \mid \omega(q) > 0\}$ est fini.

### Notations

Chaque multiensemble fini $\omega$ sur $Q$ peut être représenté comme une séquence d'éléments de $Q$, notée $seq(\omega)$, où chacun de ses éléments apparaît autant de fois que sa multiplicité dans $\omega$. Par exemple, si $q_1, q_2, q_3$ sont des éléments de $Q$ et $\omega$ est le multiensemble tel que $\omega(q_1) = 2$, $\omega(q_2) = 1$, $\omega(q_3) = 2$ et $\omega(q) = 0$ pour tout $q$ différent de $q_1, q_2, q_3$, alors $seq(\omega)$ est $q_1, q_1, q_2, q_3, q_3$ (ou toute autre séquence contenant les mêmes éléments dans un ordre différent). De façon générale, si $\omega$ est le multiensemble vide, alors $seq(\omega)$ est la séquence vide. Si $\omega$ n'est pas le multiensemble vide, $q \in Q$ est tel que $\omega(q) > 0$ et $\omega'$ est le multiensemble sur $Q$ tels que $\omega(q) = \omega'(q) + 1$ et pour tout $q' \neq q$, $\omega(q') = \omega'(q')$, alors $seq(\omega)$ est $q, seq(\omega')$. On écrira $\{\!|seq(\omega)|\!\}$ pour le multiensemble (fini) $\omega$, et donc $\{\!|\}\!\}$ pour le multiensemble vide. Notons que, sauf dans quelques cas rares facilement identifiables, cette façon d'écrire les multiensembles

sera utilisée pour la représentation des arbres non ordonnées et d'arité non bornée que nous considérons dans cette thèse (voir la section 1.3 ci-dessous).

## Opérations sur les multiensembles

L'*union de multiensembles* ou *somme de multiensembles* est l'opération binaire sur les multiensembles qui à deux multiensembles $\omega, \omega'$ sur $Q$ associe le multiensemble $\omega''$ sur $Q$ tel que pour tout $q$ dans $Q$, $\omega''(q) = \omega(q) + \omega'(q)$. Il est facile de voir que l'union de multiensembles est une opération associative et commutative d'élément neutre $0^Q$. Suivant l'utilisation des multiensembles, la somme de $\omega$ et $\omega'$ sera notée $\omega \uplus \omega'$ ou $\omega + \omega'$. Plus précisément, nous réservons la notation $\omega \uplus \omega'$ pour les arbres (voir la section 1.3 ci-dessous), et dans tous les autres cas nous utilisons la notation $\omega + \omega'$.

Si $\lambda$ est un entier naturel et $\omega$ est un multiensemble sur $Q$, le *produit* de $\lambda$ et $\omega$, noté $\lambda\omega$, est le multiensemble sur $Q$ tel que pour tout $q$ dans $Q$, $(\lambda\omega)(q) = \lambda\omega(q)$.

La somme et le produit sont étendus aux ensembles de multiensembles : soient $\Omega, \Omega'$ des sous-ensembles de $\mathbb{N}^Q$ et $\lambda$ un entier naturel. La somme de $\Omega$ et $\Omega'$, notée $\Omega + \Omega'$, est l'ensemble dans $\wp(\mathbb{N}^Q)$

$$\Omega + \Omega' = \{\omega + \omega' \mid \omega \in \Omega, \omega' \in \Omega'\}.$$

Le produit de $\lambda$ et $\Omega$, noté $\lambda\Omega$, est l'ensemble dans $\wp(\mathbb{N}^Q)$

$$\lambda\Omega = \{\lambda\omega \mid \omega \in \Omega\}.$$

Finalement, la somme duale $\Omega \dotplus \Omega'$ des deux ensembles du multiensembles $\Omega$ et $\Omega'$ est définie par

$$\Omega \dotplus \Omega' = \{\omega'' \mid \forall \omega, \omega'.\ \omega'' = \omega + \omega' \Rightarrow (\omega \in \Omega \text{ ou } \omega \in \Omega')\}.$$

## Classes d'ensembles de multiensembles

Par analogie avec les ensembles de vecteurs d'entiers sans étoile [Gaubert and Giua, 1999] et les ensembles de vecteurs d'entiers semilinéaires, nous introduisons les ensembles de multiensembles sans étoile et les ensembles de multiensembles semilinéaires.

**Définition 1.1 (Ensembles de multiensembles linéaires)** Soit $Q$ un ensemble fini non vide et soient $\mathbf{p}_1, \ldots, \mathbf{p}_k$ et $\mathbf{b}$ des multiensembles dans $\mathbb{N}^Q$. L'ensemble $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ est défini par :

$$\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k) = \{\mathbf{b} + \lambda_1\mathbf{p}_1 + \cdots + \lambda_k\mathbf{p}_k \mid \lambda_1, \ldots, \lambda_k \in \mathbb{N}\}.$$

Tout ensemble de multiensembles de la forme $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ est dit *ensemble linéaire*. Le multiensemble $\mathbf{b}$ est appelée *base* et les multiensembles $\mathbf{p}_1, \ldots, \mathbf{p}_k$ des *périodes*.

**Définition 1.2 (Ensembles de multiensembles sans étoile)** Soit $Q$ un ensemble fini non vide. Un sous-ensemble de $\mathbb{N}^Q$ est dit *sans étoile* s'il peut s'écrire comme une union finie d'ensembles linéaires dont les périodes sont des multiensembles unitaires : toute période est égale à $1_q$ pour un certain $q \in Q$.

Différentes autres définitions existent pour les ensembles sans étoile. En théorie des langages, un ensemble de mots est sans étoile s'il peut être défini à l'aide d'une expression régulière sans utiliser l'étoile et en utilisant le complémentaire. Un ensemble de vecteurs (de taille fini) est sans étoile s'il est fini ou bien s'il peut être obtenu par à partir d'ensembles de vecteurs sans étoile en utilisant les opérations booléennes. Ils existent également d'autres caractérisations équivalentes des ensembles sans étoile. Voir par exemple [Gaubert and Giua, 1999] où certaines de ces équivalences sont établies.

**Définition 1.3 (Ensembles de multiensembles semilinéaires)** Soit $Q$ un ensemble fini et non vide. Un sous ensemble de $\mathbb{N}^Q$ est dit *semilinéaire* s'il peut s'écrire comme une union finie d'ensembles linéaires.

Il est immédiat de ces définitions que tout ensemble de multiensembles sans étoile est également un ensemble semilinéaire.

Les résultats suivants, connus pour les ensembles de vecteurs sans étoile et semilinéaires, sont facilement adaptés aux ensembles de multiensembles.

**Définition 1.4 (Ensemble effectif)** Un ensemble sans étoile ou semilinéaire est dit *effectif* s'il est donné sous la forme d'une union finie d'ensemble linéaires, chacun de ces ensembles linéaires étant donne par sa base et ses périodes.

La notion d'ensemble effectif se réfère donc à la représentation de l'ensemble, et non pas à sa nature. Par exemple, un ensemble de multiensembles sans étoile pourrait être défini par une machine de Turing, ou par une fonction récursive qui énumère ses éléments, ou par une expression construite à partir d'ensembles finis et d'opérations booléennes. Aucune de ces trois représentations ne définit un ensemble effectif, même s'il pourrait être possible d'en déduire une représentation effective sous la forme d'union d'ensembles linéaires.

Terminons par ce fait bien connu sur les ensembles sans étoile et semilinéaires.

**Fait 1.1** Les ensembles sans étoile et les ensembles semilinéaires sont clos par union, intersection et complémentaire. De plus, si ces ensembles sont effectifs, alors les clôtures sont également effectives (càd l'union, l'intersection et le complémentaire peuvent être calculés et représentés par des unions finies d'ensembles linéaires donnés par leurs bases et leurs périodes).

## 1.2 Arithmétique de Presburger

L'arithmétique de Presburger est la théorie du premier ordre sur la structure $\langle \mathbb{N}, +, = \rangle$ des nombres entiers naturels avec l'addition. Soit $\mathcal{Z}$ un ensemble dénombrable de variables sur les entiers naturels ; les éléments de $\mathcal{Z}$ sont notés $z, z', \ldots$. Les formules de l'arithmétique de Presburger sont définies par la syntaxe :

$$\begin{aligned} p &::= v = v \mid \neg p \mid p \vee p \mid \exists z.p \\ v &::= n \mid z \mid v + v \end{aligned}$$

où $n$ désigne un nombre entier quelconque. Les sous-formules obtenues pour le non terminal $v$ sont appelés des *termes numériques*. L'ensemble des formules de l'arithmétique Presburger pouvant être construites sur l'ensemble de variables $\mathcal{Z}$ est noté $PresbForm(\mathcal{Z})$.

Les formules de l'arithmétique de Presburger sont interprétées sur la structure $\langle \mathbb{N}, +, = \rangle$. Soit $\zeta$ une valuation des variables, c'est à dire $\zeta$ est une application de $\mathcal{Z}$ dans $\mathbb{N}$. $\zeta$ est étendue aux termes numériques comme suit : si $v = n$ pour un certain entier naturel $n$, alors $\zeta(v) = n$ ; si $v = v_1 + v_2$, pour les termes numériques $v_1, v_2$, alors $\zeta(v) = \zeta(v_1) + \zeta(v_2)$. On dit que la valuation $\zeta$ est *solution* de la formule $p$, ou encore que $\zeta$ *satisfait* $p$, noté $\zeta \models p$, si :

- $p$ est $v = v'$ et $\zeta(v) = \zeta(v')$ ;
- $p$ est $\neg p'$ et $\zeta \not\models p'$ ;
- $p$ est $p' \vee p''$ et $\zeta \models p'$ ou $\zeta \models p''$ ;
- $p$ est $\exists z.p'$ et il existe un entier naturel $n$ tel que $\zeta[z \mapsto n] \models p'$.

Il est bien connu que l'ensemble des solutions d'une formule de Presburger est semilinéaire. Avec les notations ci-dessus, et remarquant que toute valuation $\zeta$ peut être assimilée à un multiensemble sur $\mathcal{Z}$, cette propriété s'exprime de la façon suivante :

**Fait 1.2** Soit $p(z_1, \ldots, z_k)$ une formule de l'arithmétique de Presburger à $k$ variables libres, précisément $z_1, \ldots, z_n$ ; soit $Z$ l'ensemble $\{z_1, \ldots, z_k\}$. Considérons l'ensemble de multiensembles $\Omega$ inclus dans $\mathbb{N}^{\mathcal{Z}}$ défini par :

$$\Omega = \left\{ \zeta \in \mathbb{N}^{\mathcal{Z}} \mid \zeta \models p(z_1, \ldots, z_n) \right\}.$$

L'ensemble $\Omega$ est semilinéaire.

## 1.3 Modèle d'arbres et représentations des arbres

Soit $\Sigma$ un ensemble dénombrable d'étiquettes. Les éléments de $\Sigma$ sont notés $a, b$ avec éventuellement un indice. Tout au long de ce document, nous considérons des structures arborescentes finies dont les arêtes sont étiquetés par des éléments de $\Sigma$.

Un *arbre* $t$ est un triplet $\langle N, E, \mathsf{etq} \rangle$, où $N$ est un ensemble fini non vide de nœuds, $E \subseteq N \times N$ est un ensemble fini d'arêtes et etq est une application de l'ensemble des arêtes $E$ vers l'ensemble d'étiquettes $\Sigma$ qui définit l'étiquetage des arêtes de l'arbre. Pour l'arête $e = (v, v')$ (avec $v, v'$ dans $N$), le nœud $v$ est appelé *source* de $e$ et le nœud $v'$ est appelé *destination* de $e$. De plus, l'ensemble d'arêtes $E$ garantit que l'arbre soit connecté et admet un nœud racine. Formellement, il existe un nœud particulier noté $\mathsf{racine}(t)$ tel que (i) $\mathsf{racine}(t)$ n'est la destination d'aucune arête dans $E$ et (ii) pour tout nœud $v$ différent de $\mathsf{racine}(t)$, il existe une unique suite d'arêtes $e_1, \ldots, e_k$ telle que la source de $e_1$ est $\mathsf{racine}(t)$, la destination de $e_k$ est $v$ et pour tout $i$ dans $1..k-1$ la destination de $e_i$ et la source de $e_{i+1}$ coïncident.

Deux arbres isomorphes sont considérés étant égaux. Nous notons $\mathcal{A}_\Sigma$ l'ensemble des arbres étiquetés par des étiquettes de l'ensemble $\Sigma$. Pour un nœud $v$ de l'arbre $t = \langle N, E, \mathsf{etq} \rangle$, les *nœuds successeurs* de $v$ sont les nœuds $v'$ tels que $(v, v') \in E$ et les *arêtes successeurs* de $v$ sont les arêtes $e'$ telles que $v$ est la source de $e'$. Pour l'arête $e$ de l'arbre $t = \langle N, E, \mathsf{etq} \rangle$, les *arêtes successeurs* de $e$ sont les arêtes $e'$ telles que la source de $e'$ et la destination de $e$ coïncident. L'*arbre vide* est l'unique arbre dont l'ensemble des nœuds est un ensemble réduit à un unique élément : la racine de l'arbre. Par conséquent, l'arbre vide n'a pas d'arêtes.

Pour illustrer certains exemples, nous utiliserons une représentation graphique des arbres, comme sur la figure 1.

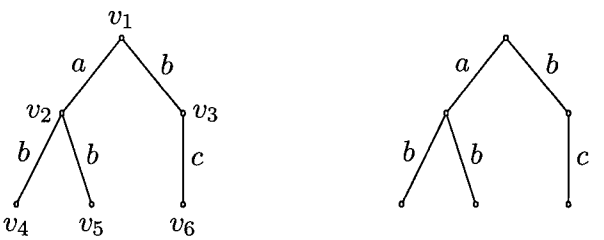



FIG. 1 – Exemple de représentations graphiques d'un arbre. A gauche, les nœuds ont été nommés.

**Exemple 1.1** Soit l'arbre $t = \langle N, E, \mathsf{etq} \rangle$ défini par :
- $N = \{v_1, v_2, v_3, v_4, v_5, v_6\}$,
- $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5), (v_3, v_6)\}$ et
- $\mathsf{etq}((v_1, v_2)) = a, \mathsf{etq}((v_1, v_3)) = b, \mathsf{etq}((v_2, v_4)) = b, \mathsf{etq}((v_2, v_5)) = b, \mathsf{etq}((v_3, v_6)) = c$,

où $a, b, c$ appartiennent à $\Sigma$. Une représentation graphique de l'arbre $t$ est donnée sur la figure 1.

Remarquons que le terme « arbre » est utilisé pour d'autres types de structures arborescentes. Très souvent, les arbres sont des termes construits sur une signature finie ; ceci revient à considérer que les nœuds de l'arbre sont étiquetés par les éléments de la signature. De plus, l'étiquette d'un nœud détermine le nombre de ses nœuds successeurs (qui est égale à l'arité de l'étiquette du nœud). Si la signature ne contient pas de symboles constants, alors l'arbre est infini. Ce type d'arbres peut être appelé « arbres ordonnés d'arité bornée ». Par analogie, nous pouvons également considérer des arbres ordonnés d'arité non borné. Ce sont des arbres pour lesquels le nombre de successeurs d'un nœud n'est pas borné *a priori* (par rapport aux « arbres termes », dans lesquels le nombre de successeurs d'un nœuds est borné, par la plus grande arité dans la signature, mais aussi fixé par l'étiquette du nœud). Dans un arbre ordonné d'arité non bornée, en plus de la relation père-fils entre les nœuds définie par les arêtes de l'arbre, on doit considérer une relation « frère suivant » qui défini un ordre total entre tous les nœds successeurs d'un nœud donné.

## Taille et hauteur d'un arbre

La *taille* de l'arbre $t = \langle N, E, \mathsf{etq} \rangle$, notée $taille(t)$ ou simplement $|t|$, est le cardinal de l'ensemble $N$ des nœuds de l'arbre. Notons que pour tout arbre, le nombre de ses nœuds est égal au nombre de ses arêtes plus un. Donc, la taille d'un arbre est égale au nombre de ses arêtes plus un.

La *hauteur* de l'arbre $t = \langle N, E, \mathsf{etq} \rangle$, notée $hauteur(t)$, est la longueur de la plus longue branche de l'arbre, càd la longueur de la plus longue suite d'arêtes $e_1, \ldots, e_k$ dans $E$ telles que $e_1$ est successeur de la racine de $t$, $e_k$ est sans successeurs dans $t$ et pour tout $i$ dans $1..k-1$, l'arête $e_{i+1}$ est successeur de l'arête $e_i$.

Nous donnons dans la suite deux représentations supplémentaires des arbres (en plus de la représentations par son ensemble de nœuds, d'arêtes et l'étiquetage des arêtes). Chacune de ces

représentations est mieux adaptée que l'autre pour l'interprétation de certaines logiques ou pour l'utilisation d'automates d'arbres.

### Les arbres comme structures logiques

On considère la signature $\sigma = \{<\} \cup \{\mathsf{etq}_b \mid b \in \Sigma\}$ où $<$ est un prédicat binaire (en notation infixée) et les $\mathsf{etq}_b$ sont des prédicat unaires. Alors à chaque arbre $t = \langle N, E, \mathsf{etq}\rangle$ dans $\mathcal{A}_\Sigma$ on peut associer la $\sigma$-structure finie $S^t = \langle E, \{\mathsf{etq}^t_b \mid b \in \Sigma\}, <^t\rangle$ où, pour chaque $b$ dans $\Sigma$ et chaque arête $e$ dans $E$, $\mathsf{etq}^t_b(e)$ est vrai dans $S^t$ si $\mathsf{etq}(e) = b$ et pour chaque couple d'arêtes $e, e'$ dans $E$, $e <^t e'$ est vrai si la destination de l'arête $e$ et la source de l'arête $e'$ coïncident.

Il est à noter que la structure logique que nous proposons ici est orientée arêtes, c'est à dire les éléments de base de $S^t$ sont les arêtes de l'arbre alors que dans la littérature, les arbres (et les graphes en général) sont plus souvent représentés par des structures logiques orientées nœuds. Notre choix sera discuté dans la section 3.1 ci-dessous.

Il existe une bijection entre les arbres et les structures logiques.

### Les arbres comme multiensembles imbriqués

Soit $\Sigma$ un ensemble dénombrable, et soient $\mathsf{Arbres}_\Sigma$ et $\mathsf{EArbres}_\Sigma$ les ensembles définis récursivement comme :

- $\mathsf{Arbres}_\Sigma$ est l'ensemble des multiensembles sur $\mathsf{EArbres}_\Sigma$ ;
- $\mathsf{EArbres}_\Sigma$ est l'ensemble des $a[t]$ où $a$ est élément de $\Sigma$ et $t$ est élément de $\mathsf{Arbres}_\Sigma$.

Les éléments de $\mathsf{EArbres}_\Sigma$ sont appelés *éléments d'arbre*.

**Fait 1.3** Il existe une bijection entre les ensembles $\mathcal{A}_\Sigma$ et $\mathsf{Arbres}_\Sigma$.

Montrons d'abord comment passer d'un arbre vers sa représentation sous forme de multiensembles imbriqués. Soit l'arbre $t = \langle N, E, \mathsf{etq}\rangle$, et construisons $s \in \mathsf{Arbres}_\Sigma$ qui représente l'arbre $t$. A chaque nœud $v$ dans $N$ nous associons un multiensemble d'éléments d'arbres $s_v$. Chacun de ces multiensembles contient exactement les éléments d'arbres $a[s_{v'}]$ tels que $(v, v')$ est une arête de l'arbre $t$ et $\mathsf{etq}(v, v') = a$.

Inversement, étant donné l'objet $s$ dans $\mathsf{Arbres}_\Sigma$, construisons l'arbre $t = \langle N, E, \mathsf{etq}\rangle$ dont $s$ est la représentation sous forme de multiensembles imbriqués. Le cardinal de l'ensemble de nœuds $N$ est égal au nombre de multiensembles apparaissant dans $s$. (On associe deux nœuds différents à deux occurrences différentes du même multiensemble apparaissant dans $s$. Par exemple, il y a autant de nœuds dans $N$ correspondant au multiensemble $\{\!\{\}\!\}$ que d'occurrences de ce multiensemble dans $s$.) Les arêtes de $s$ sont isomorphes aux éléments d'arbres apparaissant dans $s$. Pour tout élément d'arbre $e = a[s']$ apparaissant dans $s$, et pour toute occurrence de cet élément d'arbre dans $s$, il y a une arête $(v'', v')$ dans $E$ et $\mathsf{etq}(v'', v') = a$, où $v'$ est le nœud correspondant au multiensemble $s'$ et $v''$ est le nœud correspondant au multiensemble dans $s$ qui contient l'élément $e$.

Le passage d'une représentation à l'autre est linéaire dans la taille de l'arbre. De plus, la taille et la hauteur d'un arbre $t$ peuvent être déterminées à partir de sa représentation sous forme

de multiensembles imbriqués $s$. La taille est égale au nombre de multiensembles qui apparaissent dans $s$, la hauteur est la plus grande profondeur d'imbrication de multiensembles. Plus précisément, pour tout $s \in \mathsf{Arbres}_\Sigma$, $taille(s)$ et $hauteur(s)$ sont récursivement définies sur la structure de $s$ par :

- $taille(\{\!\{\}\!\}) = 1$ ;
- $taille(\{\!\{a_1[s_1], \ldots, a_k[s_k]\}\!\}) = 1 + \sum_{i \in 1..k} taille(s_i)$

et

- $hauteur(\{\!\{\}\!\}) = 0$ ;
- $hauteur(\{\!\{a_1[s_1], \ldots, a_k[s_k]\}\!\}) = 1 + \max_{i \in 1..k} taille(s_i)$.

**Exemple 1.2** La représentation multiensembliste de l'arbre de la figure 1 est

$$\{\!\{a[\{\!\{b[\{\!\{\}\!\}], b[\{\!\{\}\!\}]\}\!\}], b[\{\!\{c[\{\!\{\}\!\}]\}\!\}]\}\!\}.$$

Suite à cette correspondance entre l'ensemble des arbres $\mathcal{A}_\Sigma$ et l'ensemble $\mathsf{Arbres}_\Sigma$, les éléments de ce dernier sont appelés simplement arbres.

## Notations

Nous terminons cette section par quelques notations.

Pour tout ensemble d'étiquettes $\alpha$ et tous ensembles d'arbres $T, T'$, $\alpha[T]$ et $T \mid T'$ sont les ensembles d'arbres

$$\alpha[T] = \{a[t] \mid a \in \alpha,\ t \in T\}$$
$$T \mid T' = \left\{t \uplus t' \mid t \in T,\ t' \in T'\right\}.$$

Autrement dit, $\mid$ est une notation alternative de la somme d'ensembles de multiensembles qui sera utilisée uniquement pour les ensembles d'arbres. De ce fait, l'opérateur $\mid$ est associatif et commutatif : pour tous ensembles d'arbres $T, T', T''$, on a $T \mid T' = T' \mid T$ et $(T \mid T') \mid T'' = T \mid (T' \mid T'')$. Il est également facile à voir que pour tous ensembles d'étiquettes $\alpha, \beta$ et tous ensembles d'arbres $T, T'$, on a

$$\alpha[T] \cap \beta[T'] = (\alpha \cap \beta)[T \cap T'].$$

Par définition, l'ensemble $\alpha[T]$ est vide si et seulement si $\alpha$ est vide ou $T$ est vide.

Pour tout entier naturel $n$, tout élément d'arbre $e$ et tout ensemble d'éléments d'arbres $E$, $n \cdot e$ et $n \cdot E$ sont respectivement l'arbre et l'ensemble d'arbres définis par :

$$n \cdot e = \{\!\{\underbrace{e, \ldots, e}_{n \text{ fois}}\}\!\}$$
$$n \cdot E = \{\{\!\{e_1, \ldots, e_n\}\!\} \mid \forall i \in 1..n,\ e_i \in E\}$$

Notons qu'alors pour tout ensemble d'arbres $E$, $0 \cdot E = \{\!\{\}\!\}$ et que pour tout entier naturel strictement positif $n$, $n \cdot \varnothing = \varnothing$. Ceci implique en particulier que $n \cdot \varnothing$ n'est pas l'ensemble vide seulement si $n$ est égal à 0.

# Chapitre 2

# Automates

Sont présentés ici deux classes d'automates d'arbres et leurs propriétés. Considérer un formalisme d'automates peut s'avérer utile dans une multitude de cas, aussi bien pour résoudre des problèmes pratique que pour établir des résultats théoriques. Pour une étude large et approfondie de diverses problématiques liées aux automates d'arbres, le lecteur peut consulter [Comon et al., 1997]. Comme nous le verrons par la suite, les classes d'automates d'arbres que nous présentons ici vont nous permettre d'établir des résultats de décidabilité, de complexité et de comparaison d'expressivité pour la logique spatiale.

Les automates à contraintes numériques tels que nous les définissons ici (section 2.2) n'ont pas été considérés auparavant sous cette forme exactement. Cependant, on ne peut pas vraiment considérer que c'est une nouvelle classe d'automates. Divers types d'automates pour arbres non ordonnés ont été définis, plus ou moins proches des automates à contraintes numériques. On peut citer les automates à traits[1] [Niehren and Podelski, 1993] définis à travers une reconnaissabilité algébrique modulo associativité et commutativité ; les automates d'arbres conditionnels[2] [Lugiez and Moysset, 1994] travaillent sur une représentation multiensembliste des arbres et utilisent des contraintes de comptage pour les transitions ; les automates d'arbres équationnels[3] [Ohsaki, 2001], [Verma, 2003] reconnaissent des termes modulo des théories équationnelles, et en particulier modulo l'associativité et la commutativité ; les « automates à multiensembles rationnels » [Colcombet, 2002] utilisent dans les transitions des contraintes de comptage définies par des ensembles de multiensembles reconnaissables. Finalement, dans le domaine des données semi-structurées ont été récemment définis, d'une part, les automates à multiarbres[4] et leur variante les automates à faisceaux[5] [Lugiez and Dal Zilio, 2002] [Dal Zilio and Lugiez, 2002] [Dal Zilio and Lugiez, 2003] [Dal Zilio et al., 2004] qui utilisent des contraintes numériques exprimées dans l'arithmétique de Presburger, et font le lient avec (un fragment de) la logique spatiale (voir aussi la section 6.1) et, d'autre part, les automates de Presburger [Seidl et al., 2003] utilisant également des contraintes de l'arithmétique de Presburger.

[1] *feature automata*
[2] *conditional tree automata*
[3] *equational tree automata*
[4] *multitree automata*
[5] *sheaves automata*

Les automates de Presburger (section 2.1) ont été introduits dans [Seidl et al., 2003]. Nous donnons brièvement leur définition et quelques propriétés. Les automates à contraintes numériques sont présentés dans la section 2.2.

Les deux dernières sections du chapitre sont consacrées à l'étude des propriétés des automates à contraintes numériques, telles que déterminisation, clôture par les opérations booléennes, clôture par homomorphisme, décidabilité et complexité du test du vide et du test d'appartenance. La section 2.3 s'intéresse aux automates à contraintes numériques en général, et la section 2.4 s'intéresse à deux sous classes des automates à contraintes numériques.

## 2.1 Automates de Presburger

Les automates de Presburger ont été définis dans [Seidl et al., 2003] par Seidl, Schwentick et Muscholl dans et des variantes de ces automates ont été étudiés dans [Seidl et al., 2004]. Dans ces travaux, les auteurs considèrent des arbres d'arité non bornée, ordonnés ou non, et étiquetés sur les nœuds par des étiquettes d'un alphabet **fini**.

Un automate de Presburger est un tuple $A = (\Sigma, Q, \Delta, p)$ où $\Sigma$ est un ensemble fini de symboles appelé alphabet, $Q$ est un ensemble fini d'états, $p$ est une formule dans $PresbForm(\{z_q \mid q \in Q\})$ et $\Delta$ est une fonction de $Q \times \Sigma$ dans $PresbForm(\{z_q \mid q \in Q\})$.[6]

**Définition 2.1 (Langage d'un automate de Presburger)** Soit $A = (\Sigma, Q, \Delta, p)$ un automate de Presburger. Pour tout état $q$ de l'automate, $\mathcal{L}_A(q) \subseteq \mathsf{EArbres}_\Sigma$ est le *langage associé à l'état* $q$. Pour toute formule $p'$ dans $PresbForm(\{z_q \mid q \in Q\})$, $\mathcal{L}_A(p') \subseteq \mathcal{A}_\Sigma$ est le *langage associé à la formule* $p'$. $\mathcal{L}_A(q)$ et $\mathcal{L}_A(p')$ sont récursivement définis par :

- $\mathcal{L}_A(q)$ est l'ensemble d'éléments d'arbres

$$\bigcup_{\Delta(q,a)=p''} a[\mathcal{L}_A(p'')];$$

- soit $Q = \{q_1, \ldots, q_k\}$, alors $\mathcal{L}_A(p')$ est l'ensemble d'arbres

$$\bigcup_{\zeta \models p'} \zeta(z_{q_1}) \cdot \mathcal{L}_A(q_1) \mid \cdots \mid \zeta(z_{q_k}) \cdot \mathcal{L}_A(q_k).$$

Le *langage de l'automate* $A$, noté $\mathcal{L}(A)$, est l'ensemble d'arbres $\mathcal{L}_A(p)$.

Les automates de Presburger ont différentes bonne propriétés qui ont été étudiées dans [Seidl et al., 2003], dont certaines sont citées ci-dessous.

**Théorème 2.2 ([Seidl et al., 2003], Théorème 2)** *La famille des langages définissables par des automates de Presburger est effectivement close par union, intersection et complémentaire.*

**Théorème 2.3 ([Seidl et al., 2003], Théorème 1)** *Le vide d'un automate de Presburger est décidable.*

[6] Rappelons que $PresbForm(\{z_q \mid q \in Q\})$ est l'ensemble des formules de l'arithmétique de Presburger construites sur l'ensemble de variables $\{z_q \mid q \in Q\}$.

Un automate de Presburger $A$ est dit *déterministe* si pour tout élément d'arbre $e$, il existe un et un seul état de l'automate $q$ tel que $e$ appartient à $\mathcal{L}_A(q)$.

**Proposition 2.4 ([Seidl et al., 2003], Théorème 3)** *Pour tout automate de Presburger $A$, un automate de Presburger déterministe $A'$ peut être construit tel que $\mathcal{L}(A) = \mathcal{L}(A')$.*

## 2.2 Automates à contraintes numériques

Les automates à contraintes numériques que nous allons définir sont une extension des automates de Presburger à un alphabet $\Sigma$ dénombrable et où les contraintes de Presburger sont remplacées par des contraintes plus générales.

Un automate à contraintes numériques (ACN) $A$ est un quadruplet $(\Sigma, Q, \Delta, F)$ où $\Sigma$ est un ensemble de symboles dénombrable appelé alphabet de l'automate, $Q$ est un ensemble fini d'états, $\Delta$ est une relation sur $Q \times \wp(\Sigma) \times \wp(\mathbb{N}^Q)$ appelée relation de transition de l'automate Finalement, $F \subseteq \wp(\mathbb{N}^Q)$ est la condition d'acceptation de l'automate. Nous supposons que la relation de transition est finie (le graphe de la relation est fini).

Les triplets $(q, \alpha, D)$ appartenant à $\Delta$ sont également appelés *transitions de l'automate.*

**Définition 2.5 (Langage d'un automate)** Soit $A = (\Sigma, Q, \Delta, F)$ un automate à contraintes numériques. Pour tout état de l'automate $q$, $\mathcal{L}_A(q) \subseteq \mathsf{EArbres}_\Sigma$ est le *langage associé à l'état* $q$ et pour tout ensemble d'applications $D \subseteq \mathbb{N}^Q$, $\mathcal{L}_A(D) \subseteq \wp(\mathcal{A}_\Sigma)$ est le *langage associé à* $D$. $\mathcal{L}_A(q)$ et $\mathcal{L}_A(D)$ sont définis récursivement par :

- $\mathcal{L}_A(q)$ est l'ensemble d'élément d'arbres $\bigcup_{(q,\alpha,D)\in\Delta} \alpha[\mathcal{L}_A(D)]$ ;
- soit $Q = \{q_1, \ldots, q_n\}$, alors $\mathcal{L}_A(D)$ est l'ensemble d'arbres

$$\mathcal{L}_A(D) = \bigcup_{\mathbf{d}\in D} \mathbf{d}(q_1) \cdot \mathcal{L}_A(q_1) | \cdots | \mathbf{d}(q_n) \cdot \mathcal{L}_A(q_n).$$

Le *langage de* $A$, noté $\mathcal{L}(A)$, est l'ensemble d'arbres $\mathcal{L}_A(F)$.

**Remarque 2.6 (Forme des transitions d'un ACN)** *Soit l'automate $A = (\Sigma, Q, \Delta, F)$ et soient $(q, \alpha, D)$ et $(q, \alpha, D')$ deux transitions dans $\Delta$. Par définition de $\mathcal{L}(A)$, le langage de l'automate $A$, il est facile de voir que celui-ci ne change pas si les deux transitions $(q, \alpha, D)$ et $(q, \alpha, D')$ sont remplacées par la transition $(q, \alpha, D \cup D')$. Ceci peut être étendu à tout sous ensemble dénombrable de transition de $\Delta$ qui ne diffèrent que par leur troisième composante. Plus précisément, si $\{(q, \alpha, D_i) \mid i \in I\}$ est sous ensemble de $\Delta$ (pour un certain ensemble dénombrable $I$), alors le langage de l'automate $A = (\Sigma, Q, \Delta, F)$ est égal au langage de l'automate $A' = (\Sigma, Q, \Delta, F)$, où $\Delta'$ est la relation $(\Delta \setminus \{(q, \alpha, D_i) \mid i \in I\}) \cup \{(q, \alpha, \bigcup_{i\in I} D_i)\}$.*

*Nous en concluons que tout automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ est équivalent à un automate à contraintes numériques $A = (\Sigma, Q, \Delta', F)$ où la relation $\Delta$ est fonctionnelle dans les deux premières composantes, càd si $(q, \alpha, D)$ et $(q, \alpha, D')$ appartiennent à $\Delta'$, alors $D = D'$.*

Si l'arbre $t$ appartient à $\mathcal{L}(A)$, on dit que l'automate $A$ *reconnaît* l'arbre $t$.[7]

[7]Ce terme vient de la méthode usuellement utilisée pour tester si un arbre appartient au langage d'un automate, à savoir calculer une exécution de l'automate sur l'arbre. Voir la section suivante sur ce point.

**Exemple 2.1** Nous présentons ici plusieurs exemples d'automates à contraintes numériques. Ce que nous considérons ici comme la hauteur d'une arête $e$ est la hauteur de l'arbre $\{\!|e|\!\}$ constitué de cette unique arête.

(i) Automate $A = (\Sigma, Q, \Delta, F)$ reconnaissant tous les arbres de hauteur un : $Q = \{q\}$, $\Delta = \{(q, \Sigma, \{0^Q\})\}$ et $F = \{(q \mapsto n) \mid n \geq 1\}$. L'idée ici est que $\mathcal{L}_A(q)$ est l'ensemble de tous les éléments d'arbres de hauteur un, càd $\mathcal{L}_A(q) = \Sigma[\{\!|\}\!|\}]$. La condition d'acceptation $F$ exprime le fait qu'un arbre $t$ est de hauteur un si et seulement si $t = n \cdot \Sigma[\{\!|\}\!|\}]$ et $n \geq 1$.

(ii) Tout automate $A = (\Sigma, Q, \Delta, F)$ avec $F = \{0^Q\}$ reconnaît uniquement l'arbre $\{\!|\}\!|$, ceci indépendemment de $Q$ et $\Delta$.

(iii) Construisons un automate $A = (\Sigma, Q, \Delta, F)$ reconnaissant le singleton $\{t\}$ où $t$ est l'arbre $\{\!|a[\{\!|a[\{\!|\}\!|], b[\{\!|\}\!|]\}\!|], b[\{\!|b[\{\!|\}\!|], b[\{\!|\}\!|]\}\!|]\}\!|$ ou, de façon visuelle, l'arbre représenté sur la figure 1.

L'idée est d'introduire un état $q_e$ par élément d'arbre $e$ différent dans $t$ et de construire $\Delta$ de façon à ce que pour tout $e$, $\mathcal{L}_A(q_e) = \{e\}$. Les éléments d'arbres différents dans $t$ sont $e_1 = a[\{\!|\}\!|]$, $e_2 = b[\{\!|\}\!|]$, $e_3 = a[\{\!|e_1, e_2|\!\}]$ et $e_4 = b[\{\!|e_2, e_2|\!\}]$. Donc, $Q = \{q_{e_1}, q_{e_2}, q_{e_3}, q_{e_4}\}$. La fonction de transition $\Delta$ est définie par :

$$\Delta = \left\{ \begin{array}{lll} (q_{e_1}, & \{a\}, & \{0^Q\}), \\ (q_{e_2}, & \{b\}, & \{0^Q\}), \\ (q_{e_3}, & \{a\}, & \{(q_{e_1} \mapsto 1, q_{e_2} \mapsto 1, q_{e_3} \mapsto 0, q_{e_4} \mapsto 0)\}), \\ (q_{e_4}, & \{b\}, & \{(q_{e_1} \mapsto 0, q_{e_2} \mapsto 2, q_{e_3} \mapsto 0, q_{e_4} \mapsto 0)\}) \end{array} \right\}$$

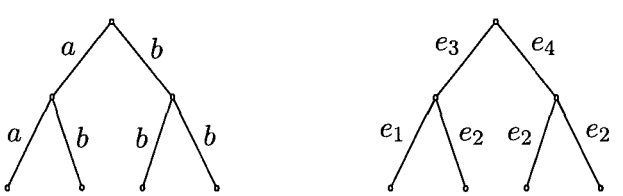



FIG. 1 – A gauche, une représentation graphique de l'arbre $t =$ et à droite une représentation graphique du même arbre sur laquelle chacune des arêtes est annotée par l'élément d'arbre qu'elle définit : $e_1 = a[\{\!|\}\!|]$, $e_2 = b[\{\!|\}\!|]$, $e_3 = a[\{\!|e_1, e_2|\!\}]$ et $e_4 = b[\{\!|e_2, e_2, e_2|\!\}]$.

(iv) Construisons un automate $A = (\Sigma, Q, \Delta, F)$ reconnaissant l'ensemble des arbres binaires complets de hauteur deux (càd ayant la même structure que l'arbre $t$ représenté sur la figure 1). L'idée est d'introduire un état $q_f$ par structure différente d'élément d'arbre apparaissant dans $t$ (la structure de $t$ excepté) et de garantir par $\Delta$ que pour toute structure, $\mathcal{L}_A(q_f)$ reconnaît exactement les arbres ayant cette structure. Les structures différentes rencontrées sont une seule arête, à laquelle on fait correspondre l'état $q_1$, et une arête avec deux successeurs qui sont elles-mêmes sans successeurs, à laquelle on fait correspondre l'état $q_2$. Donc, nous obtenons l'automate suivant : $Q = \{q_1, q_2\}$, $\Delta = \{(q_1, \Sigma, \{0^Q\}), (q_2, \Sigma, \{(q_1 \mapsto 2, q_2 \mapsto 0)\})\}$ et $F = \{(q_1 \mapsto 0, q_2 \mapsto 2)\}$.

(v) Le langage de l'automate $A = (\Sigma, Q, \Delta, F)$ défini dans ce qui suit est l'ensemble des arbres dont le nombre d'arêtes successeurs de la racine est premier : $Q = \{q\}$, $\Delta$ contient comme unique transition $(q, \Sigma, \mathbb{N}^Q)$ et $F = \{(q \mapsto n) \mid n \text{ nombre premier}\}$.

(vi) Le langage de l'automate $A = (\Sigma, Q, \Delta, F)$ défini dans ce qui suit est l'ensemble des arbres plats (de hauteur un) pour lesquels le nombre d'arêtes successeurs de la racine étiquetées par $a$ est pair et le nombre d'arêtes successeurs de la racine étiquetées par $b$ est impair : $Q = \{q_a, q_b, q\}$, $\Delta$ contient les transitions $(q_a, \{a\}, \{0^Q\})$, $(q_b, \{b\}, \{0^Q\})$, $(q, \overline{\{a, b\}}, \{0^Q\})$ et $F = \{(q_a \mapsto n_a, q_b \mapsto n_b, q \mapsto n) \mid n_a \text{ est pair}, n_b \text{ est impair}, n \in \mathbb{N}\}$. L'idée ici est que l'ensemble d'états de l'automate doit permettre de faire la distinction entre les éléments d'arbres (de hauteur un) étiquetés par $a, b$ ou une autre étiquette.

(vii) Le langage de l'automate $A = (\Sigma, Q, \Delta, F)$ défini dans ce qui suit est l'ensemble des arbres plats (de hauteur un) pour lesquels les arêtes successeurs de la racine peuvent être partitionnés en deux multiensemble de taille égale, le premier contenant uniquement des éléments d'arbres étiquetés par $a$ ou $b$ et le second contenant uniquement des éléments d'arbres étiquetés par $a$ ou $c$. Autrement dit, $t$ appartient à $\mathcal{L}(A)$ si et seulement si $t = \{\!|e_1, \ldots, e_n|\!\} \uplus \{\!|e'_1, \ldots, e'_n|\!\}$ et pour tout $i$ dans $1..n$, $e_i$ appartient à $\{a, b\}[\mathcal{A}_\Sigma]$ et $e'_i$ appartient à $\{a, c\}[\mathcal{A}_\Sigma]$. Alors $Q = \{q_b, q_c\}$ et la fonction de transition assure que $\mathcal{L}_A(q_b) = \{a, b\}[\mathcal{A}_\Sigma]$ et $\mathcal{L}_A(q_c) = \{a, c\}[\mathcal{A}_\Sigma]$. Donc, $\Delta$ contient les transitions $(q_b, \{a, b\}, \{0^Q\})$ et $(q_c, \{a, c\}, \{0^Q\})$. La condition d'acceptation est $F = \{(q_b \mapsto n, q_c \mapsto n) \mid n \in \mathbb{N}\}$.

(viii) Le langage de l'automate $A = (\Sigma, Q, \Delta, F)$ défini dans ce qui suit est l'ensemble des arbres binaires pour lesquels chaque nœud a zéro ou deux arêtes successeurs et dans ce dernier cas l'une de ces arêtes est étiquetée par $a$ et l'autre par $b$ : $Q = \{q_a, q_b\}$, $\Delta$ contient les transitions $(q_a, \{a\}, D)$ et $(q_b, \{b\}, D)$ où $D = \{(q_a \mapsto 0, q_b \mapsto 0), (q_a \mapsto 1, q_b \mapsto 1)\}$ et $F = D$.

(ix) Le langage de l'automate $A = (\Sigma, Q, \Delta, F)$ défini dans ce qui suit est l'ensemble des arbres dont tous les nœuds ont autant d'arêtes successeurs étiquetées par $a$ que d'arêtes successeurs étiquetées par $b$ : $Q = \{q_a, q_b\}$, $\Delta$ contient les transitions $(q_a, \{a\}, D)$, $(q_b, \{b\}, D)$ où $D = \{(q_a \mapsto n, q_b \mapsto n) \mid n \in \mathbb{N}\}$ et $F = D$.

(x) Les deux derniers exemples illustrent comment on peut définir des propriétés faisant intervenir la hauteur des arbres. Tout d'abord, présentons un automate $A = (\Sigma, Q, \Delta, F)$ de langage l'ensemble des arbres dont la racine a au plus quatre arêtes successeurs de hauteur supérieure à deux : $Q = \{q_1, q_2\}$, $\Delta$ contient les transitions $(q_1, \Sigma, \{0^Q\})$, $(q_2, \Sigma, \{(q_1 \mapsto n_1, q_2 \mapsto n_2) \mid n_1 \geq 1, n_2 \in \mathbb{N}\})$ et $F = \{(q_1 \mapsto n_1, q_2 \mapsto n_2) \mid n_2 \leq 4, n_1 \in \mathbb{N}\}$. Ici, $\mathcal{L}_A(q_1)$ est l'ensemble des éléments d'arbres de hauteur un et $\mathcal{L}_A(q_2)$ est l'ensemble des éléments d'arbres de hauteur supérieure ou égale à deux.

(xi) Finalement, l'automate $A = (\Sigma, Q, \Delta, F)$ reconnaît les arbres pour qui toutes les arêtes à profondeur impaire (en commençant à compter à 1 pour le niveau sous la racine) sont étiquetées par $a$ et il n'y a pas de restriction pour les autres arêtes. L'ensemble d'états de l'automate est $Q = \{q_{aa}, q_{\overline{a}}, q_a, q_{\text{err}}\}$. L'ensemble des transitions de l'automate est $(q, \alpha, D)$, où $q, \alpha, D$ sont donnés par les lignes du tableau ci-dessous, où $m_1, m_2, m_3$ désignent n'importe quel nombre naturel (positif ou nul), et $n_1, n_2$ sont des nombres stric-

tement positifs.

| | $D$ | $\alpha$ | $q$ |
|---|---|---|---|
| 1 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto 0, q_a \mapsto 0, q_{\text{err}} \mapsto 0)\}$ | $\{a\}$ | $q_{aa}$ |
| 2 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto 0, q_a \mapsto 0, q_{\text{err}} \mapsto 0)\}$ | $\Sigma \setminus \{a\}$ | $q_{\overline{a}}$ |
| 3 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto n_1, q_a \mapsto 0, q_{\text{err}} \mapsto 0)\}$ | $\Sigma \setminus \{a\}$ | $q_{\text{err}}$ |
| 4 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto n_1, q_a \mapsto 0, q_{\text{err}} \mapsto 0)\}$ | $\{a\}$ | $q_a$ |
| 5 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto n_1, q_a \mapsto n_2, q_{\text{err}} \mapsto 0)\}$ | $\Sigma$ | $q_{\text{err}}$ |
| 6 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto 0, q_a \mapsto n_1, q_{\text{err}} \mapsto 0)\}$ | $\Sigma$ | $q_{\overline{a}}$ |
| 7 | $\{(q_{aa} \mapsto m_1, q_{\overline{a}} \mapsto m_2, q_a \mapsto m_3, q_{\text{err}} \mapsto n_1)\}$ | $\Sigma$ | $q_{\text{err}}$ |

L'ensemble d'acceptation $F$ est $\{(q_{aa} \to m_1, q_{\overline{a}} \to 0, q_a \mapsto m_3, q_{\text{err}} \mapsto 0) \mid m_1, m_3 \in \mathbb{N}\}$. Pour une présentation intuitive de l'ensemble d'états et de règles de cet automate, nous utilisons la notion d'exécution définie dans la section suivante. Soit $r$ une exécution de l'automate. L'état $q_{aa}$ va être associé par $r$ aux arêtes définissant un élément d'arbre ne contenant que l'étiquette $a$. L'état $q_{\overline{a}}$ sera associé aux arêtes qui ne sont pas étiquetées par $a$ et dont toutes les arêtes successeurs et leur descendants sont étiquetées par $a$ (règle 2). Une arête à laquelle $r$ associe l'état $q_{\overline{a}}$ définit une succession d'états $q_a$ et $q_{\overline{a}}$ sur le chemin entre cette arête et la racine de l'arbre. Ainsi, si une arête a une arête successeur à laquelle $r$ associe l'état $q_{\overline{a}}$, elle doit obligatoirement être étiquetée par $a$ dans l'arbre (les règles 3 et 4) et inversement, si $r$ associe $q_a$ ou $q_{aa}$ à toutes les arêtes successeurs d'une arête, alors celle-ci peut porter n'importe quelle étiquette et se trouve sur un niveau $q_{\overline{a}}$ (la règle 6). Finalement, la règle 5 interdit le mélange d'états $q_a$ et $q_{\overline{a}}$ au même niveau, et la règle 7 assure la propagations de l'état d'erreur vers le haut.

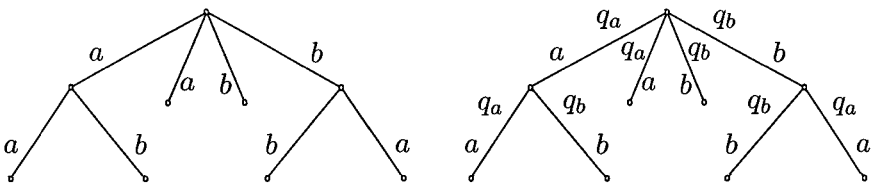
### 2.2.1 Exécution d'un automate et calcul d'une exécution

Soit $t$ un arbre et $A = (\Sigma, Q, \Delta, F)$ un automate à contraintes numériques. Une *exécution*[8] de l'automate $A$ sur l'arbre $t$ est une application $r$ qui à toute arête de $t$ associe un état de l'automate $A$ et telle que, si $r(e) = q$ et $e'$ est l'élément d'arbre défini par l'arête $e$, alors $e'$ appartient à $\mathcal{L}_A(q)$. D'après la définition de $\mathcal{L}_A(q)$, ceci est équivalent à dire qu'il existe une transition $(q, \alpha, D)$ de l'automate telle que l'étiquette de l'arête $e$ appartient à $\alpha$ et le multiensemble $\{\!|r(e_1), \ldots, r(e_k)|\!\}$ appartient à $D$, où $\{e_1, \ldots, e_k\}$ est l'ensemble des arêtes successeurs de $e$. Une exécution est dite *acceptante* si le multiensemble $\{\!|r(e_1), \ldots, r(e_k)|\!\}$ appartient à $F$, pour $\{e_1, \ldots, e_k\}$ étant l'ensemble des arêtes successeurs de la racine. Il est facile de voir que l'arbre $t$ appartient au langage de l'automate $A$ si et seulement s'il existe une exécution acceptante de $A$ sur $t$.

**Exemple 2.2** Considérons l'automate $A$ défini dans l'exemple 2.1 (ix) ; $A$ reconnaît l'ensemble des arbres dont tous les nœuds ont autant d'arêtes successeurs étiquetées par $a$ que d'arêtes successeurs étiquetées par $b$. Rappelons que $A = (\Sigma, Q, \Delta, F)$ et $Q = \{q_a, q_b\}$, $\Delta$ contient les transitions $(q_a, \{a\}, D)$, $(q_b, \{b\}, D)$ où $D = \{(q_a \mapsto n, q_b \mapsto n) \mid n \in \mathbb{N}\}$ et $F = D$.

Une exécution de cet automate sur l'arbre $t$ $\{\!|a[\{\!|\}\!|], b[\{\!|\}\!|], a[\{\!|a[\{\!|\}\!|], b[\{\!|\}\!|]\}\!|], b[\{\!|a[\{\!|\}\!|], b[\{\!|\}\!|]\}\!|]\}\!|$ (à gauche) est présentée ci-dessous (à droite) :

[8] Nous utilisons ici le mot « exécution » comme traduction du mot anglais « run » habituellement utilisé dans ce cas.




Remarquons que cette exécution est acceptante car le multiensemble $(q_a \mapsto 2, q_b \mapsto 2)$ appartient à la condition d'acceptation de l'automate.

Comme d'habitude, tester l'existence d'une exécution acceptante de l'automate $A$ pour l'arbre $t$ est une méthode pour tester si $t$ appartient au langage de $A$. Naturellement, pour faire ce test il suffit d'énumérer toutes les exécutions possibles et tester si l'une d'entre elles est acceptante (faisable pour des arbres finis et des automates à nombre fini d'états). Mais la méthode plus astucieuse est de construire directement une exécution acceptante ; on dit alors qu'on *calcule* une exécution. Il y a usuellement deux manières pour calculer une exécution acceptante : en commençant par les feuilles de l'arbre ou en commençant par la racine. Dans le premier cas, on parle d'une exécution *ascendante*, et dans le second d'une exécution *descendante*. Plus précisément, pour le calcul d'une exécution ascendante on commence par affecter des états aux arêtes sans successeurs et on continue à affecter des états en « remontant » dans l'arbre. Pour le calcul d'une exécution descendante on commence par affecter des états aux arêtes successeurs de la racine et on affecte les états aux autres arêtes en « descendant » dans l'arbre.

**Calcul d'une exécution ascendante**

Comme nous l'avons remarqué, étant donné un automate $A$ et un arbre $t$, dire qu'il existe une exécution acceptante de $A$ pour $t$ est équivalent à dire que l'arbre $t$ est reconnu par l'automate $A$. Ceci nous donne donc une méthode pour tester si un arbre $t$ appartient au langage d'un automate : il suffit de tester l'existence d'une exécution acceptante. Dans ce qui suit nous présentons une méthode de construction d'une exécution ascendante acceptante, lorsque celle-ci existe.

Considérons l'automate $A = (\Sigma, Q, \Delta, F)$ et l'arbre $t$. Soit $r$ l'exécution que nous sommes en train de calculer ; initialement $r(e)$ est indéfini pour toute arête $e$ de l'arbre $t$. Les arêtes dont on calcule l'image par $r$ en premier sont les arêtes sans successeurs ; soit $e$ une telle arête. Si $\mathrm{etq}(e) = a$, alors pour toute transition de l'automate de la forme $(q, \alpha, D)$ avec $a \in \alpha$ et $0^Q \in D$, on peut définir $r(e)$ comme étant $q$. Donc, pour pouvoir calculer $r(e)$ lorsque $e$ est une arête sans successeurs, il suffit de pouvoir identifier toutes les transitions de l'automate $(q, \alpha, D)$ pour lesquelles $0^Q$ appartient à $D$ et $\mathrm{etq}(e)$ appartient à $\alpha$. Soit maintenant $e$ une arête telle que pour toute arête $e'$ successeur de $e$, $r(e')$ a été calculé et soit $a$ l'étiquette de $e$. Considérons le multiensemble $\mathbf{d} = \{\!|r(e_1), \ldots, r(e_k)|\!\}$, où $\{e_1, \ldots, e_k\}$ est l'ensemble des arêtes successeurs de $e$. Alors pour toute transition de l'automate $(q, \alpha, D)$ telle que $a \in \alpha$ et $\mathbf{d} \in D$, l'état $q$ est une valeur correcte pour $r(e)$. Donc, pour pouvoir calculer $r(e)$ pour une arête $e$ quelconque, on doit pourvoir identifier les transitions de l'automate $(q, \alpha, D)$ telles que $a \in \alpha$ et $\mathbf{d} \in D$, ceci

pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$. Il est facile de voir que ces deux conditions sont également suffisantes pour décider si une exécution est acceptante. Nous définissons ci-dessous la notion d'automate à exécution calculable qui garantit que ces deux conditions sont satisfaites. Notons que cette notion fait référence à la **représentation** d'un automate, et non pas à l'automate lui même (notamment, représentation de la relation de transition et représentation des ensembles de multiensembles apparaissant dans les transitions de l'automate).

**Définition 2.7 (Automate à exécution calculable)** L'automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ est dit *à exécution calculable* si

(i) les triplets $(q, \alpha, D)$ appartenant à $\Delta$ peuvent être effectivement énumérés ;

(ii) pour toute étiquette $a$ et tout ensemble d'étiquettes $\alpha$ tel que $(q, \alpha, D)$ est une transition de l'automate, on peut décider si $a$ appartient à $\alpha$ et

(iii) pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$ et pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^Q$ dans

$$\{F\} \cup \{D \mid (q, \alpha, D) \text{ dans } \Delta\}$$

on peut décider si $\mathbf{d}$ appartient à $D$.

Remarquons que la première de ces conditions sera probablement toujours satisfaite. Par contre, suivant le type d'ensembles d'étiquettes et d'ensembles de multiensembles apparaissant dans les transitions de l'automate et de leur représentation, les deux autres conditions ne seront pas forcément satisfaites.

Notons également que ces conditions sont aussi valables pour tester si une exécution (arbitraire) de l'automate $A$ sur l'arbre $t$ est une exécution acceptante.

**Lemme 2.8** *Pour tout automate à exécution calculable $A$ et tout arbre $t$, il est décidable si $t$ appartient à $\mathcal{L}(A)$.*

*Preuve* On sait que $t$ appartient à $\mathcal{L}(A)$ si et seulement s'il existe une exécution acceptante de $A$ sur $t$. Il suffit alors de calculer toutes les exécutions de $A$ sur $t$ et de tester pour chacune de ces exécutions si elle est acceptante.

□

### 2.2.2 Automate déterministe et automate complet

Comme pour d'autres types d'automates, les notions d'automate déterministe et d'automate complet peuvent être définies pour les automates à contraintes numériques.

**Définition 2.9 (Automate déterministe)** L'automate à contraintes numériques $A$ est dit *déterministe* si pour tout couple d'états $q, q'$, on a $\mathcal{L}_A(q) \cap \mathcal{L}_A(q') = \varnothing$.

**Définition 2.10 (Automate complet)** L'automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ est dit *complet* si $\bigcup_{q \in Q} = \mathsf{EArbres}_\Sigma$.

Si $A$ est un automate déterministe, alors pour tout arbre $t$, il existe au plus une exécution de $A$ pour $t$. Si $A$ est complet, il existe au moins une exécution de $A$ pour $t$.

### 2.2.3 Sous classes d'automates à contraintes numériques

Dans la définition des automates à contraintes numériques nous n'avons posé aucune restriction ni sur les ensembles d'étiquettes, ni sur les ensembles de multiensembles qui peuvent apparaître dans une transition de l'automate. En particulier, on pourrait même imaginer des automates dans lesquels la relation de transition utilise des ensembles non récursivement énumérables. Naturellement, l'intérêt de tels automates est limité. Dans ce qui suit nous définissons deux sous-classes d'automates à contraintes numériques obtenues par des restrictions sur les ensembles d'étiquettes et les ensembles de multiensembles pouvant être utilisés dans la relation de transition. Comme nous le verrons plus tard dans cet ouvrage, les deux classes que nous présentons ici ont différentes propriétés intéressantes, par exemple l'appartenance et le test du vide sont décidables, ces classes sont closes par les opérations booléennes, les automates de ces classes peuvent être déterminisés.

**Définition 2.11 (Automates à contraintes numériques sans étoile)** L'automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ est dit *à contraintes sans étoile*, ou simplement *sans étoile*, si pour tout triplet $(q, \alpha, D)$ appartenant à la relation de transition $\Delta$
- l'ensemble d'étiquettes $\alpha$ est fini ou cofini et
- l'ensemble de multiensembles $D$ est un ensemble sans étoile.

**Définition 2.12 (Automates à contraintes numériques semilinéaire)** L'automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ est dit *à contraintes semilinéaires*, ou simplement *semilinéaire* si pour tout triplet $(q, \alpha, D)$ appartenant à la relation de transition $\Delta$,
- l'ensemble d'étiquettes $\alpha$ est fini ou cofini et
- l'ensemble de multiensembles $D$ est un ensemble semilinéaire.

Sachant que les ensembles semilinéaires sont exactement les modèles des formules de Presburger (le fait 1.2), on en déduit que les automates à contraintes numériques définis sur un alphabet $\Sigma$ fini sont équivalents aux automates de Presburger, càd un langage d'arbres $L \subseteq \mathcal{A}_\Sigma$ pour $\Sigma$ fini est définissable par un automate à contraintes numériques si et seulement s'il est définissable par un automate de Presburger.

**Définition 2.13 (Automate effectif)** L'automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ sans étoile (respectivement semilinéaire) est dit *effectif* si tout ensemble de multiensembles $D$ tel que $(q, \alpha, D)$ est une transition de l'automate (pour un certain était $q \in Q$ et un certain ensemble d'étiquettes $\alpha$) est un ensemble effectif et la condition d'acceptation $F$ est un ensemble effectif.

Notons que pour qu'un ACN sans étoile ou semilinéaire effectif soit à exécution calculable, il suffit qu'on puisse énumérer les triplets $(q, \alpha, D)$ appartenant à sa relation de transition. Les deux autres conditions pour avoir un automate à exécution calculable sont garanties par la définition des ACN sans étoile et semilinéaires effectifs : d'une part, tout ensemble d'étiquettes $\alpha$ apparaissant dans une transition de l'automate est fini ou cofini et donc le test d'appartenance à $\alpha$ est décidable (pour une représentation « raisonnable » de $\alpha$) et d'autre part, le test d'appartenance à un ensemble de multiensembles sans étoile ou semilinéaire effectif est également décidable.

## 2.3 Propriétés des automates à contraintes numériques

Nous étudions ici les propriétés des automates à contraintes numériques telles la déterminisation, la clôture par les opérations booléennes et par homomorphisme, décidabilité du test du vide et du test d'appartenance et leur complexité.

Nous commençons par la déterminisation des ACN (dans la section 2.3.1) : nous établissons que tout ensemble d'arbres définissable par un ACN est aussi définissable par un ACN déterministe, dont nous présentons la définition. On dit alors que les ACN sont « déterminisables ». Ensuite nous établissons que les ACN sont clos par les opérations booléennes. Rappelons qu'une classe d'automate est close par union (respectivement par intersection et par complémentaire) si pour tout couple d'ensembles d'arbres $S, T$ définissables par un automate de cette classe, l'ensemble d'arbres $S \cup T$ (respectivement $S \cap T$ et $\complement S$) sont également définissables par un automate de cette classe. Nous montrons également que les ACN sont clos par les homomorphismes d'étiquettes (dans la section 2.3.3). Intuitivement, un homomorphisme d'étiquettes est une application de $\Sigma$ dans un ensemble d'étiquettes $\Sigma'$ étendue de manière canonique sur les arbres. Finalement, nous nous intéressons au test du vide (dans la section 2.3.4) et au test d'appartenance (dans la section 2.3.5) des ACN. Le problème du test du vide est, étant donné un automate, décider si le langage de cet automate est l'ensemble vide. Nous donnons une condition pour que le test du vide d'un ACN soit décidable ; cette condition porte sur les ensembles de multiensembles $D$ apparaissant dans les règles de l'automate. Nous proposons également un algorithme pour le test du vide, ce qui nous permet d'établir une borne maximale de la complexité du test du vide, lorsque le problème est décidable. Le problème du test d'appartenance est, étant donné un automate $A$ et un arbre $t$, tester si $t$ appartient au langage de $A$. Comme nous avons déjà noté, il suffit pour cela de tester l'existence d'une exécution acceptante de $A$ pour $t$. Nous définissons la notion d'*automate à exécution calculable*, qui donne des conditions sur la représentation de l'automate $A$ garantissant la possibilité de calculer une exécution acceptante de $A$ pour $t$, et donc de répondre au problème du test d'appartenance. Cela nous donne également une borne maximale de la complexité du test d'appartenance.

### 2.3.1 Déterminisation

Nous montrons ici que les automates à contraintes numériques sont « déterminisables » : pour tout automate à contraintes numériques $A$, il existe un automate à contraintes numériques déterministe $A_{\text{det}}$ tel que les langages de $A$ et $A_{\text{det}}$ coïncident. Pour faire la preuve, nous définissons un tel automate $A_{\text{det}}$.

Considérons pour la suite l'automate $A = (\Sigma, Q, \Delta, F)$. Nous allons construire l'automate déterministe $A_{\text{det}} = (\Sigma, Q_{\text{det}}, \Delta_{\text{det}}, F_{\text{det}})$ tel que $\mathcal{L}(A) = \mathcal{L}(A_{\text{det}})$.

Tout d'abord, l'ensemble des états de $A_{\text{det}}$ est l'ensemble des parties de $Q$ :

$$Q_{\text{det}} = \wp(Q).$$

Pour tout état $w$ dans $Q_{\text{det}}$, soit $\Theta_w \subseteq \mathsf{EArbres}_\Sigma$ l'ensemble d'éléments d'arbres défini par

$$\Theta_w = \bigcap_{q \in w} \mathcal{L}_A(q) \cap \bigcap_{q \in Q \setminus w} \mathsf{EArbres}_\Sigma \setminus \mathcal{L}_A(q). \tag{2.1}$$

Nous allons construire l'ensemble de transitions $\Delta_{\text{det}}$ de telle façon que pour tout $w \in Q_{\text{det}}$,

$$\mathcal{L}_{A_{\text{det}}}(w) = \Theta_w. \tag{2.2}$$

Il est immédiat que si (2.1) est vérifié, alors l'automate $A_{\text{det}}$ est déterministe.

Pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$, soit $H_{\mathbf{d}} \subseteq \mathbb{N}^{Q_{\text{det}}}$ l'ensemble défini par :

$$\mathbf{h} \in H_{\mathbf{d}} \text{ si et seulement si } \begin{cases} \forall w \in Q_{\text{det}}. \forall q \in w.\ \exists n_{w,q} \in \mathbb{N} \text{ tels que :} \\ \quad - \forall q \in Q.\ \left(\sum_{w \in Q_{\text{det}} \mid q \in w} n_{w,q}\right) = \mathbf{d}(q) \text{ et} \\ \quad - \forall w \in Q_{\text{det}}.\ \left(\sum_{q \in w} n_{w,q}\right) = \mathbf{h}(w) \end{cases}$$

Remarquons que cette définition nous garantit que $\mathbf{h}(\varnothing) = 0$ et donc $H_{\mathbf{d}}$ est un ensemble fini.

Pour la suite de la construction de l'automate déterministe, nous adoptons une notation. Soit $\Theta$ l'application de $Q_{\text{det}}$ dans $\wp(\text{EArbres}_\Sigma)$ qui à tout $w$ dans $Q_{\text{det}}$ associe l'ensemble $\Theta_w$ défini ci-dessus (2.1). Pour tout ensemble de multiensembles $H$ dans $\mathbb{N}^{Q_{\text{det}}}$, nous notons par $[\![H, \Theta]\!]$ l'ensemble d'arbres

$$[\![H, \Theta]\!] = \bigcup_{\mathbf{h} \in H} \mathbf{h}(w_1) \cdot \Theta(w_1) \mid \cdots \mid \mathbf{h}(w_k) \cdot \Theta(w_k)$$

où $\{w_1, \cdots, w_k\}$ est l'ensemble $Q_{\text{det}}$. (Rappelons que pour tout entier naturel $n$ et tout ensemble d'éléments d'arbres $T$, $n \cdot T$ est l'ensemble d'arbres $\{\{\!|e_1, \ldots, e_n|\!\} \mid e_i \in E\}$.)

**Lemme 2.14** *Pour tout $D \subseteq \mathbb{N}^Q$, soit $H_D = \bigcup_{\mathbf{d} \in D} H_{\mathbf{d}}$. Alors $\mathcal{L}_A(D) = [\![H_D, \Theta]\!]$.*

*Preuve* Soit $Q = \{q_1, \ldots, q_n\}$ et $Q_{\text{det}} = \{w_1, \ldots, w_k\}$. Supposons d'abord que l'arbre $t$ appartient à $\mathcal{L}_A(D)$, càd, par définition, il existe un multiensemble $\mathbf{d}$ dans $D$ tel que $t$ appartient à l'ensemble d'arbres

$$\mathbf{d}(q_1) \cdot \mathcal{L}_A(q_1) \mid \cdots \mid \mathbf{d}(q_n) \cdot \mathcal{L}_A(q_n).$$

Autrement dit, ils existent des éléments d'arbres $e_j^q$ pour tout $q \in Q$ et tout $j \in 1..\mathbf{d}(q)$ tels que

$$t = \{\!|e_1^{q_1}, \ldots, e_{\mathbf{d}(q_1)}^{q_1}|\!\} \uplus \cdots \uplus \{\!|e_1^{q_n}, \ldots, e_{\mathbf{d}(q_n)}^{q_n}|\!\} \tag{2.3}$$

et $e_j^q \in \mathcal{L}_A(q)$ pour tous $q, j$. Remarquons que, par définition des $\Theta_w$, pour tous $q, j$, l'élément d'arbre $e_j^q$ appartient à exactement un des $\Theta(w)$, pour $w$ dans $Q_{\text{det}}$ ; de plus, si $e_j^q$ appartient à $\Theta(w)$, alors nécessairement $q$ appartient à $w$. Pour tout $q$ dans $Q$ et pour tout $w$ dans $Q_{\text{det}}$ tel que $q \in w$, soit $t_{w,q}$ le multiensemble constitué des éléments d'arbres parmi $e_1^q, \ldots, e_{\mathbf{d}(q)}^q$ qui appartiennent à $\Theta(w)$ et soit $n_{w,q}$ le nombre d'éléments de $t_{u,q}$. Par définition des $\Theta(w)$ on sait que

($\star$) pour tous $w, q, w', q'$ tels que $q \in w$ et $q' \in w'$, les multiensembles $t_{w,q}$ et $t_{w',q'}$ ont des éléments en commun si et seulement si $w = w'$.

En utilisant la définition des $t_{w,q}$, on en déduit que

$$t = \uplus_{w \in Q_{\text{det}}, q \in w} t_{w,q}. \tag{2.4}$$

Soit maintenant $\mathbf{h}$ le multiensemble dans $\mathbb{N}^{Q_{\text{det}}}$ tel que pour tout $w$ dans $Q_{\text{det}}$, $\mathbf{h}(w) = \sum_{q \in w} n_{w,q}$. Maintenant, par définition des $t_{w,q}$ et $n_{w,q}$, on sait que pour tous $w, q$, le multiensemble (càd

l'arbre) $t_{w,q}$ appartient à $n_{w,q} \cdot \Theta(w)$. Donc, de (2.4) nous déduisons facilement que l'arbre $t$ appartient à

$$\mathbf{h}(w_1) \cdot \Theta(w_1) \,|\, \cdots \,|\, \mathbf{h}(w_k) \cdot \Theta(w_k). \tag{2.5}$$

Remarquons finalement que, par définition des $t_{w,q}$ et des $n_{w,q}$ et par ($\star$), pour tout $q$ dans $Q$, $\sum_{w \in Q_{\text{det}} | q \in w} n_{w,q} = \mathbf{d}(q)$. Il en suit que $\mathbf{h}$ appartient à $H_{\mathbf{d}}$ et donc à $H_D$, et de (2.5) nous déduisons que $t$ appartient à $[\![H_D, \Theta]\!]$.

Supposons maintenant que l'arbre $t$ appartient à $[\![H_D, \Theta]\!]$. Soit $\mathbf{h}$ le multiensemble dans $\mathbb{N}^{Q_{\text{det}}}$ tel que l'arbre $t$ appartient à l'ensemble d'arbres

$$\mathbf{h}(w_1) \cdot \Theta(w_1) \,|\, \cdots \,|\, \mathbf{h}(w_k) \cdot \Theta(w_k).$$

Autrement dit, l'arbre $t$ est de la forme

$$t = \{\!|e_1^{w_1}, \ldots, e_{\mathbf{h}(w_1)}^{w_1}|\!\} \uplus \cdots \uplus \{\!|e_1^{w_k}, \ldots, e_{\mathbf{h}(w_k)}^{w_k}|\!\} \tag{2.6}$$

où pour tous $w, j$, l'élément d'arbre $e_j^w$ appartient à $\Theta(w)$. Soit $\mathbf{d}$ le multiensemble dans $D$ tel que $\mathbf{h}$ appartient à $H_{\mathbf{d}}$. Par définition, nous savons qu'ils existent les entiers naturels $n_{w,q}$ pour tout $w$ dans $Q_{\text{det}}$ et pour tout $q$ dans $w$ tels que

$$\text{pour tout } q \text{ dans } Q, \quad \sum_{w \in Q_{\text{det}} \,|\, q \in w} n_{w,q} = \mathbf{d}(q) \tag{2.7}$$

$$\text{pour tout } w \text{ dans } Q_{\text{det}}, \quad \sum_{q \in w} n_{w,q} = \mathbf{h}(w) \tag{2.8}$$

Nous savons, par définition des $\Theta(w)$, que pour tout $w$ dans $Q_{\text{det}}$ et pour tout $q \in w$, l'ensemble $\Theta(w)$ est inclus dans $\mathcal{L}_A(q)$. Soient maintenant les éléments d'arbres $t_{w,q}$, pour tout $w$ dans $Q_{\text{det}}$ et pour tout $q$ dans $w$ tels que :

- pour tous $w, q$, $t_{w,q}$ est composé de $n_{w,q}$ éléments d'arbres parmi $e_1^w, \ldots, e_{\mathbf{h}(w)}^w$ ;
- pour tout $w$ dans $Q_{\text{det}}$, $\{\!|e_1^{w_1}, \ldots, e_{\mathbf{h}(w_1)}^{w_1}|\!\} = \uplus_{q \in w} t_{w,q}$.

Par définition des $n_{w,q}$ et par (2.8), ceci est toujours possible. Maintenant, par définition des $\Theta(w)$, on sait que pour tout $w$ dans $Q_{\text{det}}$ et pour tout $q$ dans $w$, chacun des éléments d'arbres composant $t_{w,q}$ appartient à $\mathcal{L}_A(q)$. Donc, par (2.7) nous déduisons que pour tout $q$ dans $Q$, l'arbre $\uplus_{w \in Q_{\text{det}} | q \in w} t_{w,q}$ est composé de $\mathbf{d}(q)$ éléments d'arbres et chacun de ces éléments d'arbres appartient à $\mathcal{L}_A(q)$. Autrement dit, l'arbre $t$ appartient à $\mathbf{d}(q_1) \cdot \mathcal{L}_A(q_1) \,|\, \cdots \,|\, \mathbf{d}(q_n) \cdot \mathcal{L}_A(q_n)$ et donc $t$ appartient à $\mathcal{L}_A(D)$.

□

Suite au lemme précédent, pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^Q$, soit $H_D \subseteq \mathbb{N}^{Q_{\text{det}}}$ l'ensemble de multiensembles tel que $\mathcal{L}_A(D) = [\![H_D, \Theta]\!]$. Par définition, pour tout $q \in Q$,

$$\mathcal{L}_A(q) = \bigcup_{(q,\alpha,D) \in \Delta} \alpha[\mathcal{L}_A(D)]$$

et donc, par le lemme 2.14, pour tout $q \in Q$,

$$\mathcal{L}_A(q) = \bigcup_{(q,\alpha,D) \in \Delta} \alpha[[\![H_D, \Theta]\!]].$$

Donc, par définition, pour tout $w \in Q_{\text{det}}$,

$$\Theta_w = \bigcap_{q \in Q \mid q \in w} \left( \bigcup_{(q,\alpha,D)\in\Delta} \alpha[[\![H_D, \Theta]\!]] \right) \cap \bigcap_{q \in Q \mid q \notin w} \mathsf{EArbres}_\Sigma \setminus \left( \bigcup_{(q,\alpha,D)\in\Delta} \alpha[[\![H_D, \Theta]\!]] \right).$$

Maintenant, pour tout $w \in Q_{\text{det}}$ et pour tout $q \notin w$,

$$\mathsf{EArbres}_\Sigma \setminus \left( \bigcup_{(q,\alpha,D)\in Delta} \alpha[[\![H_D, \Theta]\!]] \right) = \bigcap_{(q,\alpha,D)\in\Delta} \mathsf{EArbres}_\Sigma \setminus \alpha[[\![H_D, \Theta]\!]].$$

Il est facile de voir que pour tout ensemble d'étiquettes $\alpha$ et tout ensemble d'arbres $S$, $\mathsf{EArbres}_\Sigma \setminus \alpha[S] = \overline{\alpha}[\mathcal{A}_\Sigma] \cup \alpha[\mathcal{A} \setminus S]$. En utilisant cette dernière égalité et le lemme 6.10, nous déduisons que pour tout ensemble de multiensembles $D$, $\mathsf{EArbres}_\Sigma \setminus \alpha[[\![H_D, \Theta]\!]] = \overline{\alpha}[[\![\mathbb{N}^{Q_{\text{det}}}, \Theta]\!]] \cup \alpha[[\![\mathbb{N}^{Q_{\text{det}}} \setminus H_D, \Theta]\!]]$. Nous en déduisons que pour tout $w \in Q_{\text{det}}$,

$$\Theta_w =$$

$$\bigcap_{q \in Q \mid q \in w} \left( \bigcup_{(q,\alpha,D)\in\Delta} \alpha[[\![\mathbb{N}^{Q_{\text{det}}}, \Theta]\!]] \right) \cap \bigcap_{q \in Q \mid q \notin w} \bigcap_{(q,\alpha,D)\in\Delta} \left( \overline{\alpha}[[\![H_D, \Theta]\!]] \cup \alpha[[\![\mathbb{N}^{Q_{\text{det}}} \setminus H_D, \Theta]\!]] \right)$$

Dans cette expression, toutes les unions et intersections sont finies, donc l'expression peut être mise sous la forme d'une union d'intersections, et il est facile de voir que chacune des ces intersections est de la forme $\bigcap_{i\in I} \alpha_i[[\![H_i, \Theta]\!]]$, où $I$ est un ensemble fini, les $\alpha_i$ sont des ensembles d'étiquettes et les $H_i$ sont des ensembles de multiensembles dans $\mathbb{N}^{Q_{\text{det}}}$. Remarquons maintenant que chacune des intersections se réduit en un ensemble d'éléments d'arbres de la forme $\alpha[[\![H, \Theta]\!]]$, où $\alpha \subseteq \Sigma$ et $H \subseteq \mathbb{N}^{Q_{\text{det}}}$. En effet, d'une part, pour tous ensembles d'étiquettes $\alpha, \beta$ et tous ensembles d'arbres $S, T$, on a $\alpha[S] \cap \beta[T] = (\alpha \cap \beta)[S \cap T]$ et d'autre part, il n'est pas difficile de voir que pour tous ensembles de multiensembles $H, H' \subseteq \mathbb{N}^{Q_{\text{det}}}$, on a $[[\![H, \Theta]\!] \cap [[\![H, \Theta']\!] = [[\![H \cap H', \Theta]\!]$. [9] Soit donc $w \in Q_{\text{det}}$ et les ensembles d'étiquettes $\alpha_j$ et les ensembles de multiensembles $H_j$, pour $j \in J$ et $J$ un ensemble fini, tels que $\Theta_w = \bigcup_{j\in J} \alpha_j[[\![H_j, \Theta]\!]]$. Soit la relation de transition $\Delta_{\text{det}}$ sur $Q_{\text{det}} \times \wp(\Sigma) \times \wp(\mathbb{N}^{Q_{\text{det}}})$ qui contient exactement les tuples

$$(w, \alpha_j, H_j)$$

pour tout $w$ dans $Q_{\text{det}}$ et pour tout $j$ dans $J$ avec $J$, les $\alpha_j$ et les $H_j$ comme définis ci-dessus. Soit finalement $F_{\text{det}} = H_F$. D'après ce qui a été dit précédemment, il est facile de voir que l'automate $A_{\text{det}} = (\Sigma, Q_{\text{det}}, \Delta_{\text{det}}, F_{\text{det}})$ reconnaît le même ensemble d'arbres que l'automate $A$ et, par (2.1), c'est aussi un automate déterministe.

Nous avons donc montré que pour tout automate à contraintes numériques on peut définir un automate à contraintes numériques déterministe équivalent. Remarquons également que l'automate $A_{\text{det}}$ tel que nous l'avons défini, est également un automate complet ; ceci vient de fait que $w = \varnothing$ est état de $A_{\text{det}}$.

**Proposition 2.15** *Pour tout automate à contraintes numériques $A$, il existe un automate à contraintes numériques déterministe et complet $A_{det}$ tel que $\mathcal{L}(A) = \mathcal{L}(A')$.*

[9] Pour cette dernière propriété, voir également le lemme 6.10.

**Remarques**

Le nombre d'états de l'automate déterministe est exponentiel dans le nombre d'états de l'automate d'origine. De plus, de manière globale, le nombre de transitions pour un état de l'automate déterministe peut être exponentiel en le nombre d'états de l'automate d'origine (càd pour certains états $w$ de l'automate déterministe, le nombre de transitions de la forme $(w, \alpha, D) \in \Delta$ est en $\mathcal{O}(2^{|Q|})$, où $Q$ est l'ensemble d'états de l'automate d'origine).

Remarquons que si l'automate de départ $A$ est à exécution calculable, alors l'automate $A_{\text{det}}$ est également à exécution calculable. Ceci signifie qu'étant donné un multiensemble $\mathbf{h}$ sur $\mathbb{N}^{Q_{\text{det}}}$, on peut décider si $\mathbf{h}$ appartient à chacun des $H_D$ et, de ce fait, à l'ensemble $H$ apparaissant dans une transition de $A_{\text{det}}$. Notons que pour cela il n'est pas nécessaire de « calculer » les ensembles $H_D$, il suffit de pouvoir effectuer un test d'appartenance aux ensembles $D$ apparaissant dans une transition de l'automate d'origine. Techniquement cela correspond bien au fait de calculer tous les exécutions possibles.

### 2.3.2 Clôture par les opérations booléennes

Nous étudions la clôture des ACN par complémentaire, union et intersection.

**Clôture par complémentaire**

Les ACN sont clos par complémentaire : pour tout ACN $A$, il existe un ACN $A'$ dont le langage est le complémentaire de $\mathcal{L}(A)$ dans l'ensemble de tous les arbres $\mathcal{A}_\Sigma$.

Nous avons vu dans la section 2.3.1 que pour tout automate $A = (\Sigma, Q, \Delta, F)$, il existe un automate déterministe et complet équivalent $A_{\text{det}} = (\Sigma, Q_{\text{det}}, \Delta_{\text{det}}, F_{\text{det}})$. D'autre part, nous avons remarqué que l'application qui à chaque état $q$ dans $Q_{\text{det}}$ associe l'ensemble d'éléments d'arbres $\mathcal{L}_{A_{\text{det}}}(q)$ est une partition de l'ensemble des éléments d'arbres ; soit $\Theta$ cette application. Alors, avec les notations introduites pour la déterminisation d'un automate, $\mathcal{L}(A) = \mathcal{L}(A_{\text{det}}) = [\![F_{\text{det}}, \Theta]\!]$. Il n'est pas difficile de voir que $\mathcal{A}_\Sigma \setminus \mathcal{L}(A_{\text{det}}) = [\![\mathbb{N}^{Q_{\text{det}}} \setminus F_{\text{det}}, \Theta]\!]$. Nous en concluons que l'automate $A_{\text{compl}} = (\Sigma, Q_{\text{det}}, \Delta, \mathbb{N}^{Q_{\text{det}}} \setminus F_{\text{det}})$ est tel que $\mathcal{L}(A_{\text{compl}}) = \mathcal{A}_\Sigma \setminus \mathcal{L}(A)$ et donc les automates à contraintes numériques sont clos par complémentaire.

**Clôture par union**

Les ACN sont clos par union : pour tous automates $A, B$, il existe un automate $C$ dont le langage est égal à $\mathcal{L}(A) \cup \mathcal{L}(B)$.

Soient $A = (\Sigma, Q_A, \Delta_A, F_A)$ et $B = (\Sigma, Q_B, \Delta_B, F_B)$ deux automates. Sans perte de généralité, nous pouvons supposer que les ensembles d'états $Q_A$ et $Q_B$ sont disjoints. Nous commençons par définir les automates $A'$ et $B'$ qui sont équivalents à $A$ et $B$ respectivement et dont l'ensemble d'états est $Q_A \cup Q_B$. Pour tous ensembles finis non vides $Q, Q'$ et tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$, soit $\text{aug}(\mathbf{d}, Q, Q')$ le multiensemble dans $\mathbb{N}^{Q \cup Q'}$ qui coïncide avec

$\mathbf{d}$ sur l'ensemble $Q$ et associe 0 à tout $q$ dans $Q'$. Formellement, pour tout $q$ dans $Q \cup Q'$,

$$(\mathsf{aug}(\mathbf{d}, Q, Q'))(q) = \begin{cases} \mathbf{d}(q) & \text{si } q \in Q \\ 0 & \text{sinon.} \end{cases}$$

Nous étendons cette notation aux ensembles de multiensembles : pour tout ensemble de multiensembles $D$ inclus dans $\mathbb{N}^Q$, $\mathsf{aug}(D, Q, Q') \subseteq \mathbb{N}^{Q \cup Q'}$ est l'ensemble de multiensembles

$$\mathsf{aug}(D, Q, Q') = \{\mathsf{aug}(\mathbf{d}, Q, Q') \mid \mathbf{d} \in D\}.$$

La relation de transition de l'automate $A' = (\Sigma, Q_A \cup Q_B, \Delta'_A, F'_A)$ est définie par :

$$\Delta'_A = \{(q, \alpha, \mathsf{aug}(D, Q, Q')) \mid (q, \alpha, D) \in \Delta_A\}$$

et la condition d'acceptation $F'_A$ est l'ensemble $\mathsf{aug}(F_A, Q, Q')$.

L'automate $B' = (\Sigma, Q_A \cup Q_B, \Delta'_B, F'_B)$ est défini de façon similaire.

Il est facile de voir que $\mathcal{L}(A) = \mathcal{L}(A')$ et $\mathcal{L}(B) = \mathcal{L}(B')$. En effet, pour tout état $q$ dans $Q_B$, $\mathcal{L}_{A'}(q)$ est vide, de même $\mathcal{L}_{B'}(q')$ est vide pour tout $q'$ dans $Q_A$ et pour tout $D$ dans $\mathbb{N}^{Q_A \cup Q_B}$, $\mathcal{L}_{A'}(D)$ ne dépend pas des $\mathcal{L}_{A'}(q)$ pour $q$ appartenant à $Q_B$. Considérons maintenant l'automate $C = (\Sigma, Q_A \cup Q_B, \Delta'_A \cup \Delta'_B, F'_A \cup F'_B)$.

**Lemme 2.16** *Le langage de l'automate $C$ est l'union des langages des automates $A$ et $B$.*

*Preuve* D'après la définition de $C$, il n'est pas difficile de voir que pour tout arbre $t$, si $r$ est une exécution acceptante de $C$ sur $t$, alors $r$ est une exécution acceptante de $A$ sur $t$ ou $r$ est une exécution acceptante de $B$ sur $t$. L'inverse est également vrai : toute exécution acceptante de $A$ sur $t$ est également une exécution acceptante de $C$ sur $t$, et pareil pour l'automate $B$.

□

### Clôture par intersection

La clôture par intersection des automates à contraintes numériques suit directement de la clôture par union et de la clôture par complémentaire, utilisant que pour tous automates $A, B$, $\mathcal{L}(A) \cap \mathcal{L}(B) = \mathcal{A}_\Sigma \setminus ((\mathcal{A}_\Sigma \setminus \mathcal{L}(A)) \cup (\mathcal{A}_\Sigma \setminus \mathcal{L}(B)))$. Cependant, dans ce cas, le calcul de l'automate déterministe demande trois calculs d'un automate complémentaire, et donc trois calculs d'un automate déterministe, ce qui est beaucoup trop coûteux.

Un automate intersection peut être construit également en utilisant un automate produit. Soient $A = (\Sigma, Q, \Delta, F)$ et $A' = (\Sigma, Q', \Delta', F')$ deux automates. Considérons l'automate produit $A_\times = (\Sigma, Q_\times, \Delta_\times, F_\times)$ défini par : l'ensemble des états $Q_\times$ est le produit cartésien $Q \times Q'$, la relation de transition $\Delta_\times$ est

$$\Delta_\times = \{((q, q'), \alpha \cap \alpha', \mathsf{prod}(D, D', Q_\times)) \mid (q, \alpha, D) \in \Delta, (q', \alpha', D') \in \Delta'\}$$

où pour $D, D'$ étant des ensembles de multiensembles dans $\mathbb{N}^Q$ et $\mathbb{N}^{Q'}$ respectivement, l'ensemble $\mathsf{prod}(D, D', Q_\times)$ est le sous ensemble de $\mathbb{N}^{Q_\times}$ défini par :

$$\bigcup_{\mathbf{d} \in D, \mathbf{d}' \in D'} \mathsf{prod}(\mathbf{d}, \mathbf{d}', Q_\times)$$

et, finalement, pour tous multiensembles $\mathbf{d}, \mathbf{d}'$ dans $\mathbb{N}^Q$ et $\mathbb{N}^{Q'}$ respectivement, l'ensemble de multiensemble $\mathsf{prod}(\mathbf{d}, \mathbf{d}', Q_\times)$ est défini par :

$$\mathsf{prod}(\mathbf{d}, \mathbf{d}', Q_\times) = \left\{ h \in \mathbb{N}^{Q_\times} \;\middle|\; \begin{array}{l} \forall q \in Q \mid \sum_{q' \in Q} \mathbf{h}((q,q')) = \mathbf{d}(q) \\ \text{et} \\ \forall q' \in Q \mid \sum_{q \in Q} \mathbf{h}((q,q')) = \mathbf{d}(q') \end{array} \right\}$$

Finalement, la condition d'acceptation est $\mathsf{prod}(F, F', Q_\times)$.

Cette construction assure que pour tout état $(q, q')$ de l'automate produit $A_\times$, le langage de cet état dans $A_\times$ est l'intersection des langages des états $q$ dans $A$ et $q'$ dans $A'$, càd

$$\mathcal{L}_{A_\times}((q,q')) = \mathcal{L}_A(q) \cap \mathcal{L}_{A'}(q').$$

Notons que si les automates $A$ et $A'$ sont déterministes et complets, alors l'automate produit $A \times A'$ l'est aussi.

**Remarques**

Le calcul de l'automate complémentaire est aussi coûteux que le calcul de l'automate déterministe, et si un automate est à exécution calculable, c'est aussi le cas de son automate complémentaire. Pour l'union, l'automate union a une taille qui n'excède pas la somme des tailles des automates d'origine. Le caractère d'exécution calculable est préservé. Par contre, l'union de deux automates déterministes, telle qu'elle est décrite, ne résulte pas en un automate déterministe. Finalement, un automate intersection de deux automates peut être construit dont le nombre d'états n'excède pas le produit du nombre d'états des deux automates d'origine.

Les résultats de cette section nous mènent à la proposition ci-dessous :

**Proposition 2.17** *Les automates à contraintes numériques sont clos par union, intersection et complémentaire.*

### 2.3.3 Clôture par homomorphisme

Nous commençons par définir ce qu'est un homomorphisme d'étiquettes et ce que nous étendons par « clôture par homomorphisme (d'étiquettes) ». Ensuite nous montrons que les automates à contraintes numériques sont clos par les homomorphismes d'étiquettes.

**Définition 2.18 (Homomorphisme d'étiquettes)** Soient $\Sigma'$ un ensemble infini dénombrable d'étiquettes (pas forcément disjoint de $\Sigma$) et soit $h_\Sigma$ une application de $\Sigma$ dans $\Sigma'$. L'*homomorphisme d'étiquettes* défini par $h_\Sigma$ est l'application $h : \mathsf{EArbres}_\Sigma \cup \mathcal{A}_\Sigma \to \mathsf{EArbres}_{\Sigma'} \cup \mathcal{A}_{\Sigma'}$ définie par :

- Pour tout élément d'arbre $a[t]$ dans $\mathsf{EArbres}_\Sigma$, $h(a[t])$ est l'élément d'arbre $h_\Sigma(a)[h(t)]$ ;
- Pour tout arbre $\{\!|e_1, \ldots, e_n|\!\}$ dans $\mathcal{A}_\Sigma$, $h(\{\!|e_1, \ldots, e_n|\!\})$ est l'arbre $\{\!|h(e_1), \ldots, h(e_n)|\!\}$.

Intuitivement, l'homomorphisme d'étiquettes $h$ est un réétiquetage d'arbres par $h_\Sigma$.

Pour l'ensemble d'étiquettes $\alpha \subseteq \Sigma$, nous noterons par $h_\Sigma(\alpha)$ l'ensemble d'étiquettes de $\Sigma'$ qui est l'image de $\alpha$ par $h_\Sigma$. De même, si $S$ est un ensemble d'arbres de $\mathcal{A}_\Sigma$, $h(S)$ est l'ensemble d'arbres de $\mathcal{A}_{\Sigma'}$ image de $S$ par l'homomorphisme d'étiquettes $h$.

On dit qu'une classe d'automates est close par homomorphisme d'étiquette si pour tout automate $A$ de cette classe et pour tout homomorphisme d'étiquettes, il existe un automate de la même classe dont le langage est $h(\mathcal{L}(A))$.

Soit $h$ l'homomorphisme d'étiquettes défini par l'application $h_\Sigma : \Sigma \to \Sigma'$ et soit $A$ un automate à contraintes numériques. Nous allons définir l'automate $A' = (\Sigma', Q', \Delta', F')$ tel que $\mathcal{L}(A') = h(\mathcal{L}(A))$.

Tout d'abord, l'ensemble d'états de $A'$ est le même que l'ensemble d'états de $A$, càd

$$Q' = Q$$

Pour la relation de transition $\Delta'$, elle est définie par :

$$\Delta' = \{(q, h_\Sigma(\alpha), D) \mid (q, \alpha, D) \in \Delta\}.$$

Finalement, la condition d'acceptation $F'$ est égale à $F$.

Par sa définition même, $A'$ est un automate à contraintes numériques. Montrons que la construction de $A'$ est correcte.

**Lemme 2.19** $\mathcal{L}(A') = h(\mathcal{L}(A))$.

*Preuve* Nous allons montrer que pour tout état dans $Q$, $\mathcal{L}_{A'}(q) = h(\mathcal{L}_A(q))$ et pour tout ensemble de multiensembles $D$ dans $\mathbb{N}^Q$, $\mathcal{L}_{A'}(D) = h(\mathcal{L}_A(D))$. La preuve se fait par récurrence sur la taille des arbres. Nous définissons la hauteur d'un élément d'arbre par : $hauteur(e) = hauteur(\{\!|e|\!\})$. Pour tout entier naturel $n$, soit $\mathcal{A}_n$ l'ensemble des arbres de hauteur au plus $n$ et soit $\mathcal{EA}_n$ l'ensemble des éléments d'arbres de hauteur au plus $n$. Nous montrons que pour tout entier naturel $n$, pour tout état $q$ dans $Q_\mathcal{S}$ et pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^{Q_\mathcal{S}}$

(i) $h(\mathcal{L}_A(q)) \cap \mathcal{EA}_n = \mathcal{L}_{A'}(q) \cap \mathcal{EA}_n$ et

(ii) $h(\mathcal{L}_A(D)) \cap \mathcal{A}_n = \mathcal{L}_{A'}(q) \cap \mathcal{A}_n$.

La preuve est une récurrence simple sur $n$, utilisant uniquement les définitions de $\mathcal{L}_A(q)$ et $\mathcal{L}_A(D)$.

□

Nous en déduisons immédiatement la clôture des automates à contraintes numériques par les homomorphismes d'étiquettes.

**Lemme 2.20** *Les automates à contraintes numériques sont clos par homomorphisme d'étiquettes.*

### 2.3.4 Test du vide

Le test du vide d'un automate est décider si le langage de cet automate est égal à l'ensemble vide. Les deux questions qui se posent sont, d'abord, savoir si le test du vide est décidable et ensuite, si c'est le cas, connaître la complexité pour effectuer ce test.

Remarquons que le test du vide pour les ACN n'est pas décidable dans le cas général, ceci à cause de la grande généricité de la définition de ces automates, en particulier, dans une transition $(q, \alpha, D)$ d'un automate à contraintes numériques,

(i) l'ensemble $\alpha$ est quelconque, et à priori le vide n'est pas décidable pour (la classe de) cet ensemble ;

(ii) l'ensemble $D$ est quelconque, et à priori le vide n'est pas décidable pour (la classe de) cet ensemble.

Cette deuxième condition s'applique également à l'ensemble d'acceptation d'un automate ($F$). Nous illustrons ci-dessous deux exemples simples qui montrent que ces deux propriétés font obstacle à la décidabilité du vide d'un automate.

- Considérons l'automate $A = (\Sigma, \{q\}, \Delta, F)$ à un seul état, avec $\Delta$ contenant l'unique règle $(q, \alpha, \{0^Q\})$ et $F = \{1_q\}$. Intuitivement, cet automate reconnaît les arbres à une seule arête étiquetée par une étiquette dans $\alpha$. Le langage de cet automate est vide si et seulement si $\alpha$ est l'ensemble vide.
- Considérons l'automate $A = (\Sigma, \{q\}, \Delta, F)$ à un seul état, avec $\Delta$ contenant l'unique règle $(q, \Sigma, \{0^Q\})$ et $F$ est un sous ensemble de $\mathbb{N}$ défini comme le langage d'une machine de Turing. L'automate $A$ reconnaît les arbres plats (de hauteur un) ayant $k$ arêtes, pour $k$ appartenant à $F$. Le langage de cet automate est vide si et seulement si l'ensemble $F$ est vide.

Ces deux exemples peuvent paraître anecdotiques. Cependant, le deuxième cas est rencontré dans cette thèse, lorsqu'on s'intéresse à un automate à contraintes numériques dont le langage est un ensemble défini par une formule de la logique spatiale (logique définie dans le chapitre suivant) . En effet, dans le chapitre 5 qu'il existe une formule logique de la logique spatiale pour laquelle on peut définir un automate à contraintes numériques reconnaissant exactement l'ensemble d'arbres satisfaisant cette formule, et tel que la condition d'acceptation de cet automate est un ensemble récursivement énumérable quelconque.

Dans ce qui suit, nous donnons une condition suffisante pour que le vide d'un automate à contraintes numériques soit décidable. A part exclure les deux cas de figure (i) et (ii) ci-dessus, cette condition comprend une restriction supplémentaire.

Soit $A = (\Sigma, Q, \Delta, F)$ un automate à contraintes numériques. Pour tout état $q$ de l'automate, nous disons que $q$ est *accessible dans l'automate* $A$ si l'ensemble $\mathcal{L}_A(q)$ n'est pas vide. De la même façon, pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^Q$, nous disons que $D$ est *accessible dans l'automate* $A$ si l'ensemble $\mathcal{L}_A(D)$ n'est pas vide. D'après ces définitions, le langage de l'automate $A$ est non vide si et seulement si la condition d'acceptation $F$ est accessible dans $A$.

Rappelons que, par définition, pour tout ensemble de multiensemble $D$, le langage associé à $D$ est $\mathcal{L}_A(D) = \bigcup_{\mathbf{d} \in D} \mathbf{d}(q_1) \cdot \mathcal{L}_A(q_1) \mid \cdots \mid \mathbf{d}(q_k) \cdot \mathcal{L}_A(q_k)$. Rappelons également que pour tout entier naturel $n$ et pour tout ensemble d'éléments d'arbres $E$, l'ensemble d'arbres $n \cdot E$ est vide seulement si $n > 0$ et $E = \varnothing$. Alors l'ensemble $D$ est accessible dans $A$ si et seulement s'il existe un multiensemble $\mathbf{d}$ dans $D$ tel que pour tout $q$ dans $Q$, l'ensemble $\mathbf{d}(q) \cdot \mathcal{L}_A(q)$ est non vide, et donc, si et seulement si

($\star$) il existe $\mathbf{d}$ dans $D$ tel que pour tout $q$ dans $Q$, $\mathbf{d}(q) > 0$ implique que $\mathcal{L}_A(q)$ est non vide (càd l'état $q$ est accessible dans $A$).

Considérons maintenant la fonction booléenne

$$\mathsf{accessible}(A = (\Sigma, Q, \Delta, F), D, Q')$$

qui pour l'automate $A = (\Sigma, Q, \Delta, F)$, l'ensemble de mulitiensembles $D \subseteq \mathbb{N}^Q$ et l'ensemble d'états $Q' \subseteq Q$ retourne vrai si et seulement si il existe $\mathbf{d}$ dans $D$ tel que pour tout $q$ dans $Q$, si $\mathbf{d}(q) > 0$, alors $q$ appartient à $Q'$. Notons que la décidabilité du test du vide dépendra de la possibilité de donner un algorithme pour cette fonction. D'après ce qui a été dit juste avant, si $Q'$ est un ensemble d'états accessibles dans $A$ et que $\mathsf{accessible}(A = (\Sigma, Q, \Delta, F), D, Q')$ retourne vrai, alors nous pouvons conclure que $D$ est accessible dans $A$. Alors, si nous connaissons l'ensemble $Q' \subseteq Q$ des états accessibles dans $A$, $\mathsf{accessible}(A = (\Sigma, Q, \Delta, F), F, Q')$ retourne vrai si et seulement si le langage de l'automate $A$ est non vide.

Intéressons nous maintenant au calcul de l'ensemble des états accessibles d'un automate. Rappelons que, par définition, le langage associé à l'état $q$ est $\mathcal{L}_A(q) = \bigcup_{(q,\alpha,D)\in\Delta} \alpha[\mathcal{L}_A(D)]$. Donc, l'état $q$ est accessible si et seulement si ils existent un ensemble d'étiquettes $\alpha$ non vide et un ensemble de multiensembles $D$ tels que $(q, \alpha, D)$ est une transition de l'automate et l'ensemble $D$ est accessible dans $A$. Considérons maintenant la fonction $\mathsf{états\_acc\_aux}(A = (\Sigma, Q, \Delta, F), Q')$ suivante :

$\mathsf{états\_acc\_aux}(A = (\Sigma, Q, \Delta, F), Q') =$
  $Q''$ : variable acceptant des sous-ensembles de $Q'$ ;
  $Q'' = Q'$ ;
  pour tout $(q, \alpha, D)$ dans $\Delta$ faire
    si $\alpha \neq \varnothing$ et $\mathsf{accessible}(A, D, Q')$
    alors ajouter $q$ à $Q''$ ;
    fin si ;
  fin faire ;
  retourner $Q''$ ;

D'après ce qui a été dit précédemment, il est facile de voir que pour tout automate $A = (\Sigma, Q, \Delta, F)$ et tout ensemble d'états $Q' \subseteq Q$, l'ensemble $Q''$ retourné par $\mathsf{états\_acc\_aux}(A = (\Sigma, Q, \Delta, F), Q')$ est exactement l'ensemble des états de l'automate qui sont accessibles sachant que les états dans $Q'$ sont accessibles. Alors, en itérant cette fonction commençant par $Q'$ étant l'ensemble vide et jusque saturation, nous allons obtenir l'ensemble des états accessibles de l'automate $A$. Ceci se fait par la fonction $\mathsf{états\_acc}(A = (\Sigma, Q, \Delta, F))$ suivante :

$\mathsf{états\_acc}(A = (\Sigma, Q, \Delta, F)) =$
  $Q', Q_\mathrm{a}$ : variables acceptant des sous-ensembles de $Q$ ;
  $Q_\mathrm{a} = \varnothing$ ;
  faire
    $Q' = \mathsf{états\_acc\_aux}(Q_\mathrm{a})$ ;
  tant que $Q' \neq Q_\mathrm{a}$ ;
  retourner $Q_\mathrm{a}$ ;

Remarquons que commencer par $Q_\mathrm{a}$ étant égal à l'ensemble vide n'est pas gênant, puisque $\mathsf{états\_acc\_aux}(A = (\Sigma, Q, \Delta, F), \varnothing)$ n'est pas forcément un ensemble vide ; plus précisément, $\mathsf{états\_acc\_aux}(A = (\Sigma, Q, \Delta, F), \varnothing)$ est l'ensemble des états $q$ tels qu'il existe une transition $(q, \alpha, D)$ dans $\Delta$ avec $\alpha$ étant non vide et $D$ contenant le multiensemble $0^Q$.

Nous pouvons maintenant définir la fonction booléenne vide($A$) qui retourne vrai si et seulement si le langage de l'automate $A$ est vide

$$\begin{array}{l}\mathsf{vide}(A = (\Sigma, Q, \Delta, F)) = \\ \quad \text{retourner non accessible}(A, F, \text{états\_acc}(A))\,;\end{array}$$

Notons maintenant que la définition (effective) de ces quatre fonctions est possible à deux conditions, qui sont énoncées dans le lemme suivant et définissent des conditions suffisantes pour que le vide d'un automate à contraintes numériques quelconque puisse être calculé.

**Lemme 2.21** *Soit $A = (\Sigma, Q, \Delta, F)$ un automate à contraintes numériques. Soit $\mathcal{D}$ l'ensemble des contraintes numériques apparaissant dans la définition de l'automate, càd*

$$\mathcal{D} = \{F\} \cup \{D \subseteq \mathbb{N}^Q \mid \textit{il existe } \alpha \subseteq \Sigma,\ q \in Q \textit{ tels que } (q, \alpha, D) \in \Delta\}.$$

*Si pour tout ensemble d'étiquettes $\alpha$ apparaissant dans une transition de l'automate, on peut décider si $\alpha$ est vide et si pour tout sous-ensemble d'états de l'automate $Q' \subseteq Q$ et pour tout ensemble de multiensembles $D$ dans $\mathcal{D}$ on peut décider la propriété suivante*

*($\star$) il existe $\mathbf{d} \in D$ tel que pour tout état $q' \in Q$, $\mathbf{d}(q') > 0$ implique $q' \in Q'$*

*alors on peut décider si $\mathcal{L}(A)$ est vide.*

**Remarques**

Nous pouvons donner uniquement une estimation du coût de la procédure de test du vide d'un automate $A = (\Sigma, Q, \Delta, F)$, qui est paramétrée par le coût pour tester la propriété ($\star$). La complexité du test du vide, tel qu'il a été défini, est en $\mathcal{O}(|Q| \times |\Delta| \times f(A))$, $|\Delta|$ désigne le nombre de transitions de l'automate et $f(A)$ est la complexité (au pire) pour tester la propriété ($\star$) pour $Q' \subseteq Q$ et $D$ apparaissant dans une transition de l'automate $A$.

### 2.3.5 Test d'appartenance

Rappelons que le test d'appartenance est, étant donné un automate $A$ et un arbre $t$, décider si $t$ est reconnu par $A$. Nous avons donné précédemment une méthode possible pour effectuer le test d'appartenance, à savoir construire une exécution ascendante de l'automate $A$ sur l'arbre $t$ ; ceci est toujours possible lorsque l'automate $A$ est à exécution calculable (par définition). Un automate a exécution calculable est un automate $A = (\Sigma, Q, \Delta, F)$ satisfaisant ces trois conditions :

- l'ensemble des triplets $(q, \alpha, D)$ appartenant à $\Delta$ est fini et on peut effectivement énumérer ces triplets,
- pour toute étiquette $a$ et tout ensemble d'étiquettes $\alpha$ tel que $(q, \alpha, D)$ est une transition de l'automate, on peut décider si $a$ appartient à $\alpha$ et
- pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$ et pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^Q$ dans
$$\{F\} \cup \{D \mid (q, \alpha, D) \text{ dans } \Delta\}$$
on peut décider si $\mathbf{d}$ appartient à $D$.

Nous avons également présenté brièvement comment on calcule de manière ascendante une exécution acceptable pour l'automate $A$ sur l'arbre $t$, lorsque $A$ est à exécution calculable. Ici nous donnons juste une estimation du coût du calcul d'une exécution pour un automate déterministe et complet. Soit l'automate à contraintes numériques déterministe et complet $A = (\Sigma, Q, \Delta, F)$, et supposons que nous avons une représentation à exécution calculable de $A$. Étant donné un arbre $t$, voici une fonction permettant de calculer une exécution de $A$ sur $t$ :

```
        calculer_execution(A = (Σ, Q, Δ, F), t) =
1.        r : exécution en train d'être calculé, initialisé par r(e) = indefini pour
                                  toute arête e de t ;
2.        pour toute arête e de l'arbre, dans un ordre ascendant, faire
3.          soit {e_1, ..., e_k} l'ensemble des arêtes successeurs de e ;
4.          soit d le multiensemble {r(e_1), ..., r(e_k)}
5.          pour tout (q, α, D) appartenant à Δ faire
6.            si etq(e) ∈ α et d ∈ D alors
7.              r(e) := q ;
8.            fin si ;
9.          fin pour ;
10.       fin pour ;
```

Pour une représentation adéquate de l'arbre $t$, les lignes 3. et 4. de cette fonction peuvent s'exécuter en temps constant. Pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$, notons par $|\mathbf{d}|$ le nombre d'éléments de $\mathbf{d}$ (chaque élément étant compté autant de fois que sa multiplicité). Il est facile de voir que le multiensemble $\mathbf{d}$ construit à la ligne 5. de cet algorithme est tel que $|\mathbf{d}| \leq |t|$.

La complexité du calcul d'une exécution est en $\mathcal{O}(|t| \times |\Delta| \times f(A,t))$, où $|\Delta|$ désigne le nombre de transitions de l'automate $A$ et $f(A,t)$ est la fonction qui majore la complexité du test $\mathbf{d} \in D$ pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$ tel que $|\mathbf{d}| \leq |t|$ et pour tout ensemble de multiensembles $D$ apparaissant dans une transition l'automate $A$ ou en tant que condition d'acceptation de celui-ci.

## 2.4 Propriétés des automates à contraintes numériques sans étoile et semilinéaires

Nous reprenons ici les résultats sur les ACN en général présentés dans la section précédente et essayons de voir comment ils s'appliquent aux ACN sans étoile et aux ACN semilinéaires. Dans cette étude, nous prendrons également en considération si les automates considérés sont effectifs ou non. Dans les cas où il s'agit de définir un automate résultat (par exemple pour la déterminisation, pour la clôture par les opérations booléennes) nous essayerons que cet automate résultat soit effectif si les automates donnés au départ le sont.

### 2.4.1 Déterminisation

Nous montrons que les ACN sans étoile et semilinéaires sont effectivement déterminisables : càd si $A$ est un ACN sans étoile (respectivement semilinéaire), alors l'automate déterministe $A_{\text{det}}$

équivalent à $A$ défini dans la section 2.3.1 est sans étoile (respectivement semilinéaire). De plus, si l'automate $A$ est effectif, on peut donner une représentation effective de $A_{\text{det}}$.

Nous commençons par le cas des ACN sans étoile.

### ACN sans étoile

Soit $A$ un ACN sans étoile, nous devons montrer que l'automate $A_{\text{det}}$ comme défini précédemment est un ACN sans étoile. D'après la construction de $A_{\text{det}}$ (voir la section 2.3.1) et en utilisant le fait que les ensembles de multiensembles sans étoile sons clos par union et complémentaire, il suffit de montrer que pour tout ensemble $D \subseteq \mathbb{N}^Q$ sans étoile, l'ensemble $H_D \subseteq \mathbb{N}^{Q_{\text{det}}}$ est un ensemble sans étoile. Rappelons que, par définition, l'ensemble $H$ est sans étoile si c'est une union finie d'ensembles linéaires dont les périodes sont des multiensembles unitaires (càd de la forme $1_q$). Comme les ensembles sans étoile sont clos par union, il suffit de montrer que si $D$ est un ensemble linéaire de périodes unitaires, alors $H_D$ est un ensemble sans étoile. Nous commençons par un résultat intermédiaire.

**Lemme 2.22** *Pour tous multiensembles* $\mathbf{d}_1, \mathbf{d}_2$ *dans* $\mathbb{N}^Q$, *on a*

$$H_{\mathbf{d}_1+\mathbf{d}_2} = H_{\mathbf{d}_1} + H_{\mathbf{d}_2}.$$

*Preuve* Supposons d'abord que $\mathbf{h} \in H_{\mathbf{d}_1+\mathbf{d}_2}$. Donc, par définition,

$$\begin{array}{l} \forall w \in Q_{\text{det}}. \forall q \in w.\ \exists n_{w,q} \in \mathbb{N} \text{ tels que :} \\ \quad - \forall q \in Q.\ \left(\sum_{w \in Q_{\text{det}} \mid q \in w} n_{w,q}\right) = (\mathbf{d}_1 + \mathbf{d}_2)(q) \text{ et} \\ \quad - \forall w \in Q_{\text{det}}.\ \left(\sum_{q \in w} n_{w,q}\right) = \mathbf{h}(w) \end{array}$$

et $h(\varnothing) = 0$. Soient maintenant, pour tout $w$ dans $Q_{\text{det}}$ et pour tout $q$ dans $w$, les nombres $n^{(1)}_{w,q}$ et $n^{(2)}_{w,q}$ tels que $n^{(1)}_{w,q} + n^{(2)}_{w,q} = n_{w,q}$ et tels que pour $i$ dans $1, 2$ et pour tout $q$ dans $Q$, on a $\sum_{w \in Q_{\text{det}} \mid q \in w} n^{(i)}_{w,q} = \mathbf{d}_i(q)$. Notons que les nombres $n^{(i)}_{w,q}$ existent toujours. Soit maintenant $\mathbf{h}_1, \mathbf{h}_2 \in \mathbb{N}^{Q_{\text{det}}}$ les multiensembles tels que pour tout $w$ dans $Q_{\text{det}}$, $\mathbf{h}_i(w) = \sum_{q \in w} n^{(i)}_{w,q}$, ceci pour $i$ étant 1 ou 2. Alors il est facile de voir que, d'une part, $\mathbf{h}_1$ appartient à $H_{\mathbf{d}_1}$ et $\mathbf{h}_2$ appartient à $H_{\mathbf{d}_2}$ et, d'autre part, $\mathbf{h} = \mathbf{h}_1 + \mathbf{h}_2$. Donc, $\mathbf{h}$ appartient à $H_{\mathbf{d}_1} + H_{\mathbf{d}_2}$.

Supposons maintenant que $\mathbf{h} \in H_{\mathbf{d}_1} + H_{\mathbf{d}_2}$, càd il existe $\mathbf{h}_1 \in H_{\mathbf{d}_1}$ et $\mathbf{h}_2 \in H_{\mathbf{d}_2}$ tels que $\mathbf{h} = \mathbf{h}_1 + \mathbf{h}_2$. Alors, par définition, pour $i$ étant 1 ou 2, on a :

$$\begin{array}{l} \forall w \in Q_{\text{det}}. \forall q \in w.\ \exists n^{(i)}_{w,q} \in \mathbb{N} \text{ tels que :} \\ \quad - \forall q \in Q.\ \left(\sum_{w \in Q_{\text{det}} \mid q \in w} n^{(i)}_{w,q}\right) = \mathbf{d}_i(q) \text{ et} \\ \quad - \forall w \in Q_{\text{det}}.\ \left(\sum_{q \in w} n^{(i)}_{w,q}\right) = \mathbf{h}_i(w) \end{array}$$

et $\mathbf{h}_i(\varnothing) = 0$. Considérons maintenant, pour tout $w$ dans $Q_{\text{det}}$ et pour tout $q$ dans $w$, le nombre $n_{w,q} = n^{(1)}_{w,q} + n^{(2)}_{w,q}$. Alors, pour tout $q$ dans $Q$, $\sum_{w \in Q_{\text{det}} \mid q \in w} n_{w,q} = \mathbf{d}_1(q) + \mathbf{d}_2(q)$ et pour tout $u$ dans $Q_{\text{det}}$, $\sum_{q \in w} n_{w,q} = \mathbf{h}_1(w) + \mathbf{h}_2(w)$. Alors, en utilisant que $\mathbf{d}_1(q) + \mathbf{d}_2(q) = (\mathbf{d}_1 + \mathbf{d}_2)(q)$ et que $\mathbf{h}_1(w) + \mathbf{h}_2(w) = \mathbf{h}(w)$, et par définition de $H_{\mathbf{d}_1+\mathbf{d}_2}$, il est facile de voir que $h$ appartient à $H_{\mathbf{d}_1+\mathbf{d}_2}$.

□

Par associativité et commutativité de la somme de multiensemble et d'ensembles de multiensembles, l'égalité établie dans le précédent lemme peut naturellement être étendue pour les sommes d'un nombre fini d'éléments.

**Lemme 2.23** *Pour tout état $q$ dans $Q$,*

$$H_{1_q} = \bigcup_{w \in Q_{det} | q \in w} \{1_w\}.$$

*Preuve* Conséquence facile de la définition : soit $q$ fixé. On peut voir que pour tout $q'$ différent de $q$, tous les nombres $n_{w,q'}$ sont nuls et que $n_{w,q}$ est égal à un pour un unique $w$ tel que $q \in w$ ; soit donc $w$ tel que $n_{w,q} = 1$. Alors, $\mathbf{h}(w) = 1$, et $\mathbf{h}(w') = 0$ pour tout $w'$ différent de $w$.

□

**Lemme 2.24** *Soit $D$ l'ensemble linéaire* $\mathsf{Lin}(\mathbf{b}, 1_{q_1}, \ldots, 1_{q_k})$*, où $q_1, \ldots, q_k$ sont des états dans $Q$. Soit $B$ l'ensemble*

$$\bigcup_{\mathbf{h} \in H_\mathbf{b}} \mathsf{Lin}(\mathbf{h}, \mathbf{h}_1^{(1)}, \ldots, \mathbf{h}_1^{(m_1)}, \ldots, \mathbf{h}_k^{(1)}, \ldots, \mathbf{h}_k^{(m_k)})$$

*où pour tout $i$ dans $1..k$, $\{\mathbf{h}_i^{(1)}, \ldots, \mathbf{h}_i^{(m_i)}\}$ est l'ensemble $H_{1_{q_i}}$. Alors $H_D = B$.*

*Preuve* Par définition, $\mathbf{h}'$ appartient à $B$ si et seulement si il existe $\mathbf{h}$ dans $H_\mathbf{b}$ et pour tout $i$ dans $1..k$ ils existent les entiers naturels $\lambda_i^{(1)}, \ldots, \lambda_i^{(m_i)}$ tels que

$$\mathbf{h}' = \mathbf{h} + \lambda_1^{(1)}\mathbf{h}_1^{(1)} + \cdots + \lambda_1^{(m_1)}\mathbf{h}_1^{(m_1)} + \cdots + \lambda_k^{(1)}\mathbf{h}_k^{(1)} + \cdots + \lambda_k^{(m_k)}\mathbf{h}_k^{(m_k)}.$$

C'est le cas si et seulement si $\mathbf{h}$ appartient à l'ensemble

$$H_\mathbf{b} + \lambda_1 H_{1_{q_1}} + \cdots + \lambda_k H_{1_{q_k}},$$

où pour tout $i$ dans $1..k$, $\lambda_i = \sum_{j \in 1..m_i} \lambda_i^{(j)}$. Or, par le lemme 2.22, ce dernier ensemble est égal à $H_\mathbf{d}$, où $\mathbf{d}$ est le multiensemble $\mathbf{b} + \lambda_1 1_{q_1} + \cdots + \lambda_k 1_{q_k}$, càd $\mathbf{h}'$ appartient à $H_\mathbf{d}$ pour un certain $\mathbf{d}$ dans $\mathsf{Lin}(\mathbf{b}, 1_{q_1}, \ldots, 1_{q_k})$.

□

Maintenant, d'après le lemme 2.23, les multiensembles $\mathbf{h}_1^{(1)}, \ldots, \mathbf{h}_1^{(m_1)}, \ldots, \mathbf{h}_k^{(1)}, \ldots, \mathbf{h}_k^{(m_k)}$ ci-dessus sont des multiensembles unitaires, et donc $H_D$ est un ensemble sans étoile. Remarquons également que si l'ensemble sans étoile $D$ est effectif, nous avons une représentation effective de $H_D$.

**Proposition 2.25** *Pour tout ACN sans étoile effectif $A$, il existe un ACN sans étoile effectif déterministe et complet $A_{det}$ tel que $\mathcal{L}(A) = \mathcal{L}(A_{det})$.*

Pour finir, notons que les remarques sur la taille de l'automate déterministe faites dans le cas général sont valables également ici. Le nombre d'états de l'automate déterministe est exponentiel dans le nombre d'états de l'automate d'origine. Le nombre de transitions pour chaque état peut également être en $\mathcal{O}(2^{|Q|})$, où $Q$ est l'ensemble d'états de l'automate d'origine.

### ACN semilinéaires

Le fait que les ACN semilinéaires sont déterminisables est connu, à travers leur équivalence avec les automates de Presburger [Seidl et al., 2003]. En effet, la notion d'automate déterministe pour les automates de Presburger correspond à la notion d'automate déterministe et complet pour les ACN semilinéaires. Nous avons donc le résultat suivant de déterminisation des ACN semilinéaires.

**Lemme 2.26** *Pour tout ACN semilinéaire $A$, un ACN déterministe $A'$ peut être construit tel que $\mathcal{L}(A) = \mathcal{L}(A')$.*

*Preuve* D'après la proposition 2.4, soit $A'' = (\Sigma, Q, \Delta, p)$ un automate de Presburger déterministe tel que $\mathcal{L}(A'') = \mathcal{L}(A)$. Par définition, nous savons que $\Delta$ est une fonction de $Q \times \Sigma$ dans $PresbForm(\{z_q \mid q \in Q\})$, l'ensemble des formules de l'arithmétique de Presburger construites sur l'ensemble de variables $\{z_q \mid q \in Q\}$. Considérons l'ACN semilinéaire $A' = (\Sigma, Q, \Delta', F)$ où la relation de transition $\Delta'$ est

$$\Delta' = \{(q, \{a\}, solutions(p)) \mid q, a \text{ tels que } \Delta(q, a) = p'\},$$

où pour toute formule de l'arithmétique de Presburger $p'$ dans $PresbForm(\{z_q \mid q \in Q\})$, $solitions(p')$ est l'ensemble de multiensembles de $\mathbb{N}^Q$ défini par

$$solutions(p') = \{(q_1 \to \zeta(z_{q_1}), \ldots, q_k \to \zeta(z_{q_k}) \mid \zeta \models p')\},$$

où nous avons supposé $Q = \{q_1, \ldots, q_k\}$. La condition d'acceptation $F$ est $solutions(p)$. Nous savons que l'ensemble de multiensembles semilinéaire $solutions(p')$ peut être construit pour toute formule de l'arithmétique de Presburger $p'$.

□

Ce lemme nous donne le résultat théorique montrant que la classe des ACN semilinéaires est déterminisable. Il serait cependant intéressant de savoir si et comment un ACN semilinéaire déterministe effectif peut être construit directement à partir d'un ACN semilinéaire effectif. Montrons que l'automate $A_{\text{det}}$ tel qu'il a été défini dans la section 2.3.1 peut être effectivement construit. Comme pour le cas des ACN sans étoile, il suffirait de pouvoir calculer (et représenter effectivement) l'ensemble $H_D$ pour $D$ étant un ensemble semilinéaire, et donc, l'ensemble $H_D$ pour $D$ étant un ensemble linéaire. La construction et la preuve sont similaires au cas sans étoile : pour l'ensemble linéaire $D = \mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$, l'ensemble $H_D$ est l'ensemble semilinéaire

$$\mathbf{h}' = \mathbf{h} + \lambda_1^{(1)}\mathbf{h}_1^{(1)} + \cdots + \lambda_1^{(m_1)}\mathbf{h}_1^{(m_1)} + \cdots + \lambda_k^{(1)}\mathbf{h}_k^{(1)} + \cdots + \lambda_k^{(m_k)}\mathbf{h}_k^{(m_k)}.$$

où pour tout $i$ dans $1..k$, $\{\mathbf{h}_i^{(1)}, \ldots, \mathbf{h}_i^{(m_i)}\}$ est l'ensemble $H_{\mathbf{p}_i}$.

En ce qui concerne la taille de l'automate $A_{\text{det}}$, le nombre d'états de $A_{\text{det}}$ est exponentiel dans le nombre d'états de l'automate $A$. Le nombre de transitions pour chaque état peut aussi être exponentiel dans le nombre d'états de l'automate d'origine.

Nous terminons cette section en remarquant que l'automate déterministe défini de cette façon a une taille beaucoup trop grande pour être exploitable.

### 2.4.2 Clôture par les opérations booléennes

Nous savons par [Seidl et al., 2003] que les ACN semilinéaires sont clos par union et intersection. D'après les constructions ci-dessus, il est facile de voir que les ACN sans étoile sont effectivement clos par union et complémentaire. La clôture par complémentaire vient du fait que les ensembles sans étoile sont clos par complémentaire, et la clôture par union suit du fait facilement vérifiable que si $D$ est une ensemble sans étoile, alors l'ensemble $\{\mathtt{aug}(\mathbf{d}), Q, Q' \mid \mathbf{d} \in D\}$ est également sans étoile (voir la section 2.3.2 pour la définition de cet ensemble).

**Proposition 2.27** – *[Seidl et al., 2003] Les automates à contraintes numériques semilinéaires sont clos effectivement par les opérations booléennes.*
– *Les automates à contraintes numériques sans étoile sont effectivement clos par les opérations booléennes.*

### 2.4.3 Clôture par homomorphisme

Dans la section 2.3.3 nous avons montré que les automates à contraintes numériques sont clos par homomorphisme d'étiquettes. Ce n'est pas le cas des automates à contraintes numériques sans étoile ou semilinéaires. Nous donnons en exemple d'ACN sans étoile $A$ et un homomorphisme $h$ tel que l'image du langage de $A$ par $h$ n'est pas définissable par un ACN sans étoile. Ensuite nous montrons que les ACN sans étoile et semilinéaires sont clos par les homomorphismes d'étiquettes surjectifs.

Soit $\Sigma$ l'ensemble d'étiquettes $\{a_i \mid i \in \mathbb{N}\}$ et $\Sigma'$ l'ensemble $\{b_i \mid i \in \mathbb{N}\}$, et considérons l'application $h_\Sigma$ de $\Sigma$ dans $\Sigma'$ définie par :

$$\text{pour tout } k \text{ dans } \mathbb{N}, \quad h_\Sigma(a_{2k}) = b_0 \text{ et } h_\Sigma(a_{2k+1}) = b_{2k+1}.$$

Soit l'automate (sans étoile) $A = (\Sigma, Q, \Delta, F)$ avec $Q = \{q\}$, $\Delta = \{(q, \Sigma \setminus \{a_2, a_3, a_4\}, \{0^Q\})\}$ et $F = 1_q$. Intuitivement, cet automate reconnaît les arbres composés d'une unique arête dont l'étiquette est différente de $a_2, a_3, a_4$. Il n'est pas difficile de voir que $h(\mathcal{L}(A))$ est l'ensemble d'arbres composés d'une unique arête dont l'étiquette est une parmi $b_0, b_1, b_5, b_7, \ldots, b_{2k+1}, \ldots$. Cet ensemble ne peut pas être reconnu par un ACN sans étoile. En effet, rappelons que dans un automate sans étoile, tout ensemble d'étiquettes apparaissant dans une transition de l'automate doit être fini ou cofini, ce qui n'est pas le cas de l'ensemble $\{b_0, b_1, b_5, b_7, \ldots, b_{2k+1}, \ldots\}$.

Dans la suite nous présentons plus en détail la raison pour laquelle la méthode utilisée dans la section 2.3.3 pour montrer que les ACN sont clos par homomorphisme d'étiquettes ne marche pas dans le cas particulier des ACN sans étoile. Ceci nous permettra également de montrer la clôture de ces automates par les homomorphismes surjectifs.

Rappelons d'abord comment nous avons établi la clôture dans le cas des ACN en général (section 2.3.3). Soit $h_\Sigma : \Sigma \to \Sigma'$ une application et $h$ est l'homomorphisme défini par $h_\Sigma$, soit $A = (\Sigma, Q, \Delta, F)$ un automate à contraintes numériques. Dans la section 2.3.3 nous avons montré que l'automate $A' = (\Sigma', Q, \Delta', F)$, avec $\Delta' = \{(q, h_\Sigma(\alpha), D) \mid (q, \alpha, D) \in \Delta\}$ satisfait $\mathcal{L}(A') = h(\mathcal{L}(A))$.

Si l'automate $A$ ci-dessus est sans étoile, ce n'est pas forcément le cas de l'automate $A'$, ceci à cause des ensemble $h_\Sigma(\alpha)$ apparaissant dans les transitions de $A'$ qui ne sont pas forcément finis ou cofinis, comme nous avons vu un peu plus haut. Par contre, remarquons que les

contraintes numériques apparaissant dans $A'$ sont les mêmes que dans $A$. Donc, si toutes les contraintes numériques dans $A$ sont des ensembles sans étoile (respectivement semilinéaires), c'est également le cas de l'automate $A'$. Ceci nous mène à remarquer que si l'application $h_\Sigma$ est telle que l'image par $h_\Sigma$ de tout ensemble fini ou cofini est un ensemble fini ou cofini, alors l'automate $A'$ est sans étoile (respectivement semilinéaire) si $A$ est sans étoile (respectivement semilinéaire). Cette condition est vérifiée par une application $h_\Sigma$ surjective.

**Lemme 2.28** *Si $h_\Sigma : \Sigma \to \Sigma$ est une application surjective, alors pour tout $\alpha \subseteq \Sigma$ fini ou cofini, $h_\Sigma(\alpha)$ est fini ou cofini.*

*Preuve* Soit $\alpha$ un sous ensemble fini de $\Sigma$. Par définition même, $h_\Sigma(\alpha)$ est un sous ensemble fini de $\Sigma'$. Montrons que $h_\Sigma(\overline{\alpha})$ est un ensemble cofini de $\Sigma'$. Par surjectivité de $h_\Sigma$, il est facile de voir que

$$h_\Sigma(\overline{\alpha}) = \Sigma' \setminus (h_\Sigma(\alpha) \setminus h_\Sigma(\overline{\alpha}))$$

L'ensemble $h_\Sigma(\alpha) \setminus h_\Sigma(\overline{\alpha})$ étant fini, nous en déduisons que $h_\Sigma(\overline{\alpha})$ est un ensemble cofini.

□

Maintenant, un homomorphisme d'étiquettes $h$ est surjectif si et seulement si l'application $h_\Sigma$ le définissant est surjective. Nous en déduisons que

**Lemme 2.29** *Les ACN sans étoile et semilinéaires sont clos par les homomorphismes d'étiquettes surjectifs.*

### 2.4.4 Test du vide

Nous montrons que le test du vide est décidable pour les ACN sans étoile et les ACN semilinéaires. Il suffit de montrer que la condition donnée dans le lemme 2.21 est vérifiée pour ces deux fragments. Ceci n'est pas très difficile à voir ; considérons juste le cas des ACN semilinéaires, le cas des ACN sans étoile étant un cas particulier.

Soit l'ACN semilinéaire $A = (\Sigma, Q, \Delta, F)$. Étant donné $Q' \subseteq Q$ sous ensemble des états de $A$ et $D$ ensemble de contraintes numériques dans $\mathbb{N}^Q$, nous devons décider si

($\star$) $D$ contient un élément $\mathbf{d}$ tel que les composantes non nulles de $\mathbf{d}$ correspondent à des états dans $Q'$.

Rappelons que $D$ est un ensemble semilinéaire, donc une union finie d'ensembles linéaires. Alors il suffit de pouvoir décider la propriété pour chacun des ensembles linéaires composant $D$. Soit donc $L = \mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ un ensemble linéaire. Nous devons pouvoir décider si $L$ contient un élément $\mathbf{d}$ satisfaisant ($\star$). Il est facile de voir que c'est le cas seulement si $\mathbf{b}$ satisfait cette propriété, càd pour tout $q$ dans $Q$, si $\mathbf{b}(q)$ est strictement positif, alors $q$ appartint à $Q'$.

**Lemme 2.30** *Le vide est décidable pour les automates à contraintes numériques semilinéaires et les automates à contraintes numériques sans étoile.*

En ce qui concerne la complexité du test du vide, rappelons que dans le cas des ACN en général elle est en $\mathcal{O}(|Q| \times |\Delta| \times f(A))$, où $|\Delta|$ est le nombre de transitions de l'automate et $f(A)$ est la complexité au pire des pour tester la propriété ($\star$) pour $Q' \subset Q$ et $D$ un ensemble

apparaissant dans une transition de l'automate. Dans le cas particulier des ACN sans étoile, $(\star)$ se résume à tester s'il existe un ensemble linéaire $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ composant $D$ tel que toute composante non nulle de $\mathbf{b}$ correspond à un état dans $Q'$. Ceci se fait en $|Q| \times K(A)$, où $K(A)$ est le plus grand nombre d'ensembles linéaires composant un ensemble semilinéaire $D$ apparaissant dans une règle de l'automate $A$.

Des résultats établis dans la section 2.3.4 et de ce qui précède, nous déduisons

**Lemme 2.31** *Pour un ACN sans étoile ou semilinéaire effectif $A = (\Sigma, Q, \Delta, F)$, tester si $\mathcal{L}(A)$ est vide peut se faire en un temps $\mathcal{O}(|\Delta| \times |Q|^2 \times K(A))$.*

### 2.4.5 Test d'appartenance

Nous montrons que le test d'appartenance est décidable pour les ACN sans étoile et les ACN semilinéaires, càd pour tout ACN sans étoile ou semilinéaire $A$ et pour tout arbre $t$, on peut décider si $t$ appartient au langage de $A$.

Commençons par les ACN sans étoile, pour lesquels le test d'appartenance est très facile. Soit $A = (\Sigma, Q, \Delta, F)$ un ACN sans étoile. Rappelons qu'il s'agit de tester si un multiensemble $\mathbf{d} \in \mathbb{N}^Q$ appartient à $D$, où $D$ est un ensemble de multiensembles apparaissant dans une transition de l'automate $A$, et donc $D$ est un ensemble sans étoile. Par définition, $D$ est représenté comme une union d'ensembles linéaires de la forme $\mathsf{Lin}(\mathbf{b}, 1_{q_1}, \ldots, 1_{q_k})$, où $\{q_1, \ldots, q_k\} \subseteq Q$. Donc, il suffit de pouvoir tester si $\mathbf{d}$ appartient à l'un de ces ensembles linéaires. Ceci est très simple : $\mathbf{d}$ appartient à $\mathsf{Lin}(\mathbf{b}, 1_{q_1}, \ldots 1_{q_k})$ si et seulement si pour tout $q$ dans $Q$, $\mathbf{d}(q) \geq \mathbf{b}(q)$ et pour tout $q$ dans $Q$, si $\mathbf{d}(q) > \mathbf{b}(q)$, alors il existe $j$ parmi $1..k$ tel que $q_j = q$.

En ce qui concerne la complexité, le test si $\mathbf{d}$ appartient à $\mathsf{Lin}(\mathbf{b}, 1_{q_1}, \ldots, 1_{q_k})$ se fait en $|Q|^2$ (en remarquant que $k \leq |Q|$). Rappelons que (voir la section 2.3.5), dans le cas des automates à contraintes numériques en général, la complexité d'un calcul d'exécution de l'automate est en $\mathcal{O}(|t| \times |\Delta| \times f(A,t))$, où $f(A,t)$ est une fonction qui majore la complexité du test $\mathbf{d} \in D$ pour tout multiensemble $\mathbf{d}$ tel que $|\mathbf{d}| \leq |t|$ et pour tout ensemble $D$ apparaissant dans une transition de l'automate $A$. Nous en déduisons facilement que pour tout arbre $t$, tester si $t$ appartient à $\mathcal{L}(A)$ peut se faire en $\mathcal{O}(|t| \times |\Delta| \times |Q|^2 \times K(A))$, où $K(A)$ est le plus grand nombre d'ensembles linéaires définissant un des ensembles de multiensembles $D$ apparaissant dans une transition l'automate $A$.

Regardons maintenant le cas d'un automate $A = (\Sigma, Q, \Delta, F)$ à contraintes semilinéaires, et plus précisément comment en peut tester si un multiensemble $\mathbf{d}$ appartient à l'ensemble semilinéaire $D$ apparaissant dans une transition de l'automate $A$. Encore une fois, il suffit de pouvoir tester si $\mathbf{d}$ appartient à l'ensemble linéaire $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$. Ceci peut être effectué en temps exponentiel.

Une autre façon à envisager pour le test d'appartenance est de construire, pour tout ensemble de multiensembles $D$ apparaissant dans une transition de l'automate, un automate (de mots) reconnaissant les représentations binaires des multiensembles $\mathbf{d}$ apparaissant à $D$ (voir par exemple [Boudet and Comon, 1996], [Wolper and Boigelot, 2000]). Cet automate est construit à partir de la formule de Presburger dont les solutions sont l'ensemble $D$. La construction de cette formule est linéaire dans $K$, où $K$ est la somme de toutes les multiplicités de toutes les

bases et périodes définissant l'ensemble $D$. La taille de l'automate reconnaissant $D$ est au pire des cas triplement exponentielle dans la taille de la formule de Presburger [Klaedtke, 2004].

# Chapitre 3

# Logiques

Nous présentons d'abord la logique monadique du second ordre (MSO) et son extension, la logique MSO de Presburger. Ensuite nous définissons la logique spatiale LS dont l'étude est l'objet central de cette thèse. Nous présentons différents fragments et variantes de la logique spatiale, obtenus par des restrictions syntaxiques de celle-ci.

## 3.1 Logique monadique du second ordre et extension

### 3.1.1 Logique monadique du second ordre (MSO)

Les logiques monadiques du second ordre (MSO) sont obtenues comme la restriction des logiques du second ordre aux prédicats du second ordre monadique ou bien comme l'extension des logiques du premier ordre (FO) par des variables ensemblistes.

Le grand succès des logiques MSO est dû à une suite de résultats reliant ces logiques à des automates sur différentes structures, ce qui traduit également certaines bonnes propriétés de ces logiques.

- Un langage de mots finis est reconnaissable par un automate fini si et seulement s'il est MSO définissable [Büchi, 1960], [Elgot, 1961].
- Un langage de termes sur une signature finie dont tous les symboles sont d'arité finie[1] est reconnaissable par un automate d'arbres fini si et seulement s'il est MSO définissable [Doner, 1970], [Thatcher and Wright, 1968].
- Un langage d'arbres ordonnés d'arité non bornée est reconnaissable par un automate à haie si et seulement s'il est MSO définissable. Ce résultat, appartenant au « folklore » est explicitement annoncé dans [Neven and Schwentick, 2002].

Lorsque nous parlons de « logiques MSO », nous entendons une famille de logiques qui diffèrent uniquement par leurs prédicats de base, qui sont généralement combinés entre eux par des opérations booléennes et des quantifications sur des variables du premier ou second ordre. Par exemple, MSO sur les mots est construite sur les prédicats de base $P_a(u)$ vrai si la $u^{\text{ème}}$ lettre du mot est un $a$ et $u \leq u'$ vraie si la position $u$ précède la position $u'$. Pour les arbres

[1] Ces termes sont le plus souvent appelés arbres ; à la différence des arbres que nous considérons, ce sont des arbres ordonnés et d'arité bornée.

ordonnés d'arité non bornée, une logique MSO utilisée très souvent a pour prédicats de base premier$(u, u')$ vrai si le nœud $u'$ est le premier fils du nœud $u$ et suivant$(u, u')$ vrai si le nœud $u$ et $u'$ ont le même nœud père et $u'$ suit immédiatement $u$. Dans la suite, nous présentons une logique MSO adaptée aux arbres non ordonnés d'arité non bornée qui nous intéressent dans ce travail.

### Syntaxe

Nous considérons $\mathcal{U}_1$ et $\mathcal{U}_2$ deux familles dénombrables de variables. Les éléments de $\mathcal{U}_1$ sont des variables du premier ordre, nous utilisons la lettre $u$ avec éventuellement un indice pour désigner ces variables. Les éléments de $\mathcal{U}_2$ sont des variables du second ordre, notées $U$. Les formules de la logique MSO sont définies par la syntaxe :

$$\psi ::= \mathsf{etq}_a(u) \ \text{(pour } a \in \Sigma) \ \mid \ u < u' \ \mid \ u \in U \ \mid \ \psi \vee \psi \ \mid \ \neg\psi \ \mid \ \exists u.\psi \ \mid \ \exists U.\psi$$

Une occurrence de variable dans une formule MSO est dite *libre* si elle n'apparaît sous aucun opérateur de quantification existentielle $\exists$. Une formule est dite *close* si elle n'admet pas de variables libres.

### Sémantique

Les formules de MSO sont interprétées sur des arbres vus comme des structures logiques. Rappelons que $\sigma$ est la signature $\{<\} \cup \{\mathsf{etq}_b \mid b \in \Sigma\}$ où $<$ est un prédicat binaire (en notation infixée) et les $\mathsf{etq}_b$ sont des prédicat unaires, et que tout arbre $t$ peut être représenté comme une $\sigma$-structure $S^t$ finie. Soit $S$ une $\sigma$-structure finie de domaine $E$ et $\gamma$ une valuation des variables : $\gamma$ associe des éléments de $E$ aux variables de premier ordre et des sous-ensembles de $E$ aux variables de second ordre. Pour la valuation $\gamma$, la variable $u$ dans $\mathcal{U}_1$ et l'élément $e$ de $E$, on note $\gamma[u \mapsto e]$ la valuation qui associe $e$ à la variable $u$ et est identique à $\gamma$ pour toutes les variables différentes de $u$. La valuation $\gamma[U \mapsto E']$, pour $U \in \mathcal{U}_2$ et $E' \subseteq E$ est définie de façon similaire. On dit que la structure $S$ *satisfait* la formule $\psi$ pour la valuation $\gamma$, noté $S, \gamma \models \psi$, si :

- $\psi$ est $\mathsf{etq}_a(u)$ et $\mathsf{etq}_a(\gamma(u))$ est vrai dans $S$ ;
- $\psi$ est $u < u'$ et $\gamma(u) < \gamma(u')$ est vrai dans $S$ ;
- $\psi$ est $u \in U$ et $\gamma(u)$ appartient à $\gamma(U)$ ;
- $\psi$ est $\psi' \vee \psi''$ et $S, \gamma \models \psi'$ ou $S, \gamma \models \psi''$ ;
- $\psi$ est $\neg\psi'$ et $S, \gamma \models \psi'$ n'est pas vérifié ;
- $\psi$ est $\exists u.\psi'$ et $S, \gamma[u \mapsto a] \models \psi'$ est vérifié pour un certain $a$ élément de $E$ ;
- $\psi$ est $\exists U.\psi'$ et $S, \gamma[U \mapsto E'] \models \psi'$ est vérifié pour un certain $E'$ sous-ensemble de $E$.

Pour une formule $\psi$ close, nous écrirons parfois $S \models \psi$ à la place de $S, \gamma \models \psi$. Pour une formule de MSO close $\psi$, le *langage* associé à $\psi$, noté $[\![\psi]\!]$, est l'ensemble d'arbres $t$ tels que $S^t \models \psi$. Pour un ensemble d'arbres $T$, nous disons que $T$ *est MSO-définissable* s'il existe une formule close $\psi$ de la logique MSO telle que $T = [\![\psi]\!]$.

Si $\psi$ et $\psi'$ sont des formules de MSO, alors les formules dérivées $\psi \wedge \psi'$, $\forall u.\psi$, $\forall U.\psi$, $\psi \rightarrow \psi'$, $\psi \leftrightarrow \psi'$ sont définies en utilisant la négation de la façon habituelle.Nous allons librement utiliser ces formules dérivées.

**Exemple 3.1** Nous définissons la formule MSO close $\psi$ qui est satisfaite par les arbres $t$ tels que toute arête de $t$ étiquetées par $a$ a au plus deux arêtes successeurs et toute arête de $t$ étiquetée par $b$ a au plus trois successeurs. Définissons d'abord quelques formules auxiliaires : $u = u'$ permettant de tester l'égalité de deux arêtes, $succ(u, U)$ vérifiée si $U$ est l'ensemble des arêtes successeurs de $u$ et $card_2(U)$ et $card_3(U)$ vraies si le cardinal de l'ensemble $U$ est inférieur à deux, respectivement à trois.

$$
\begin{aligned}
u = u' &::= \forall U.u \in U \leftrightarrow u' \in U \\
succ(u, U) &::= \forall u'.u' \in U \leftrightarrow u < u' \\
card_2(U) &::= \forall u_1, u_2, u_3.(u_1 \in U \wedge u_2 \in U \wedge u_3 \in U) \rightarrow (u_1 = u_2 \vee u_2 = u_3 \vee u_3 = u_1) \\
card_3(U) &::= \forall u_1, u_2, u_3, u_4.(u_1 \in U \wedge u_2 \in U \wedge u_3 \in U \wedge u_4 \in U) \rightarrow \\
&\qquad (u_1 = u_2 \vee u_2 = u_3 \vee u_3 = u_4 \vee u_1 = u_3 \vee u_1 = u_4 \vee u_2 = u_3)
\end{aligned}
$$

Alors $\psi$ est la formule

$$\forall U, \forall u.succ(u, U) \rightarrow ((\mathsf{etq}_a(u) \rightarrow card_2(U)) \wedge (\mathsf{etq}_b(u) \rightarrow card_3(U)))$$

### 3.1.2 Logique MSO de Presburger (PMSO)

La logique PMSO [Seidl et al., 2003] est introduite comme extension de MSO dans le cadre des langages de documents semi-structurés. Dans sa définition initiale, la logique est interprétée sur des arbres étiquetés sur les nœuds par des étiquettes d'un ensemble fini de symboles. Nous l'adaptons ici pour des arbres étiquetés sur les arêtes par un ensemble de symboles dénombrable. PMSO permet de définir des conditions arithmétiques sur les cardinaux de sous-ensembles de successeurs d'un nœud de l'arbre. Ces conditions arithmétiques sont définies au moyen de formules de l'arithmétique de Presburger.

#### Syntaxe

La logique PMSO est définie comme extension de MSO avec la formule de base $u/p$, où $p$ est une formule de l'arithmétique de Presburger construite sur l'ensemble de variables numériques $\{z_U \mid U \in \mathcal{U}_2\} \cup \mathcal{Z}$ (on suppose qu'aucun des $z_U$ n'appartient à $\mathcal{Z}$). Les formules de PMSO sont donc définies par :

$$\psi ::= \mathsf{etq}_a(u) \text{ (pour } a \in \Sigma) \mid u < u' \mid u/p \mid u \in U \mid \psi \vee \psi \mid \exists u.\psi \mid \exists U.\psi$$

#### Sémantique

Pour la $\sigma$-structure $S$ et la valuation des variables de premier et second ordre $\gamma$, $S, \gamma \models u/p$ est vérifié si $\zeta \models p$, où $\zeta$ est la valuation des variables dans $\mathcal{Z}$ qui à la variable $z_U$ de $\mathcal{Z}$ associe le nombre d'éléments de $\gamma(U)$ qui sont aussi des arêtes successeurs de $\gamma(u)$. Formellement, pour tout $z_U$ dans $\mathcal{Z}$, $\zeta(z_U) = Card(\gamma(U) \cap E')$, où $E'$ est l'ensemble des arêtes successeurs de $\gamma(u)$. L'interprétation des autres types de formules est inchangée par rapport à MSO.

De la même façon que pour la logique MSO, si $\psi$ est une formule close, alors nous pourrons écrire $S \models \psi$ à la place de $S, \gamma \models \psi$. Pour une formule close $\psi$, $[\![\psi]\!]$ désigne l'ensemble d'arbres $t$ tels que $S^t \models \psi$. Pour un ensemble d'arbres $T$, nous disons que $T$ *est PMSO-définissable* s'il existe une formule close $\psi$ de la logique PMSO telle que $T = [\![\psi]\!]$.

**Exemple 3.2** Nous construisons la formule PMSO close $\psi$ reconnaissant les arbres $t$ tels que chaque arête de l'arbre a autant d'arêtes successeurs étiquetées par $a$ que d'arêtes successeurs étiquetées par $b$. Pour toute étiquette $a$, la formule auxiliaire $étiquette_a(U)$ est vraie si $U$ est l'ensemble des arêtes de l'arbre étiquetées par $a$.

$$étiquette_a(U) ::= \forall u.u \in U \leftrightarrow \mathsf{etq}_a(u)$$
$$\psi ::= \forall U_a, \forall U_b.(étiquette_a(U_a) \wedge étiquette_b(U_b)) \rightarrow \forall u.u/z_{U_a} = z_{U_b}$$

### 3.1.3 Logiques monadiques du second ordre et automates à contraintes numériques

Nous présentons ici des équivalences qui existent entre les logiques MSO et PMSO et les automates à contraintes numériques.

Dans [Boneva and Talbot, 2005] nous avons montré que la logique MSO est équivalente aux automates à contraintes numériques sans étoile.[2]

**Théorème 3.1** *Un ensemble d'arbres est MSO-définissable si et seulement s'il est le langage d'un automate à contraintes numériques sans étoile.*

*Preuve* Pour montrer que tout langage d'arbres reconnaissable par un ACN sans étoile est MSO-définissable, on définit une formule MSO qui vérifie l'existence d'une exécution acceptante de l'automate. Inversement, pour montrer que tout ensemble d'arbres MSO-définissable est reconnu par ACN sans étoile, l'idée est de construire un automate pour chaque formule MSO atomique, et d'utiliser la clôture des ACN sans étoile par les opérations booléennes et la projection. Cette idée (classique dans les preuves d'équivalence de MSO avec une classe d'automates) ne marche pas directement pour les ACN sans étoile à cause des ensembles co-finis d'étiquettes qui peuvent apparaître dans les transitions ; nous utilisons une extension des ACN comme représentation intermédiaire. Une preuve détaillée est présentée dans la section A.1 l'annexe A.

□

Si on considère que l'ensemble d'étiquettes $\Sigma$ est fini, alors la logique PMSO est équivalente aux automates de Presburger ; ce résultat est établi dans [Seidl et al., 2003]. Nous montrons que $\Sigma$ est un ensemble dénombrable, alors la logique PMSO est équivalente aux automates à contraintes numériques semilinéaires.

**Théorème 3.2** *Un ensemble d'arbres est PMSO-définissable si et seulement s'il est le langage d'un automate à contraintes numériques semilinéaires (voir aussi [Seidl et al., 2003], Théorème 5).*

[2] Dans [Boneva and Talbot, 2005], ce résultat est établi pour une définition différente des automates à contraintes numériques.

*Preuve* La preuve est similaire à celle du théorème 3.1.

□

## 3.2 Logique spatiale (LS)

La logique spatiale que nous présentons ici est une extension d'un fragment de la logique des ambients [Cardelli and Gordon, 2000a]. Il s'agit du fragment dit spatial, qui est interprété sur des arbres et non sur des processus du calcul des ambients comme c'est le cas de la logique dans [Cardelli and Gordon, 2000a]. Cette logique a été utilisée comme la base du langage de requêtes pour données semi-structurées TQL [Cardelli and Ghelli, 2001].

### 3.2.1 Syntaxe

Soit $\mathcal{X}_e$ un ensemble dénombrable de variables d'étiquettes notées $x, y, \ldots$, $\mathcal{X}_a$ un ensemble dénombrable de variables d'arbres notées $X, Y, \ldots$ et $\mathcal{X}_r$ un ensemble dénombrable de variables de récursion notées $\xi, \ldots$. Les formules de LS sont définies par la syntaxe :

| | | |
|---|---|---|
| $\phi ::=$ | $\mathbf{0}$ | arbre vide |
| | $\eta[\phi]$ | imbrication ou emboîtement |
| | $\phi \mid \phi$ | composition |
| | $\top$ | vérité |
| | $\neg\phi$ | négation |
| | $\phi \vee \phi$ | disjonction |
| | $X$ | variable d'arbre |
| | $\exists X.\phi$ | quantification existentielle sur les arbres |
| | $\exists x.\phi$ | quantification existentielle sur les étiquettes |
| | $\xi$ | variable de récursion |
| | $\mu\xi.\phi$ | plus petit point fixe |
| | $\eta = \eta$ | test d'égalité d'étiquettes |
| $\eta ::=$ | $a$ | étiquette constante |
| | $x$ | variable d'étiquettes |

où $a$ est une étiquette dans $\Sigma$. Pour garantir l'existence du plus petit point fixe (formule $\mu\xi.\phi$), nous imposons la restriction syntaxique que dans la formule $\mu\xi.\phi$, toutes les occurrences libres[3] de la variable $\xi$ dans $\phi$ apparaissent sous un nombre pair de négations dans $\phi$.

Une occurrence de la variable d'étiquette $x$ (respectivement occurrence de la variable d'arbre $X$) est dite *libre* dans la formule $\phi$ si elle n'apparaît pas sous une quantification $\exists x$ (respectivement $\exists X$) dans la formule $\phi$. De la même façon, une occurrence de variable de récursion $\xi$ est dite *libre* dans $\phi$ si elle n'apparaît sous un opérateur de point fixe $\mu\xi$. Une formule est dite *close* si elle n'admet pas d'occurrences libres de variables.

On note par $varrec(\phi)$ l'ensemble des variables de récursion apparaissant dans la formule $\phi$.

[3] Voir définition de variable libre ci-dessous.

Tout au long de cette thèse et à chaque fois qu'il s'agit d'une formule $\phi$ de la logique LS, nous supposons que deux occurrences distinctes des opérateurs $\exists$ et $\mu$ dans $\phi$ lient des variables différentes et qu'aucune variable n'apparaît à la fois libre et non libre dans une formule. Cette propriété peut toujours être assurée par un renommage des variables non libres.

### 3.2.2 Sémantique

L'interprétation des formules de la logique spatiale est donnée en termes d'ensembles d'arbres. Soit $\rho$ une valuation des variables d'étiquettes et des variables d'arbres, et soit $\delta$ une valuation des variables de récursion ; c'est à dire, $\rho$ une application de $\mathcal{X}_e \cup \mathcal{X}_a$ dans $\Sigma \cup \mathsf{Arbres}_\Sigma$ qui respecte les types des variables (càd associe des étiquettes aux variables d'étiquettes et des arbres aux variables d'arbres) et $\delta$ est une application de $\mathcal{X}_r$ dans $\wp(\mathsf{Arbres}_\Sigma)$. Pour deux valuations $\rho$ et $\delta$, on note $dom(\rho)$ et $dom(\delta)$ leurs domaines respectifs.

L'interprétation de la formule $\phi$ sous les valuations $\rho$ et $\delta$, notée $[\![\phi]\!]_{\rho,\delta}$, est un ensemble d'arbres défini récursivement sur la structure de $\phi$ comme suit (nous utilisons ici la représentation des arbres comme des multiensembles imbriqués) :

$$
\begin{array}{lcl}
[\![\mathbf{0}]\!]_{\rho,\delta} & = & \{\{\!|\}\!\} \\
[\![\eta[\phi]]\!]_{\rho,\delta} & = & \{\rho(\eta)\}[[\![\phi]\!]_{\rho,\delta}] \\
[\![\phi \,|\, \phi']\!]_{\rho,\delta} & = & [\![\phi]\!]_{\rho,\delta} \,|\, [\![\phi']\!]_{\rho,\delta} \\
[\![\top]\!]_{\rho,\delta} & = & \mathsf{Arbres}_\Sigma \\
[\![\neg\phi]\!]_{\rho,\delta} & = & \mathsf{Arbres}_\Sigma \setminus [\![\phi]\!]_{\rho,\delta} \\
[\![\phi \vee \phi']\!]_{\rho,\delta} & = & [\![\phi]\!]_{\rho,\delta} \cup [\![\phi']\!]_{\rho,\delta} \\
[\![X]\!]_{\rho,\delta} & = & \{\rho(X)\} \\
[\![\exists X.\phi]\!]_{\rho,\delta} & = & \bigcup_{X \in \mathsf{Arbres}_\Sigma} [\![\phi]\!]_{\rho[X \mapsto t],\delta} \\
[\![\exists x.\phi]\!]_{\rho,\delta} & = & \bigcup_{a \in \Sigma} [\![\phi]\!]_{\rho[x \mapsto a],\delta} \\
[\![\xi]\!]_{\rho,\delta} & = & \delta(\xi) \\
[\![\mu\xi.\phi]\!]_{\rho,\delta} & = & \bigcap \{S \subseteq \mathsf{Arbres}_\Sigma \mid S \supseteq [\![\phi]\!]_{\rho,\delta[\xi \mapsto S]}\} \\
[\![\eta = \eta']\!]_{\rho,\delta} & = & \begin{cases} \mathsf{Arbres}_\Sigma \text{ si } \rho(\eta) = \rho(\eta') \\ \varnothing \text{ sinon} \end{cases}
\end{array}
$$

Pour l'arbre $t$, la formule de la logique spatiale $\phi$ et les valuations $\rho, \delta$, on dit que $t$ *satisfait* $\phi$ pour les valuations $\rho$ et $\delta$, écrit $t, \rho, \delta \models \phi$, si $t$ appartient à $[\![\phi]\!]_{\rho,\delta}$. Lorsque $\phi$ est une formule close, nous écrirons parfois $[\![\phi]\!]$ et $t \models \phi$ à la place de $[\![\phi]\!]_{\rho,\delta}$ et $t, \rho, \delta \models \phi$ respectivement.

### 3.2.3 Opérateurs dérivés

Plusieurs opérateurs dérivés ont été introduits pour la logique des ambients et la logique TQL. Nous donnons ici la définition de certains de ces opérateurs.

| | | |
|---|---|---|
| $\bot$ | $= \neg\top$ | faux |
| $\phi \wedge \phi$ | $= \neg(\neg\phi \vee \neg\phi)$ | conjonction |
| $\overline{\mathbf{0}}$ | $= \neg\mathbf{0}$ | arbre non vide |
| $\phi \,\|\, \phi'$ | $= \neg(\neg\phi \mid \neg\phi')$ | composition duale |
| $\phi \mid\!\Rrightarrow \phi'$ | $= \neg(\phi \mid \neg\phi')$ | |
| $\phi \rightarrow \phi'$ | $= \neg\phi \vee \phi'$ | implication |
| $\forall X.\phi$ | $= \neg(\exists X.\neg\phi)$ | quantification universelle sur les arbres |
| $\forall x.\phi$ | $= \neg(\exists x.\neg\phi)$ | quantification universelle sur les étiquettes |
| $\nu\xi.\phi$ | $= \neg\mu\xi.\neg\phi\langle\xi \rightarrow \neg\xi\rangle$ | plus grand point fixe |
| $\phi*$ | $= \mu\xi.\phi \mid \xi \vee \mathbf{0}$ | étoile |

où $\phi\langle\xi \rightarrow \neg\xi\rangle$ est la formule obtenue en remplaçant dans $\phi$ toutes les occurrence libres de la variable $\xi$ par la formule $\neg\xi$.

### 3.2.4 Propriétés de la satisfiabilité

L'interprétation des formules logiques peut également s'exprimer en termes de la relation de satisfaction. Les propriétés énoncées ci-dessous pourraient être prises comme définition de la satisfaction, mais l'interprétation sous forme d'ensembles que nous avons donnée est plus adaptée pour les formules à point fixe.

**Lemme 3.3** *Pour tout arbre $t$, toutes formules $\phi, \phi'$ et toutes valuations $\rho$ et $\delta$, les équivalences suivantes sont vérifiées :*

$$
\begin{array}{lcl}
t,\rho,\delta \models \mathbf{0} & \Leftrightarrow & t = \{\!|\}\\
t,\rho,\delta \models \eta[\phi] & \Leftrightarrow & \exists t' \in \mathsf{Arbres}_\Sigma.\, t = \rho(\eta)[t'] \wedge t', \rho, \delta \models \phi\\
t,\rho,\delta \models \phi \mid \phi' & \Leftrightarrow & \exists t', t'' \in \mathsf{Arbres}_\Sigma.\, t = t' \uplus t'' \wedge t, \rho, \delta \models \phi \wedge t', \rho, \delta' \models \phi'\\
t,\rho,\delta \models \top & & \\
t,\rho,\delta \models \neg\phi & \Leftrightarrow & t,\rho,\delta \not\models \phi\\
t,\rho,\delta \models \phi \vee \phi' & \Leftrightarrow & t,\rho,\delta \models \phi \vee t,\rho,\delta \models \phi'\\
t,\rho,\delta \models X & \Leftrightarrow & t = \rho(X)\\
t,\rho,\delta \models \exists X.\phi & \Leftrightarrow & \exists s \in \mathsf{Arbres}_\Sigma.\, t, \rho[X \rightarrow s], \delta \models \phi\\
t,\rho,\delta \models \exists x.\phi & \Leftrightarrow & \exists a \in \Sigma.\, t, \rho[x \rightarrow a], \delta \models \phi\\
t,\rho,\delta \models \xi & \Leftrightarrow & t \in \delta(\xi)\\
t,\rho,\delta \models \mu\xi.\phi & \Leftrightarrow & t,\rho,\delta \models \phi\langle\xi \rightarrow \mu\xi.\phi\rangle\\
t,\rho,\delta \models \overline{\mathbf{0}} & \Leftrightarrow & t \neq \{\!|\}\\
t,\rho,\delta \models \phi \,\|\, \phi' & \Leftrightarrow & \forall t', t''.\, t = t' \uplus t'' \Rightarrow (t', \rho, \delta \models \phi \vee t'', \rho, \delta \models \phi')\\
t,\rho,\delta \models \phi \mid\!\Rrightarrow \phi' & \Leftrightarrow & \forall t', t''.\, (t = t' \uplus t'' \wedge t', \rho, \delta \models \phi) \Rightarrow t'', \rho, \delta \models \phi'\\
t,\rho,\delta \not\models_{\rho,\delta} \bot & & \\
t,\rho,\delta \models \phi \wedge \phi' & \Leftrightarrow & t,\rho,\delta \models \phi \wedge t,\rho,\delta \models \phi'\\
t,\rho,\delta \models \forall X.\phi & \Leftrightarrow & \forall s \in \mathsf{Arbres}_\Sigma.\, t, \rho[X \rightarrow s], \delta \models \phi\\
t,\rho,\delta \models \forall x.\phi & \Leftrightarrow & \forall a \in \Sigma.\, t, \rho[x \rightarrow a], \delta \models \phi\\
t,\rho,\delta \models \phi* & \Leftrightarrow & \exists n \in \mathbb{N}.t \in n \cdot [\![\phi]\!]
\end{array}
$$

*Preuve* Presque toutes ces propriétés ont été montrées dans [Cardelli and Ghelli, 2004], sauf celles concernant les opérateurs dérivés $\overline{0}$ et $\phi \models\!\Rightarrow \phi'$. Pour $\overline{0}$, la propriété suit directement de la définition de cet opérateur, et pour $\phi \models\!\Rightarrow \phi'$ se déduit de la définition de l'opérateur et des propriétés pour $\phi \mid \phi'$ et $\neg\phi$.

□

Dans la suite nous utilisons le lemme 3.3 sans y faire référence.

### 3.2.5 Fragments et variantes de la logique

Dans cette thèse nous allons étudier différents fragment et variantes de la logique spatiale.

#### Le fragment minimal mini-LS

Nous appelons ce fragment « minimal » puisqu'il se réduit uniquement aux opérateurs spatiaux de la logique et les connecteurs booléens : en quelque sorte c'est le plus petit fragment de LS contenant les opérateurs spatiaux. mini-LS est le fragment de LS suivant :

$$\phi ::= \mathbf{0} \mid a[\phi] \mid \phi \mid \phi \mid \top \mid \neg\phi \mid \phi \vee \phi.$$

où $a$ est une étiquette dans $\Sigma$.

L'expressivité de la logique mini-LS est assez limitée. Intuitivement, elle permet de décrire des éléments de la structure d'une partie de l'arbre proche de la racine et de taille limitée par la taille de la formule, comme le montre l'exemple 3.3.

**Exemple 3.3** Propriétés exprimables par la logique mini-LS :

(i) Un chemin $a.b.c$ dans l'arbre, partant de la racine :

$$a[b[c[\top] \mid \top] \mid \top] \mid \top$$

(ii) La racine possède une arête successeur étiquetée par $a$ qui possède au moins deux arêtes successeurs étiquetées par $b$

$$a[b[\top] \mid b[\top] \mid \top] \mid \top$$

(iii) Au moins deux arêtes successeurs de la racine sont étiquetées par $a$ et au plus une arête successeur de la racine est étiquetée par $b$ :

$$(a[\top] \mid a[\top] \mid \top) \wedge (\neg(b[\top] \mid b[\top] \mid \top))$$

Cette propriété peut s'écrire également

$$a[\top] \mid a[\top] \mid \neg(b[\top] \mid b[\top] \mid \top)$$

Le principal intérêt d'étudier cette logique est de comprendre l'expressivité des opérateurs spatiaux.

Notons finalement que ce fragment peut être facilement encodé dans MSO. Comme nous le verrons plus tard, un fragment qui inclut mini-LS est équivalent à MSO, mais le passage de l'un

vers l'autre nécessite d'utiliser des automates d'arbres. Nous donnons ici un encodage direct de mini-LS dans MSO.

Pour toute formule $\phi$ dans mini-LS, nous construisons la formule close $\Psi_\phi(U)$ de MSO telle que pour tout arbre $t$ dans $\mathcal{A}_\Sigma$, $t \models \phi$ si et seulement si $S^t \models \forall U.((\forall u.u \in U) \rightarrow \Psi_\phi)$, où $S^t$ est la $\sigma$-structure correspondant à $t$. Le rôle de la variable $U$ est d'assurer que le domaine d'interprétation de la formule MSO est bien le domaine de $S^t$.

$$
\begin{aligned}
\Psi_\top(U) &::= \mathit{vrai} \\
\Psi_{\neg\phi}(U) &::= \neg\Psi_\phi(U) \\
\Psi_{\phi\vee\phi'}(U) &::= \Psi_\phi(U) \vee \Psi_{\phi'}(U) \\
\Psi_{\mathbf{0}}(U) &::= U = \varnothing \\
\Psi_{a[\phi]}(U) &::= \forall U_2, U_1.(\mathit{src}(U_1, U_2) \wedge U_1 = U \setminus U_2) \rightarrow (\mathit{singl\_etiq}_a(U_2) \wedge \Psi_\phi(U_1)) \\
\Psi_{\phi\,|\,\phi'}(U) &::= \exists U', U''.\mathit{dunion}(U, U', U'') \wedge \Psi_\phi(U') \wedge \Psi_{\phi'}(U'') \wedge \\
&\qquad \forall U_1, U_2, U_3.(\mathit{src}(U, U_1) \wedge \mathit{src}(U', U_2) \wedge \mathit{src}(U'', U_3)) \rightarrow \mathit{dunion}(U_1, U_2, U_3)
\end{aligned}
$$

La définition des formules auxiliaires est donnée ci-dessous. Intuitivement, $\mathit{src}(U, U')$ est vraie si $U'$ est l'ensemble des arêtes successeurs de la racine dans l'arbre dont le domaine est $U$ ; $\mathit{dunion}(U, U', U'')$ est vraie si l'ensemble $U$ est l'union disjointe des ensembles $U'$ et $U''$ ; pour toute étiquette $a$, $\mathit{sing\_etiq}_a(U)$ est vraie si l'ensemble $U$ est un ensemble contenant une seule arête et cette arête est étiquetée par $a$.

$$
\begin{aligned}
\mathit{vrai} &::= \forall u.u = u \\
\mathit{src}(U, U') &::= \forall u'.(u' \in V) \leftrightarrow (u' \in U \wedge \forall u''.u'' \in U \rightarrow \neg(u'' < u')) \\
U = U' \cup U'' &::= \forall u.(u \in U) \leftrightarrow (u \in U' \vee u \in U'') \\
U = U' \cap U'' &::= \forall u.(u \in U) \leftrightarrow (u \in U' \wedge u \in U'') \\
U = U' \setminus U'' &::= \forall u.(u \in U) \leftrightarrow (u \in U' \wedge \neg(u \in U'')) \\
\mathit{dunion}(U, U', U'') &::= (\varnothing = U' \cap U'') \wedge (U = U' \cup U'') \\
\mathit{singl\_etiq}_a(U) &::= (\forall u, u'.(u \in U \wedge u' \in U) \rightarrow (u = u')) \wedge (\forall u.u \in U \rightarrow \mathsf{etq}_a(u))
\end{aligned}
$$

### Le fragment sans quantification $\mathrm{LS}_{|\exists}$

La quantification de la logique est connue être une source d'indécidabilité pour le problème de satisfiabilité (voir [Charatonik et al., 2003] et la section 5.1). Par contre, il n'est pas connu si en absence de quantification on peut avoir des fragments décidables de la logique. La première partie de la thèse est consacrée à l'étude de la satisfiabilité de la logique en absence de quantification. La logique $\mathrm{LS}_{|\exists}$ n'est pas un fragment purement syntaxique de la logique spatiale : nous modifions légèrement l'opérateur d'emboîtement.

Soit le nouveau constructeur de formules $\alpha[\phi]$, où $\alpha$ est sous ensemble fini ou co-fini de $\Sigma$. L'interprétation de la formule $\alpha[\phi]$ est donnée par :

$$[\![\alpha[\phi]]\!]_\delta = \alpha[[\![\phi]\!]_\delta]$$

Alors la logique $\mathrm{LS}_{|\exists}$ est définie par la syntaxe :

$$\phi ::= \mathbf{0} \mid \alpha[\phi] \mid \phi|\phi \mid \top \mid \neg\phi \mid \phi \vee \phi \mid \mu\xi.\phi$$

Remarquons que l'ajout du constructeur $\alpha[\phi]$ n'ajoute pas d'expressivité à la logique spatiale en présence de quantification, puisque $\alpha[\phi]$ peut être exprimé en utilisant la quantification : si $\alpha$ est l'ensemble fini $\{a_1, \ldots, a_k\}$, alors $\alpha[\phi] = a_1[\phi] \vee \cdots \vee a_k[\phi]$ et si $\alpha$ est l'ensemble co-fini $\Sigma \setminus \{a_1, \ldots, a_k\}$, alors $\alpha[\phi] = \exists x.(\neg(x = a_1) \wedge \cdots \wedge \neg(x = a_k) \wedge x[\phi])$.

Pour $\alpha$ un ensemble d'étiquettes fini ou cofini, nous notons $\overline{\alpha}$ son complémentaire : $\overline{\alpha} = \Sigma \setminus \alpha$. Nous avons l'équivalence suivante :

**Lemme 3.4** *Pour tout arbre $t$, toute formule $\phi$ et toutes valuations $\rho, \delta$,*

$$t, \rho, \delta \models \neg\alpha[\phi] \Leftrightarrow t, \rho, \delta \models \mathbf{0} \vee \overline{\mathbf{0}} \,|\, \overline{\mathbf{0}} \vee \overline{\alpha}[\top] \vee \alpha[\neg\phi].$$

*Preuve* Par définition, $t, \rho, \delta \models \alpha[\phi]$ si et seulement si $t = \{\!|a[t']|\!\}$ et $a \in \alpha$ et $t', \rho, \delta \models \phi$. Donc, $t, \rho, \delta \models \neg\alpha[\phi]$ si et seulement si

($\star$) $t$ n'est pas un multiensemble à un unique élément ou bien

($\star\star$) $t$ est un multiensemble à un unique élément, soit $b[t']$, tel que $b[t'], \rho, \delta$ ne satisfait pas $\alpha[\phi]$.

Par les propriétés de la satisfiabilité, il est facile de voir que ($\star$) est équivalent à $t \models \mathbf{0} \vee \overline{\mathbf{0}} \,|\, \overline{\mathbf{0}}$ et ($\star\star$) est équivalent à $b \in \overline{\alpha}$ ou $b \in \alpha$ et $t', \rho, \delta \not\models \phi$. Il est facile de voir que c'est équivalent à $t, \rho, \delta \models \overline{\alpha}[\top] \vee \alpha[\neg\phi]$.

□

De même que pour les propriétés énoncées dans le lemme 3.3, ce résultat sera utilisé explicitement lorsqu'il s'agira de la logique $\mathrm{LS}_{|\exists}$.

**Exemple 3.4** Propriétés exprimables sans quantification.

(i) Les successeurs de la racine sont tous étiquetés par $a$

$$\mu\xi.a[\top] \,|\, \xi \vee \mathbf{0}$$

(ii) Arbres à couches d'arêtes étiquetées par $a$ et arêtes étiquetées par $b$, càd
- toutes les arêtes successeurs de la racine sont étiquetés par $a$ et
- toutes les arêtes successeurs d'une arête étiquetée par $a$ sont étiquetées par $b$ et vice versa.

$$\mu\xi.a[\mu\xi'.b[\xi] \,|\, \xi' \vee \mathbf{0}] \,|\, \xi \vee \mathbf{0}$$

(iii) Arbres dans lesquels chaque nœud a autant d'arêtes successeurs étiquetés par $a$ que d'arêtes successeurs étiquetés par $b$

$$\mu\xi.a[\xi] \,|\, b[\xi] \,|\, \xi \vee \mathbf{0}$$

Remarquons que la présence de variables d'étiquettes et d'arbres est le mécanisme le plus naturel permettant de définir des requêtes non booléennes.

**La fragment sans récursion $\mathrm{LS}_{|\mu}$**

Un autre fragment intéressant de la logique est le fragment sans récursion. La récursion est un mécanisme ajoutant de l'expressivité de la logique ; il permet par exemple de définir des répétitions non bornées dans la structure des arbres ou bien d'exprimer des propriétés qui sont vérifiées à une profondeur quelconque de l'arbre (éloigné de la racine de manière non bornée). La combinaison de la récursion et de la quantification ajoute encore plus d'expressivité. Elle permet par exemple d'exprimerla répétition d'un motif non complètement défini. Pour illustration des ces intuitions, voir l'exemple 3.5.

La logique $\mathrm{LS}_{|\mu}$ est définie par la syntaxe :

$$\phi ::= \mathbf{0} \mid \eta[\phi] \mid \phi|\phi \mid \top \mid \neg\phi \mid \phi \vee \phi \mid X \mid \exists X.\phi \mid \exists x.\phi \mid \eta = \eta$$
$$\eta ::= a \mid x.$$

**Exemple 3.5** Propriétés exprimables dans la logique sans récursion.

(i) Toutes les arêtes successeurs de la racine portent la même étiquette

$$\forall x.\forall y.(x[\top]\,|\,y[\top]\,|\,\top) \rightarrow x = y$$

(ii) La logique permet d'exprimer l'isomorphisme d'arbres : deux arbres $s, t$ sont isomorphes si et seulement si l'arbre $\{\!|a[t]|\!\} \uplus \{\!|b[s]|\!\}$ satisfait la formule

$$\exists X.a[X]\,|\,b[X]$$

### 3.2.6 Exemples

Nous donnons ici des exemples de propriétés exprimables dans la logique spatiale.

(i) Arbres dans lesquels tout nœud (à l'exception des feuilles) a exactement trois arêtes successeurs, l'une étiquetée par $a$, l'autre étiquetée par $b$ et la troisième ayant une étiquette arbitraire.

$$\mu\xi.\exists x.a[\xi]\,|\,b[\xi]\,|\,x[\xi] \vee \mathbf{0}$$

Cette même propriété peut également être exprimée en utilisant le fragment sans quantification, grâce à la construction $\alpha[\phi]$ :

$$\mu\xi.a[\xi]\,|\,b[\xi]\,|\,\Sigma[\xi] \vee \mathbf{0}$$

(ii) Arbres dans lesquels tout nœud (à l'exception des feuilles) a exactement quatre arêtes successeurs, l'une étiquetée par $a$, l'autre étiquetée par $b$ et les deux autres ayant la même étiquette arbitraire.

$$\mu\xi.\exists x.a[\xi]\,|\,b[\xi]\,|\,x[\xi]\,|\,x[\xi] \vee \mathbf{0}$$

Cet exemple illustre la possibilité de définir des motifs approchés (non totalement définis) reproduits partout dans l'arbre. La propriété énoncée n'est pas exprimable dans la logique sans quantification, qui permet uniquement de tester l'appartenance d'une étiquette à un ensemble, et ne permet pas de comaprer deux étiquettes. Elle illustre la possibilité de définir des motifs approchés (non totalement définis) reproduits partout dans l'arbre.

(iii) Arbres ayant un nœud à trois arêtes successeurs étiquetées par $a$, $b$ et $c$ ; ce nœud se trouve n'importe où dans l'arbre :

$$\mu\xi.\exists x.(x[\xi] \mid \top) \vee (a[\top] \mid b[\top] \mid c[\top])$$

Cette propriété est également exprimable sans quantification à l'aide de la formule

$$\mu\xi.\Sigma[\xi] \vee (a[\top] \mid b[\top] \mid c[\top])$$

(iv) Arbres contenant un chemin dont toutes les arêtes sont étiquetées par le même symbole

$$\exists x.\mu\xi.x[\xi] \mid \top \vee \mathbf{0}$$

(v) Arbres ayant un nœud à trois arêtes successeurs étiquetées par $a$, $b$ et $c$ et tel que toutes les arêtes du chemin dans l'arbre qui mène à ce nœud sont étiquetées par la même étiquette :

$$\exists x.\mu\xi.x[\xi] \mid \top \vee (a[\top] \mid b[\top] \mid c[\top]).$$

Cette propriété ne peut pas être exprimée sans quantification.

# Deuxième partie

# Satisfiabilité et expressivité de la logique spatiale

Dans cette partie nous considérons la logique $LS_{|\exists}$, càd le fragment de la logique spatiale sans quantification et nous étudions le problème de satisfiabilité de cette logique et de certains de ses fragments, ainsi que l'expressivité de la logique. Rappelons que le problème de satisfiabilité est, étant donné une formule close $\phi$, décider si l'interprétation de cette formule est l'ensemble vide (càd décider si $[\![\phi]\!]$ est l'ensemble vide). Nous dirons que le problème de satisfiabilité est décidable si on peut décider de cette propriété, et qu'il est indécidable sinon. L'expressivité de la logique $LS_{|\exists}$ est étudiée conjointement avec le problème de satisfiabilité, et en comparaison avec les logiques MSO et PMSO.

Cette partie est organisée comme suit : dans le chapitre 4 nous présentons essentiellement des résultats techniques utilisés dans la suite, dont une façon de représenter les formules $LS_{|\exists}$ par des équations à point fixe et des propriétés de ces systèmes d'équations. Dans le chapitre 5, nous rappelons le résultat que le problème de satisfiabilité est indécidable pour la logique spatiale avec quantification et nous montrons que la satisfiabilité de la logique $LS_{|\exists}$ est également indécidable. Dans le chapitre 6 nous montrons comment définir un automate à contraintes numériques équivalent à une formule $LS_{|\exists}$. Finalement, dans le chapitre 7 nous utilisons les automates à contraintes numériques pour identifier deux fragments de la logique $LS_{|\exists}$ qui sont équivalent aux logiques MSO et PMSO respectivement.

Dans cette partie de la thèse, par « formule logique » nous entendons une formule de $LS_{|\exists}$ (sauf dans la section 5.1 où nous rappelons quelques résultats d'indécidabilité pour la logique avec quantification).

# Chapitre 4

# Formules logiques et systèmes d'équations à point fixe

Dans ce chapitre nous présentons une représentation des formules logiques sous la forme de systèmes d'équations à points fixes. Cette représentation est utilisée dans la suite pour la preuve de divers résultats. Nous donnons également des représentations normalisées de ces systèmes d'équations, qui seront utilisées plus tard pour la définition d'un automate pour une formule logique, mais qui nous permettent également d'exprimer quelques conditions qui garantissent l'unicité du point fixe pour les formules logiques.

Le chapitre est organisé comme suit : dans la section 4.1 nous définissons les notions de base liées aux systèmes d'équations à points fixes. Dans la section 4.2 nous introduisons un type de systèmes d'équations qui peut servir de représentation des formules de la logique, et nous montrons comment construire un système d'équations à partir d'une formule et inversement. Dans la section 4.3 nous présentons deux formes normales pour les systèmes d'équations. Finalement, dans la section 4.4 nous donnons quelques conditions garantissant que le plus petit et le plus grand points fixes coïncident.

## 4.1 Généralités sur les systèmes d'équations à point fixe

Cette section commence avec les définitions de quelques notions de base : treillis, fonctions monotones, points fixes, avant de définir un système d'équations à point fixe. Nous donnons ensuite quelques propriétés des systèmes d'équations qui seront utiles pour la suite. Pour plus d'informations sur ces notions, le lecteur peut se référer à [Arnold and Niwinski, 2001].

### 4.1.1 Définitions

#### Treillis, monotonie, points fixes

Un *treillis* $\langle H, \leq \rangle$ est un ensemble $H$ muni d'un ordre partiel $\leq$ et tel que pour tout couple d'éléments $h, l$ appartenant à $H$, l'ensemble $\{h, l\}$ admet une plus petite (pour l'ordre $\leq$) borne supérieure notée $h \vee_{\leq} l$ et une plus grande (pour l'ordre $\leq$) borne inférieure notée $h \wedge_{\leq} l$. Un

*treillis complet* est un treillis $\langle H, \leq \rangle$ tel que tout $L$ sous ensemble de $H$ admet une plus petite borne supérieure et une plus grande borne inférieure, notées $\bigvee_{\leq} L$ et $\bigwedge_{\leq} L$ respectivement. Nous notons alors ce treillis $\langle H, \vee_{\leq}, \wedge_{\leq} \rangle$, ou bien $\langle H, \vee, \wedge \rangle$ si la relation d'ordre $\leq$ est claire dans le contexte. Il est facile de voir que pour tout ensemble $F$, $\langle \wp(F), \vee_{\subseteq}, \wedge_{\subseteq} \rangle$ est un treillis complet, où $\subseteq$ est l'inclusion d'ensembles et $\vee_{\subseteq}$ et $\wedge_{\subseteq}$ sont l'union et intersection d'ensembles respectivement.

Soit $\langle H, \leq \rangle$ un ensemble muni d'un ordre partiel. La fonction $f : H \to H$ est dite *monotone* si pour tout $h, l$ dans $H$, $h \leq l$ implique $f(h) \leq f(l)$. Soit $n$ un entier naturel. La fonction $f : H^n \to H$ est dite *monotone dans toutes les variables* si pour tous $h_1, \ldots, h_n, l_1 \ldots, l_n$ éléments de $H$, $h_1 \leq l_1$ et ... et $h_n \leq l_n$ implique $f(h_1, \ldots, h_n) \leq f(l_1, \ldots, l_n)$.

Soit $f$ une fonction de $H$ dans $H$ (pour un ensemble $H$ quelconque). L'élément $h$ de $H$ est dit *point fixe de* $f$ si $f(h) = h$. L'ensemble des points fixes de $f$ est noté $Fix(f)$.

Le théorème de Knaster-Tarki établit que toute fonction monotone sur un treillis complet admet un plus grand point fixe et un plus petit point fixe, notés $\bigvee_{\leq} Fix(f)$ et $\bigwedge_{\leq} Fix(f)$.

**Théorème 4.1 (Knaster-Tarski)** *Soit $\langle H, \vee_{\leq}, \wedge_{\leq} \rangle$ un treillis complet et $f$ une fonction monotone de $H$ dans $H$. Alors $\bigvee_{\leq} Fix(f)$ et $\bigwedge_{\leq} Fix(f)$ appartiennent à $Fix(f)$. De plus,*

$$\bigvee_{\leq} Fix(f) = \bigvee_{\leq} \{h \in H \mid h \leq f(h)\} \qquad et \qquad \bigwedge_{\leq} Fix(f) = \bigwedge_{\leq} \{h \in H \mid f(h) \leq h\}.$$

Le plus grand et plus petit point fixe de la fonction $f$ sont usuellement notés $\nu x.f(x)$ et $\mu x.f(x)$. Soit maintenant $f$ une fonction de $H^n$ dans $H$. Nous pouvons également définir le plus petit ou plus grand point fixe de $f$ pour une de ses variables : $\mu x_i.f(x_1, \ldots, x_i, \ldots, x_n)$ est une fonction de $H^{n-1}$ dans $H$ qui à tout $(n-1)$-uplet $(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$ dans $H^{n-1}$ associe la valeur $\mu x_i.f(s_1, \ldots, s_{i-1}, x_i, s_{i+1}, \ldots, s_n)$. Une propriété importante est que le point fixe par rapport à une composante préserve la monotonie :

**Proposition 4.2 ([Arnold and Niwinski, 2001], Proposition 1.2.23)** *Soit $H$ un treillis complet et soit $f$ une fonction de $H^n$ dans $H$ monotone dans toutes les variables. Alors pour tout $i$ dans $1..n$, $\mu x_i.f(x_1, \ldots, x_i, \ldots, x_n)$ et $\nu x_i.f(x_1, \ldots, x_i, \ldots, x_n)$ sont des fonctions de $H^{n-1}$ dans $H$ monotones dans toutes les variables.*

Une conséquence de ce résultat est que l'on peut imbriquer les points fixes : càd nous pouvons considérer les fonctions $\mu x_i.\nu x_j.f(x_1, \ldots, x_n)$, $\nu x_j.\nu x_i.f(x_1, \ldots, x_n)$, etc.

### Points fixes sur un treillis produit

Nous résumons ici quelques notions liées aux points fixes sur des treillis produits. Pour plus de détails, voir la section 1.4 de [Arnold and Niwinski, 2001].

Soit $\langle H, \vee_{\leq}, \wedge_{\leq} \rangle$ un treillis complet. Alors l'ensemble $H^n$ muni de la relation d'ordre $\leq_n$ qui est l'extension habituelle de $\leq$ sur les tuples de valeurs de $H$ forme également un treillis complet. Soient $f_1, \ldots, f_n$ des fonctions de $H^n$ dans $H$ monotones dans toutes les variables. Alors le tuple de fonctions $\mathbf{f} = \langle f_1, \ldots, f_n \rangle$ définit une fonction monotone de $H^n$ dans $H^n$ de la façon suivante : pour tout $\mathbf{h} = \langle h_1, \ldots, h_n \rangle$ éléments de $H^n$ et pour tout $j$ dans $1..n$, la $j^{\text{ème}}$ composante de $\mathbf{f}(\mathbf{h})$ est $f_j(h_1, \ldots, h_n)$.

Comme $H^n$ est un treillis complet et la fonction $\mathbf{f}$ est monotone, on peut calculer le point fixe de $\mathbf{f}$, qui est un élément de $H^n$ ; ainsi le plus petit et le plus grand points fixes de $\mathbf{f}$ sont $\mu\mathbf{x}.\mathbf{f}(\mathbf{x})$ et $\nu\mathbf{x}.\mathbf{f}(\mathbf{x})$ respectivement. Ces points fixes peuvent être représentés de façon alternative comme la solution extrême du système de $n$ équations à $n$ inconnues

$$\begin{aligned} x_1 &= f_1(x_1, \ldots, x_n) \\ &\cdots \\ x_n &= f_n(x_1, \ldots, x_n) \end{aligned}$$

où les $x_1, \ldots, x_n$ sont des variables sur $H$.

La solution d'un tel système d'équation peut être calculée par ce qu'on appelle le *principe d'élimination de Gauss*. Supposons qu'on cherche à calculer le vecteur $\langle a_1, \ldots, a_n \rangle$, le $\kappa$ point fixe de l'équation ci-dessus (pour $\kappa$ étant $\mu$ ou $\nu$). En posant $g(x_2, \ldots, x_n) = \kappa x_1.f_1(x_1, \ldots, x_n)$, nous avons que $\langle a_2, \ldots, a_n \rangle$ est la « $\kappa$-solution » (càd la plus petite ou la plus grande solution) du système à $n-1$ équations

$$\begin{aligned} x_2 &= f_2(g(x_2, \ldots, x_n), x_2, \ldots, x_n) \\ &\cdots \\ x_n &= f_n(g(x_2, \ldots, x_n), x_2, \ldots, x_n) \end{aligned}$$

Dans ce cas, $a_1 = g(a_2, \ldots, a_n)$.

### Systèmes d'équations à points fixes

Les systèmes d'équations à point fixe sont une généralisation des systèmes d'équations présentés dans la section précédente. Dans un système d'équations à points fixes, les points fixes sont associés aux équations du système et non au système en entier.

**Définition 4.3 (Système d'équations à point fixe)** Soient $\langle H, \vee, \wedge \rangle$ et $\langle H', \vee', \wedge' \rangle$ des treillis complets et $f_1, \ldots, f_n$ des fonctions de $H^n \times H'$ dans $H$ monotones dans toutes les variables, et soient $x_1, \ldots, x_n, x$ des variables prenant leurs valeurs dans $H$ et $x$ une variable prenant ses valeurs dans $H'$. Un *système d'équations à point fixe* $\mathcal{S}$ est une séquence

$$x_1 \stackrel{\kappa_1}{=} f_1(x_1, \ldots, x_n, x), \cdots, x_i \stackrel{\kappa_i}{=} f_i(x_1, \ldots, x_n, x), \cdots, x_n \stackrel{\kappa_n}{=} f_n(x_1, \ldots, x_n, x)$$

où chacun des $\kappa_i$ est égal à $\mu$ ou à $\nu$.

La *solution* de $\mathcal{S}$ est le $n$-uplet défini par : si $n = 1$, alors la solution du système $x_1 \stackrel{\kappa_1}{=} f_1(x_1, x)$ est $\langle \kappa_1 x_1.f_1(x_1, x) \rangle$. Sinon, la solution du système

$$x_1 \stackrel{\kappa_1}{=} f_1(x_1, \ldots, x_n, x), \cdots, x_i \stackrel{\kappa_i}{=} f_i(x_1, \ldots, x_n, x), \cdots, x_n \stackrel{\kappa_n}{=} f_n(x_1, \ldots, x_n, x)$$

est $\langle g_1(h_2(x), \ldots, h_n(x), x), h_2(x), \ldots, h_n(x) \rangle$ où $g_1(x_2, \ldots, x_n, x) = \kappa_1 x_1.f_1(x_1, \ldots, x_n, x)$ et $\langle h_2(x), \ldots, h_n(x) \rangle$ est la solution de

$$x_2 \stackrel{\kappa_2}{=} f_2(g_1(x_2, \ldots, x_n, x), x_2, \ldots, x_n, x), \cdots, x_n \stackrel{\kappa_n}{=} f_n(g_1(x_2, \ldots, x_n), x_2, \ldots, x_n, x).$$

Pour les systèmes d'équations que nous aurons à considérer dans la suite, les fonctions $f_i$ ci-dessus ne seront pas des fonctions de la variable $x$. C'est pourquoi, nous omettons cette variable dans ce qui suit, càd un système d'équations sera une séquence de la forme

$$x_1 \stackrel{\kappa_1}{=} f_1(x_1, \ldots, x_n), \cdots, x_i \stackrel{\kappa_i}{=} f_i(x_1, \ldots, x_n), \cdots, x_n \stackrel{\kappa_n}{=} f_n(x_1, \ldots, x_n).$$

Dans ce cas, la solution d'un système d'équations est un $n$-uplet de fonctions constantes (puisque ne dépendant pas de $x$), et donc un élément de $H^n$.

Pour le système d'équations $\mathcal{S}$ étant $x_1 \stackrel{\kappa_1}{=} f_1(x_1, \ldots, x_n), \cdots, x_n \stackrel{\kappa_n}{=} f_n(x_1, \ldots, x_n)$, on note $\mathsf{Vars}(\mathcal{S})$ la séquence de variables $x_1, \ldots, x_n$ et $\{\mathsf{Vars}(\mathcal{S})\}$ l'ensemble de variables $\{x_1, \ldots, x_n\}$. La variable $x_n$ sera désignée par $\mathsf{dern}(\mathcal{S})$. La solution du système d'équations $\mathcal{S}$ est notée $Sol(\mathcal{S})$ et pour toute variable $x$ dans $\mathsf{Vars}(\mathcal{S})$, on écrit $Sol(\mathcal{S}, x)$ pour désigner la composante de la solution du système $\mathcal{S}$ correspondant à la variable $x$.

Il apparaît de la définition pour la solution d'un système d'équations que l'ordre d'élimination des variables pour le calcul de la solution a une importance. En effet, en utilisant un autre ordre d'élimination des variable on peut obtenir une solution différente, comme le montre l'exemple suivant.

**Exemple 4.1** Soit $\langle H, \leq \rangle$ un ensemble ordonné d'au moins deux éléments et tel que $\langle H, \vee_{\leq}, \wedge_{\leq} \rangle$ est un treillis complet. Nous notons $\top$ l'élément $\bigvee_{\leq} H$ et $\bot$ l'élément $\bigwedge_{\leq} H$. Il est facile de voir que les plus petit et plus grand points fixes de la fonction identité sont respectivement : $\mu x.x = \bot$ et $\nu x.x = \top$.

Soient $\mathcal{S}$ le système d'équations $x \stackrel{\mu}{=} y, y \stackrel{\nu}{=} x$. En calculant la solution de $\mathcal{S}$ éliminant la variable $x$ en premier, nous obtenons que $Sol(\mathcal{S}, x)$ est égal à $\mu x.y$ et donc à $y$. En remplaçant ceci dans la deuxième équation, nous obtenons que la solution de $\mathcal{S}$ pour $y$ est $\nu y.y = \top$. Et comme $Sol(\mathcal{S}, x) = Sol(\mathcal{S}, y)$, nous obtenons $\langle \top, \top \rangle$ comme solution du système. Calculons la solution de $\mathcal{S}$ en éliminant la variable $y$ en premier : $Sol(\mathcal{S}, y)$ est $\nu y.x$ qui est égal à $x$. Donc, en remplaçant dans l'équation pour $x$, nous avons que $Sol(\mathcal{S}, x)$ est $\mu x.x$ qui est égal à $\bot$. Alors la solution du système d'équations est $\langle \bot, \bot \rangle$.

### Solution extrême d'un système d'équations vs. solution d'un système d'équations à points fixes

Nous tenons à faire ici la distinction et le lien entre les deux types de systèmes d'équations qui ont été définis dans les deux sections précédentes. Il s'agit dans un cas de calculer une solution extrême d'un système d'équations et d'autre part de calculer la solution d'un système dans lequel des opérateurs de point fixes sont associés à chacune des équations. Les méthodes de résolution qui ont été proposées pour ces deux systèmes sont ressemblantes. Néanmoins, le principe d'élimination de Gauss qui a été proposé dans le premier cas peut être appliqué en éliminant les variables dans un ordre quelconque, alors que, comme nous l'avons vu dans l'exemple 4.1, l'élimination des variables dans un systèmes d'équations à points fixes doit se faire dans un ordre précis.

Même si cela n'apparaît pas directement des solutions, il n'est pas très difficile de voir que le lien qui existe entre ces deux types de systèmes d'équations est le suivant : la solution du système d'équations à points fixes

$$x_1 \stackrel{\kappa}{=} f_1(x_1, \ldots, x_n), \ \cdots, \ x_n \stackrel{\kappa}{=} f_n(x_1, \ldots, x_n)$$

est égale à la $\kappa$-solution du système d'équations

$$\begin{aligned} x_1 &= f_1(x_1, \ldots, x_n) \\ &\ldots \\ x_n &= f_n(x_1, \ldots, x_n) \end{aligned}$$

Il en découle que, si dans un système d'équations à points fixes un seul opérateur de point fixe est utilisé, alors sa solution peut être calculée en éliminant les variables dans n'importe quel ordre.

Dans la suite, nous utilisons principalement des systèmes d'équations à points fixes, que nous appelons « systèmes d'équations ». Pour les rares cas où nous considérons l'extrême solution d'un système d'équations sur un treillis produit, nous le préciserons explicitement.

### 4.1.2 Propriétés des systèmes d'équations

Tous les résultats énoncés dans cette section peuvent être trouvés dans le premier chapitre de [Arnold and Niwinski, 2001].

Cette première proposition montre que la solution d'un système d'équations $\mathcal{S}$ peut être calculée par blocs : $\mathcal{S}$ est découpé en deux sous-systèmes dont les solutions sont calculées de façon relativement indépendante et ensuite combinées pour obtenir la solution de $\mathcal{S}$.

**Proposition 4.4 ([Arnold and Niwinski, 2001], Proposition 1.4.13)** *La solution du système d'équations*

$$x_1 \overset{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots, x_i \overset{\kappa_i}{=} f_i(x_1, \ldots, x_n), \ldots, x_n \overset{\kappa_n}{=} f_n(x_1, \ldots, x_n)$$

*est égale à*

$$\langle h_1(h'_{i+1}, \ldots, h'_n), \ldots, h_i(h'_{i+1}, \ldots, h'_n), h'_{i+1}, \ldots, h'_n \rangle$$

*où* $\langle h_1(x_{i+1}, \ldots, x_n), \ldots, h_i(x_{i+1}, \ldots, x_n) \rangle$ *est la solution de*

$$x_1 \overset{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots, x_i \overset{\kappa_i}{=} f_i(x_1, \ldots, x_n)$$

*et* $\langle h'_{i+1}, \ldots, h'_n \rangle$ *est la solution de*

$$\begin{aligned} x_{i+1} &\overset{\kappa_{i+1}}{=} f_{i+1}(h_1(x_{i+1}, \ldots, x_n), \ldots, h_i(x_{i+1}, \ldots, x_n), x_{i+1}, \ldots, x_n), \\ &\ldots, \\ x_n &\overset{\kappa_n}{=} f_n(h_1(x_{i+1}, \ldots, x_n), \ldots, h_i(x_{i+1}, \ldots, x_n), x_{i+1}, \ldots, x_n) \end{aligned}$$

La proposition suivante montre deux façons de modifier un système d'équations en préservant sa solution.

**Proposition 4.5 ([Arnold and Niwinski, 2001], Proposition 1.4.14)** *Pour tout* $i \leq j$, *les trois systèmes*

$$\begin{aligned} x_1 &\overset{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots \\ &\ldots, x_j \overset{\kappa_j}{=} f_j(x_1, \ldots, x_n), \ldots \\ &\ldots, x_n \overset{\kappa_n}{=} f_n(x_1, \ldots, x_n) \end{aligned}$$

$$x_1 \overset{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots$$
$$\ldots, x_j \overset{\kappa_j}{=} f_j(x_1, \ldots, x_{i-1}, f_i(x_1, \ldots, x_n), x_{i+1}, \ldots, x_n), \ldots$$
$$\ldots, x_n \overset{\kappa_n}{=} f_n(x_1, \ldots, x_n)$$

$$x_1 \overset{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots$$
$$\ldots, x_j \overset{\kappa_j}{=} f_j(x_1, \ldots, x_{i-1}, \kappa_i x_i . f_i(x_1, \ldots, x_n), x_{i+1}, \ldots, x_n), \ldots$$
$$\ldots, x_n \overset{\kappa_n}{=} f_n(x_1, \ldots, x_n)$$

*ont la même solution.*

Nous terminons par un résultat qui peut être facilement déduit des résultats présentés dans [Arnold and Niwinski, 2001]. Nous omettons la preuve de ce corollaire, qui demanderait d'introduire plusieurs notations non nécessaires pour le reste de la thèse, ainsi que de reproduire presque la totalité des résultats du premier chapitre de [Arnold and Niwinski, 2001].

**Corollaire 4.6** *Soit $\mathcal{S}$ le système d'équations*

$$x_1 \overset{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots, x_i \overset{\kappa_i}{=} f_i(x_1, \ldots, x_n), \ldots, x_n \overset{\kappa_n}{=} f_n(x_1, \ldots, x_n)$$

*et soit $\mathbf{h} = \langle h_1, \ldots, h_n \rangle$ sa solution. Alors pour tout $I$ sous ensemble de $\{1, \ldots, n\}$, le système $\mathcal{S}$ a la même solution que le système*

$$x_1 \overset{\kappa_1}{=} f_1(y_1, \ldots, y_n), \ldots, x_i \overset{\kappa_i}{=} f_i(y_1, \ldots, y_n), \ldots, x_n \overset{\kappa_n}{=} f_n(y_1, \ldots, y_n)$$

*où pour tout $j$, $y_j = x_j$ si $j \notin I$ et $y_j = h_j$ si $j \in I$.*

Ce corollaire se déduit des propositions 1.4.1 et 1.4.11 de [Arnold and Niwinski, 2001].

## 4.2 Formules logiques et systèmes d'équations

Nous montrons ici que toute formule logique close peut être représentée par un système d'équations à point fixe. Plus précisément, pour toute formule logique $\phi$ on peut construire le système d'équations $\mathsf{Eq}(\phi)$ tel que $[\![\phi]\!]$ est égal à la solution de $\mathsf{Eq}(\phi)$ pour la dernière variable dans $\mathsf{Vars}(\mathsf{Eq}(\phi))$. Inversement, pour tout système d'équations $\mathcal{S}$ on peut construire une formule logique close $\mathsf{formule}(\mathcal{S})$ telle que la solution de $\mathcal{S}$ pour sa dernière variable est l'ensemble $[\![\mathsf{formule}(\mathcal{S})]\!]$.

Nous commençons par donner quelques définitions dans la section 4.2.1. Dans la section 4.2.2 nous définissons une famille de fonctions monotones qui seront utilisées comme parties droites d'équations dans le système $\mathsf{Eq}(\phi)$. Ensuite, dans la section 4.2.3, nous montrons comment calculer la formule correspondant à un système d'équations et finalement dans la section 4.2.4 donnons une méthode pour construire le système d'équations $\mathsf{Eq}(\phi)$.

### 4.2.1 Termes fonctionnels et termes à point fixe

Soit $\sigma$ une signature, càd un ensemble de symboles fonctionnels ayant chacun une arité fixe, et soit $\mathcal{V}$ un ensemble dénombrable de variables.

**Définition 4.7 (Terme fonctionnel)** L'ensemble $fonct(\sigma, \mathcal{V})$ des *termes fonctionnels sur* $\sigma$ *et* $\mathcal{V}$ est défini récursivement par :
- toute variable dans $\mathcal{V}$ est un terme fonctionnel ;
- si $op \in \sigma$ est un symbole d'arité $n$ et $f_1, \ldots, f_n$ sont des termes fonctionnels, alors $op(f_1, \ldots, f_n)$ est un terme fonctionnel.

**Définition 4.8 (Terme à point fixe)** L'ensemble $fixpt(\sigma, \mathcal{V})$ des *termes à point fixe sur* $\sigma$ *et* $\mathcal{V}$ est défini récursivement par :
- toute variable dans $\mathcal{V}$ est un terme à point fixe ;
- si $op \in \sigma$ est un symbole d'arité $n$ et $f_1, \ldots, f_n$ sont des termes à point fixe, alors $op(f_1, \ldots, f_n)$ est un terme à point fixe ;
- si $x \in \mathcal{V}$ et $f$ est un terme à point fixe, alors $\mu x.f$ et $\nu x.f$ sont des termes à point fixe.

Il est évident que $fonct(\sigma, \mathcal{V}) \subseteq fixpt(\sigma, \mathcal{V})$ pour toute signature $\sigma$ et tout ensemble de variables $\mathcal{V}$.

Les notions d'occurrence de variable libre et occurrence de variable liée dans un terme sont définies comme d'habitude, sachant que les opérateurs $\mu$ et $\nu$ lient toutes les occurrences libres de la variable $x$ dans les termes à point fixe $\mu x.f$ ou $\nu x.f$. Par simplicité, nous supposons que dans tout terme, aucune variable n'apparaît à la fois à des occurrences libres et liées, ce qui nous permet de parler de variables libres et variables liées au lieu d'occurrences libres et occurrences liées.

Soient $f, f'$ des termes fonctionnels (respectivement des termes à point fixe), et soit $x$ une variable. Nous notons $f\langle x \rightarrow f' \rangle$ le terme fonctionnel (respectivement le terme à point fixe) obtenu en remplaçant dans $f$ les occurrences libres de la variable $x$ par $f'$.

**Définition 4.9 ($\mu$-interprétation)** Soit $\sigma$ une signature dénombrable. Une *$\mu$-interprétation* $\mathcal{I}$ des symboles de $\sigma$ est un couple $(\langle D, \vee_\leq, \wedge_\leq \rangle, I)$ où $\langle D, \vee_\leq, \wedge_\leq \rangle$ est un treillis complet et $I$ est une fonction qui à tout symbole $op$ d'arité $k$ dans $\sigma$ associe une fonction monotone de $D^k$ dans $D$ notée $op^{\mathcal{I}}$.

Pour toute signature $\sigma$ et toute $\mu$-interprétation $\mathcal{I}$ de $\sigma$, cette interprétation est étendue aux termes fonctionnels de $fonct(\sigma, \mathcal{V})$ comme une composition de fonctions ; dans ce cas pour tout terme $f$ dans $fonct(\sigma, \mathcal{V})$, nous notons $f^{\mathcal{I}}$ l'interprétation de $f$ par $\mathcal{I}$ qui est une fonction de $D^k$ dans $D$, où $k$ est le nombre de variables libres de $f$. Remarquons que si $\mathcal{I}$ est une $\mu$-interprétation, alors la fonction $f^{\mathcal{I}}$ est monotone dans toutes les variables. De ce fait, $\mathcal{I}$ peut également être étendue aux termes à point fixe de $fixpt(\sigma, \mathcal{V})$ : pour tout terme $f$ et toute variable $x$, $(\mu x.f)^{\mathcal{I}}$ est la fonction $\mu x.(f)^{\mathcal{I}}$.

Si $\mathcal{I}$ est une $\mu$-interprétation de $\sigma$ et $f$ est un terme dans $fixpt(\sigma, \mathcal{V})$ de variables libres $x_1, \ldots, x_k$, nous notons $f^{\mathcal{I}}(x_1, \ldots, x_k)$ l'interprétation de $f$ par $\mathcal{I}$ si nous voulons rendre explicites les variables de $f^{\mathcal{I}}$. Dans ce cas, si $s_1, \ldots, s_k$ sont des valeurs dans $D$, nous notons $f^{\mathcal{I}}(s_1, \ldots, s_k)$ l'application de cette fonction aux valeurs $s_1, \ldots, s_k$. D'autre part, si $\gamma$ est une

valuation de domaine $x_1, \ldots, x_k$ qui à $x_i$ associe la valeur $s_i$, alors $[\![f]\!]^{\mathcal{I}}_{\gamma}$ est notation alternative à $f^{\mathcal{I}}(s_1, \ldots, s_k)$. Dans cette dernière notation, nous omettrons $\mathcal{I}$ lorsqu'il est clair dans le contexte.

Remarquons également que

**Remarque 4.10** *Pour toute signature dénombrable $\sigma$, toute $\mu$-interprétation $\mathcal{I}$ de $\sigma$ et pour tous termes $f, f' \in \mathit{fixpt}(\tau, X)$ tels que les ensembles des variables libres de $f$ et $f'$ sont inclus dans $\{x_1, \ldots, x_n\}$, pour tout $x_i$ dans $\{x_1, \ldots, x_n\}$, la fonction $(f\langle x_i \to f'\rangle)^{\mathcal{I}}(x_1, \ldots, x_i, \ldots, x_n)$ est équivalente à la fonction $f^{\mathcal{I}}(x_1, \ldots, x_{i-1}, f'^{\mathcal{I}}(x_1, \ldots, x_i, \ldots, x_n), x_{i+1}, \ldots, x_n)$.*

Considérons maintenant la séquence

$$x_1 \overset{\kappa_1}{=} f_1 \ , \ \ldots \ , \ x_n \overset{\kappa_n}{=} f_n \tag{4.1}$$

où les $x_i$ sont des variables dans un ensemble de variables $X$ et les $f_i$ sont des termes sur $\mathit{fonct}(\sigma, X)$ tels que pour tout $i$ dans $1..n$, l'ensemble des variables libres de $f_i$ est inclus dans $\{x_1, \ldots, x_n\}$ et où $\sigma$ est une signature finie. Étant donné la $\mu$-interprétation $\mathcal{I} = (\langle D, \vee_{\leq}, \wedge_{\leq}\rangle, I)$ de la signature $\sigma$, la séquence (4.1) définit le système d'équations $\mathcal{S}$

$$x_1 \overset{\kappa_1}{=} f_1^{\mathcal{I}}(x_1, \ldots, x_n) \ , \ \ldots \ , \ x_n \overset{\kappa_n}{=} f_k^{\mathcal{I}}(x_1, \ldots, x_n)$$

La solution de ce système d'équations est un $n$-uplet sur $D$ et sera notée $\mathit{Sol}_{\mathcal{I}}(\mathcal{S})$. Un tel système est appelé *système d'équations sur* $\mathit{fonct}(\sigma, X)$ si les $f_i$ sont des termes dans $\mathit{fonct}(\sigma, X)$ et est appelé *système d'équations sur* $\mathit{fixpt}(\sigma, X)$ si les $f_i$ sont des termes dans $\mathit{fixpt}(\sigma, X)$. Dans la suite, nous allons librement confondre les systèmes d'équations sur $\mathit{fonct}(\sigma, X)$ et sur $\mathit{fixpt}(\sigma, X)$ avec des séquences comme (4.1). Nous avons besoin de ce point de vue purement syntaxique pour définir des transformations syntaxiques sur les systèmes d'équations. Grâce à la remarque 4.10, toute une famille de transformations syntaxiques préservera la solution d'un système d'équations.

### 4.2.2 Les formules logiques comme des fonctions sur $\wp(\mathcal{A}_\Sigma)$

Soit $\tau$ la signature (infinie) $\{\mathbf{0}, \overline{\mathbf{0}}, \top, \bot, \vee, \wedge, |, \| \} \cup \bigcup_{\alpha \subseteq \Sigma \text{ fini ou cofini}} \{\alpha\}$, où les symboles $\mathbf{0}, \overline{\mathbf{0}}, \bot, \top$ sont des constantes, les symboles $\vee, \wedge, |, \|$ sont binaires utilisés en notation infixe et les symboles $\alpha$, pour tout $\alpha \subseteq \Sigma$ fini ou cofini, sont des symboles d'arité un et seront utilisés avec des crochets au lieu des parenthèses : si $f$ est un terme, on écrit $\alpha[f]$ au lieu de $\alpha(f)$. Considérons la $\mu$-interprétation $\mathcal{T} = (\langle \wp(\mathcal{A}_\Sigma), \vee, \wedge\rangle, T)$ des symboles de $\tau$ dont la définition est donnée sur la figure 1.

Il est facile de voir que $\mathcal{T}$ est une $\mu$-interprétation.

Maintenant, si $\mathcal{V} = \mathcal{X}_{\mathrm{r}}$[1], alors tout terme $f$ dans $\mathit{fonct}(\tau, \mathcal{X}_{\mathrm{r}})$ ou $\mathit{fixpt}(\tau, \mathcal{X}_{\mathrm{r}})$ est une formule logique utilisant les opérateurs duaux. Le lien qui existe entre la fonction $f^{\mathcal{T}}(\xi_1, \ldots, \xi_n)$ et la formule $f$ est le suivant :

pour tous ensembles d'arbres $S_1, \ldots, S_n$, $\quad f^{\mathcal{T}}(S_1, \ldots, S_n) = [\![f]\!]_{[\xi_1 \to S_1] \cdots [\xi_n \to S_n]}$.

[1]Rappelons que $\mathcal{X}_{\mathrm{r}}$ est l'ensemble de variables de récursion sur lequel sont construites les formules logiques.

$$
\begin{array}{lcl}
\mathbf{0}^{\mathcal{T}} & = & \{\{\!|\}\!\} \\
\overline{\mathbf{0}}^{\mathcal{T}} & = & \mathcal{A}_\Sigma \setminus \{\{\!|\}\!\} \\
\top^{\mathcal{T}} & = & \mathcal{A}_\Sigma \\
\bot^{\mathcal{T}} & = & \varnothing \\
\alpha^{\mathcal{T}}[S] & = & \{\{\!|a[t]|\!\} \mid a \in \alpha, t \in S\} \\
S \mid^{\mathcal{T}} S' & = & \{t \uplus t' \mid t \in S, t' \in S'\} \\
S \parallel^{\mathcal{T}} S' & = & \{t'' \mid \forall t, t'.\ t'' = t \uplus t' \Rightarrow (t \in S \text{ ou } t' \in S')\} \\
S \vee^{\mathcal{T}} S' & = & S \cup S' \\
S \wedge^{\mathcal{T}} S' & = & S \cap S'
\end{array}
$$

FIG. 1 – Définition de la $\mu$-interprétation $\mathcal{T}$. Ici $S, S'$ sont des éléments de $\wp(\mathcal{A}_\Sigma)$.

Pour la suite nous convenons que pour tout terme $f$ dans $\mathit{fixpt}(\tau, \mathcal{X}_\mathrm{r})$ à variables libres $x_1, \ldots, x_n$, $f(x_1, \ldots, x_n)$ désignera la fonction $f^{\mathcal{T}}$ et $f(S_1, \ldots, S_n)$ désignera la même fonction appliquée aux ensembles d'arbres $S_1, \ldots, S_n$.

### 4.2.3 Formule associée à un système d'équations

Nous montrons ici comment, étant donné un système d'équations $\mathcal{S}$ sur $f \in \mathit{fonct}(\tau, \mathcal{X}_\mathrm{r})$ on peut construire une formule close notée formule($\mathcal{S}$) telle que $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S})) = [\![\mathsf{formule}(\mathcal{S})]\!]$.

Une *substitution* $\Gamma$ est une séquence finie de la forme $[x_1 \to f_1] \cdots [x_n \to f_n]$, où les $x_i$ sont des variables dans $\mathcal{X}_\mathrm{r}$ distinctes deux à deux et les $f_i$ sont des termes dans $\mathit{fixpt}(\tau, \mathcal{X}_\mathrm{r})$. La séquence vide, ou substitution vide, est notée $\epsilon$.

Pour la formule logique $f$ et la substitution $\Gamma$, l'application de $\Gamma$ à $f$ est une formule notée $\Gamma(f)$ et définie récursivement sur la structure de $\Gamma$ comme suit :

- $\epsilon(f) = f$ ;
- $(\Gamma[x \to f'])(f) = \Gamma(f\langle x \to f' \rangle)$

Autrement dit, le remplacement défini par le dernier élément de $\Gamma$ est appliqué en premier.

Le *domaine* de la substitution $\Gamma$, noté $\mathit{dom}(\Gamma)$, est défini par : $\mathit{dom}(\epsilon) = \varnothing$ et $\mathit{dom}([x \to f]\Gamma) = \{x\} \cup \mathit{dom}(\Gamma)$.

Pour tout système d'équations $\mathcal{S}$ égal à $x_1 \overset{\kappa_1}{=} f_1, \ldots, x_n \overset{\kappa_n}{=} f_n$, on note $\Gamma_{\mathcal{S}}$ la substitution $[x_1 \to \kappa_1 x_1.f_1] \cdots [x_n \to \kappa_n x_n.f_n]$.

**Définition 4.11 (Formule pour un système d'équations)** Soit le système d'équations $\mathcal{S}$ sur $\mathit{fonct}(\tau, \mathcal{X}_\mathrm{r})$ suivant

$$\xi_1 \overset{\kappa_1}{=} f_1, \cdots, \xi_i \overset{\kappa_i}{=} f_i, \cdots, \xi_n \overset{\kappa_n}{=} f_n$$

Alors formule($\mathcal{S}$) $= \Gamma_{\mathcal{S}}(\xi_n)$.

**Exemple 4.2** Si $\mathcal{S}$ est le système d'équations

$$\xi_1 \overset{\nu}{=} \xi_1 \| \xi_2 \wedge \overline{\mathbf{0}}, \quad \xi_2 \overset{\mu}{=} \alpha[\xi_1] \mid \xi_2 \vee \mathbf{0}$$

alors

$$\begin{aligned}\mathsf{formule}(\mathcal{S}) &= [\xi_1 \to \nu\xi_1.\xi_1 \,\|\, \xi_2 \wedge \overline{\mathbf{0}}][\xi_2 \to \mu\xi_2.\alpha[\xi_1] \,|\, \xi_2 \vee \mathbf{0}](\xi_2)\\ &= [\xi_1 \to \nu\xi_1.\xi_1 \,\|\, \xi_2 \wedge \overline{\mathbf{0}}](\mu\xi_2.\alpha[\xi_1] \,|\, \xi_2 \vee \mathbf{0})\\ &= \mu\xi_2.\alpha[\nu\xi_1.\xi_1 \,\|\, \xi_2 \wedge \overline{\mathbf{0}}] \,|\, \xi_2 \vee \mathbf{0}\end{aligned}$$

Cette construction de $\mathsf{formule}(\mathcal{S})$ est correcte (par rapport ce qui a été annoncé au début de la section 4.2) :

**Lemme 4.12** *Pour tout système d'équations $\mathcal{S}$ sur $fonct(\tau, \mathcal{X}_r)$,*

$$Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S})) = [\![\mathsf{formule}(\mathcal{S})]\!].$$

Remarquons que ce lemme est vérifié également pour $\mathcal{S}$ étant un système d'équations sur $fixpt(\tau, \mathcal{X}_r)$. Nous l'énonçons ici sous sa forme qui sera utilisée.

*Preuve* Montrons d'abord le résultat intermédiaire suivant :

**Fait 4.1** Pour toute substitution $\Gamma$, pour tous termes $f, f'$ dans $fixpt(\tau, \mathcal{V})$ et pour toute variable $x \in \mathcal{V}$ telle que $x \notin dom(\Gamma)$, on a $(\Gamma(f))\langle x \to \Gamma(f')\rangle = (\Gamma[x \to f'])(f)$.

La preuve se fait par récurrence sur la structure du terme $f$. Pour les cas de base : si $f$ est la variable $x$, et comme $x$ n'est pas dans $dom(\Gamma)$, alors $(\Gamma(x))\langle x \to \Gamma(f')\rangle = x\langle x \to \Gamma(f')\rangle = \Gamma(f')$ et $(\Gamma[x \to f'])(x)$ est par définition égal à $\Gamma(x\langle x \to f'\rangle) = \Gamma(f')$ ; si $f$ est une des constantes $\top$, $\bot$, $\mathbf{0}$ ou $\overline{\mathbf{0}}$, alors les deux côtés de l'égalité à prouver sont égaux à $f$.

Pour les cas d'induction : si $f = \alpha[g]$, alors $(\Gamma(\alpha[g]))\langle x \to \Gamma(f')\rangle = (\alpha[\Gamma(g)])\langle x \to \Gamma(f')\rangle = \alpha[(\Gamma(g))\langle x \to \Gamma(f')\rangle]$, ce qui par hypothèse de récurrence est égal à $\alpha[(\Gamma[x \to f'])(g)]$, ce qui est égal à $(\Gamma[x \to f'])(\alpha[g]) = (\Gamma[x \to f'])(f)$. La preuve pour les autres cas est très similaire et nous la laissons au lecteur. Ceci termine la preuve du fait 4.1

Revenant à la preuve du lemme, pour tout $i$ dans $1..n$, soit $\Gamma_i$ le préfixe de $\Gamma$ de longueur $i$, càd $\Gamma_i = [\xi_1 \to \kappa_1\xi_1.f_1]\cdots[\xi_i \to \kappa_i\xi_i.f_i]$ ; dans ce cas $\Gamma_0$ est la substitution vide $\epsilon$. Supposons que $\mathcal{S}$ est le système d'équations $x_1 \overset{\kappa_1}{=} f_1, \ldots, x_n \overset{\kappa_n}{=} f_n$ et soit la suite de systèmes d'équations $(\mathcal{S}_i)_{i\in 1..n}$ sur $fixpt(\tau, \mathcal{X}_r)$ définie par : $\mathcal{S}_1 = \mathcal{S}$ et pour tout $i \in 1..n-1$, si $\mathcal{S}_i$ est égal à $x_1 \overset{\kappa_1}{=} g_1, \ldots, x_n \overset{\kappa_n}{=} g_n$, alors $\mathcal{S}_{i+1}$ est le système d'équations

$$x_1 \overset{\kappa_1}{=} g_1, \cdots, x_i \overset{\kappa_i}{=} g_i, \; x_{i+1} \overset{\kappa_{i+1}}{=} g_{i+1}\langle x_i \to \kappa_i x_i.g_i\rangle, \cdots, x_n \overset{\kappa_n}{=} g_n\langle x_i \to \kappa_i x_i.g_i\rangle.$$

Montrons d'abord que

($\star$) pour tout $i$ dans $1..n$, et pour tout $j \geq i$, la partie droite de l'équation pour $x_j$ dans $\mathcal{S}_i$ est le terme $\Gamma_{i-1}(f_j)$. La preuve se fait par récurrence sur $i$.

La propriété est vraie pour $i = 1$ puisque $\Gamma_0 = \epsilon$ et donc pour tout $j \geq 1$, $f_j = \Gamma_0(f_j)$. Soit la propriété vraie pour $i \geq 1$, nous allons montrer qu'elle est également vraie pour $i+1$. Soit $j \geq i+1$. Par définition, la partie droite de l'équation pour $x_j$ dans $\mathcal{S}_{i+1}$ est $g_j\langle x_i \to \kappa_i x_i.g_i\rangle$, où $g_j$ est la partie droite de l'équation pour $x_j$ dans $\mathcal{S}_i$. Or, par hypothèse de récurrence, $g_j = \Gamma_{i-1}(f_j)$ et $g_i = \Gamma_{i-1}(f_i)$. Donc, la partie droite de l'équation pour $x_j$ dans $\mathcal{S}_{i+1}$ est

$(\Gamma_{i-1}(f_j))\langle x_i \rightarrow \kappa_i x_i.\Gamma_{i-1}(f_i)\rangle$, ce qui est égal à $(\Gamma_{i-1}(f_j))\langle x_i \rightarrow \Gamma_{i-1}(\kappa_i x_i.f_i)\rangle$. Par le fait 4.1, ceci est égal à $(\Gamma_{i-1}[x_i \rightarrow \kappa_i x_i.f_i])(f_j)$ et donc à $\Gamma_i(f_j)$.

Maintenant, par la proposition 4.5, pour tout $i$ dans $1..n-1$, $\mathcal{S}_i$ et $\mathcal{S}_{i+1}$ ont la même solution, et donc pour tout $i$, $\mathcal{S}_i$ a la même solution que $\mathcal{S}_1 = \mathcal{S}$. Par conséquent, $Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S})) = Sol_{\mathcal{T}}(\mathcal{S}_n, \mathsf{dern}(\mathcal{S}_n))$. De $(\star)$ nous savons que la dernière équation dans $\mathcal{S}_n$ est $x_n \overset{\kappa_n}{=} \Gamma_{n-1}[f_n]$. Or, le terme $\Gamma_{n-1}(f_n)$ a comme unique variable libre $x_n$, et donc la solution de cette équation dans le système $\mathcal{S}_n$ est l'interprétation du terme (clos) $\kappa_n x_n.\Gamma_{n-1}(f_n)$, càd $[\![\kappa_n x_n.\Gamma_{n-1}(f_n)]\!]$. Nous allons montrer maintenant que le terme $\kappa_n x_n.\Gamma_{n-1}(f_n)$ est exactement $\Gamma_n(x_n)$, et donc, par définition, formule($\mathcal{S}$). En effet, $\kappa_n x_n.\Gamma_{n-1}(f_n)$ est égal à $\Gamma_{n-1}(\kappa_n x_n.f_n)$, ce qui est la même chose que $x_n\langle x_n \rightarrow \Gamma_{n-1}(\kappa_n x_n.f_n)\rangle$ et, comme $x_n$ n'est pas dans le domaine de $\Gamma_{n-1}$, c'est égal à $(\Gamma_{n-1}(x_n))\langle x_n \rightarrow \Gamma_{n-1}(\kappa_n x_n.f_n)\rangle$. Par le fait 4.1, c'est égal à $(\Gamma_{n-1}[x_n \rightarrow \kappa_n x_n.f_n])(x_n)$, c'est à dire à $\Gamma_n(x_n)$.

□

### 4.2.4 Système d'équations pour une formule

Nous montrons dans cette section comment, étant donné une formule close $\phi$, on peut construire un système d'équations sur $fonct(\tau, \mathcal{X}_r)$ tel que $Sol_{\mathcal{T}}(\mathsf{Eq}(\phi), \mathsf{dern}(\mathsf{Eq}(\phi))) = [\![\phi]\!]$ ; ce système d'équations est noté $\mathsf{Eq}(\phi)$.

Montrons d'abord que, grâce aux opérateurs logiques duaux $\bot$, $\overline{\mathbf{0}}$, $\|$, $\wedge$, et $\nu$, la négation peut être éliminée de la logique. C'est à dire, pour chaque formule close $\phi$, il existe une formule close sans négation $\mathsf{pos}(\phi)$ qui lui est équivalente. En analogie avec le $\mu$-calcul, une formule close est dite *positive* si elle est construite en utilisant uniquement les opérateurs $\mathbf{0}$, $\overline{\mathbf{0}}$, $\top$, $\alpha[\,]$, $|$, $\|$, $\vee$, $\wedge$, $\mu$ et $\nu$. [2] Soit $\phi$ une formule close. La formule $\mathsf{pos}(\phi)$ est définie récursivement sur la structure de $\phi$ comme suit :

| | | | | |
|---|---|---|---|---|
| | $\mathsf{pos}(\mathbf{0})$ | $= \mathbf{0}$ | $\mathsf{neg}(\mathbf{0})$ | $= \overline{\mathbf{0}}$ |
| | $\mathsf{pos}(\top)$ | $= \top$ | $\mathsf{neg}(\top)$ | $= \bot$ |
| | $\mathsf{pos}(\alpha[\phi])$ | $= \alpha[\phi]$ | $\mathsf{neg}(\alpha[\phi])$ | $= \mathbf{0} \vee \overline{\mathbf{0}} \mid \overline{\mathbf{0}} \vee \overline{\alpha}[\top] \vee \alpha[\mathsf{neg}(\phi)]$ |
| $(\phi \neq \xi)$ | $\mathsf{pos}(\neg\phi)$ | $= \mathsf{neg}(\phi)$ | $\mathsf{neg}(\neg\phi)$ | $= \mathsf{pos}(\phi)$ |
| | $\mathsf{pos}(\phi \mid \phi')$ | $= \mathsf{pos}(\phi) \mid \mathsf{pos}(\phi')$ | $\mathsf{neg}(\phi \mid \phi')$ | $= \mathsf{neg}(\phi) \,\|\, \mathsf{neg}(\phi')$ |
| | $\mathsf{pos}(\phi \vee \phi')$ | $= \mathsf{pos}(\phi) \vee \mathsf{pos}(\phi')$ | $\mathsf{neg}(\phi \vee \phi')$ | $= \mathsf{neg}(\phi) \wedge \mathsf{neg}(\phi')$ |
| | $\mathsf{pos}(\xi)$ | $= \xi$ | $\mathsf{neg}(\xi)$ | $= \neg\xi$ |
| | $\mathsf{pos}(\neg\xi)$ | $= \neg\xi$ | $\mathsf{neg}(\neg\xi)$ | $= \xi$ |
| | $\mathsf{pos}(\mu\xi.\phi)$ | $= \mu\xi.\mathsf{pos}(\phi)$ | $\mathsf{neg}(\mu\xi.\phi)$ | $= \nu\xi.\mathsf{neg}(\phi\langle\xi \rightarrow \neg\xi\rangle)$ |

**Lemme 4.13** *Pour toute formule close $\phi$ sur la syntaxe d'origine, $\mathsf{pos}(\phi)$ est une formule positive et équivalente à $\phi$.*

Avant de donner la preuve de ce lemme, nous faisons dans le lemme 4.15 une caractérisation de la formule $\mathsf{pos}(\phi)$ et de la relation qui existe entre $\mathsf{pos}(\phi)$ et $\phi$. Cette caractérisation est utilisée pour la preuve du lemme 4.13 mais également plus tard dans la thèse.

[2] La notion de formule positive existe également pour les formule non closes. Dans ce cas l'opérateur de négation est autorisé uniquement immédiatement devant une variable libre.

**Définition 4.14 (Chemin)** Un *chemin* est une séquence non vide d'opérateurs de la signature $\tau$ enrichis avec un rang pour les opérateurs binaires : càd un chemin est une suite d'éléments de

$$\{\neg, (\alpha)_{\alpha \subseteq \Sigma}, \mu\xi, \nu\xi, \vee_1, \vee_2, \wedge_1, \wedge_2, |_1, |_2, \|_1, \|_2\}$$

et qui se termine éventuellement par un opérateur constant parmi

$$\{\mathbf{0}, \overline{\mathbf{0}}, \top, \bot, \xi\}.$$

On utilise le point (« . ») pour séparer les éléments d'un chemin et pour la concaténation de chemins.

Chaque formule $\phi$ définit un ensemble de chemins qui sont les chemins dans l'arbre de dérivation de $\phi$. Pour les opérateurs binaires, l'indice indique si l'élément suivant dans le chemin est le premier ou le second opérande de cet opérateur. Par *chemin dans* $\phi$ nous entendons un chemin qui commence par la racine de $\phi$. D'autre part, chaque chemin $c$ dans une formule $\phi$ définit un nœud dans l'arbre de dérivation de $\phi$ et donc une sous formule de $\phi$. Nous notons par $\phi(c)$ la sous formule de $\phi$ dont la racine est le nœud de $\phi$ défini par le chemin $c$. Par exemple, si $\phi_1$ est la formule $\mu\xi.\alpha[\mathbf{0}] \parallel \xi \vee ((\neg\beta[\neg\xi] \parallel \bot) \wedge \xi')$ et $c$ le chemin $\mu\xi.\ \vee_2\ .\ \wedge_1\ .\ \|_1\ .\neg.\beta$, alors $\phi_1(c) = \beta[\neg\xi]$. Si $\phi_2$ est la formule $\alpha[\mathbf{0}] \mid \alpha[\top]$, alors $\phi_2(|_1\ .\alpha)$ est la formule $\alpha[\mathbf{0}]$ et $\phi_2(|_2\ .\alpha)$ est la formule $\alpha[\top]$. Ce deuxième exemple illustre la nécessité d'utiliser des indices pour les opérateurs binaires pour garantir que tout chemin dans $\phi$ définit de façon unique une sous formule de $\phi$.

Dans la suite, les chemins seront utilisés pour deux choses :
- Pouvoir désigner précisément une occurrence de sous formule ; par exemple dans la formule $\phi_1$ ci-dessus, les deux occurrences de $\xi$ peuvent être distinguées par le chemin qui les définit.
- Préciser le contexte dans lequel une sous formule apparaît ; par exemple dans la formule $\phi_1$ ci-dessus, un chemin peut nous indiquer si la formule $\xi$ apparaît sous un opérateur de négation ou non.

Pour raccourcir l'écriture, lorsqu'un chemin est utilisé pour définir un contexte, nous allons omettre les indices des opérateurs binaires (càd $\vee$ à la place de $\vee_1$ et $\vee_2$, etc.). Cependant, même si les indices sont omis, nous admettons que tout chemin dans $\phi$ définit de façon unique une occurrence de sous formule de $\phi$.

Les chemins des formules de la syntaxe d'origine sont constitués des opérateurs $\mathbf{0}$, $\top$, $\xi$, $\neg$, $\vee_i$, $|_i$ (pour $i$ étant 1 ou 2), les opérateurs $\alpha$ pour n'importe quel ensemble fini ou co-fini d'étiquettes $\alpha$ et des opérateurs de plus petit point fixe de la forme $\mu\xi$ (pour n'importe quelle variable de récursion $\xi$). Remarquons que, d'après la définition ci-dessus, les chemins dans la formule $\mathsf{pos}(\phi)$ ou $\mathsf{neg}(\phi)$ peuvent contenir aussi l'opérateur $\neg$, ceci à cause des règles $\mathsf{pos}(\neg\xi)$ et $\mathsf{neg}(\xi)$. Comme le lemme 4.13 n'est pas prouvé, nous supposons que c'est effectivement possible.

Dans le but de pouvoir comparer les chemins dans une formule $\phi$ avec les chemins des formules $\mathsf{pos}(\phi)$ et $\mathsf{neg}(\phi)$, nous étendons les opérations pos() et neg() sur les chemins comme

suit (le point . désigne l'opérateur de concaténation de chemins) :

$$\begin{array}{llll}
\mathsf{pos}(\mathbf{0}) & = \mathbf{0} & \mathsf{neg}(\mathbf{0}) & = \overline{\mathbf{0}} \\
\mathsf{pos}(\top) & = \top & \mathsf{neg}(\top) & = \bot \\
\mathsf{pos}(\xi) & = \xi & \mathsf{neg}(\xi) & = \neg\xi \\
\mathsf{pos}(\alpha.c) & = \alpha.\mathsf{pos}(c) & \mathsf{neg}(\alpha.c) & = \vee.\alpha.\mathsf{neg}(c) \\
\mathsf{pos}(\neg.c) & = \mathsf{neg}(c) & \mathsf{neg}(\neg.c) & = \mathsf{pos}(c) \\
\mathsf{pos}(|.c) & = |.\mathsf{pos}(c) & \mathsf{neg}(|.c) & = \|.\mathsf{neg}(c) \\
\mathsf{pos}(\vee.c) & = \vee.\mathsf{pos}(c) & \mathsf{neg}(\vee.c) & = \wedge.\mathsf{neg}(c) \\
\mathsf{pos}(\mu\xi.c) & = \mu\xi.\mathsf{pos}(c) & \mathsf{neg}(\mu\xi.c) & = \nu\xi.\mathsf{neg}(c\langle\xi \to \neg.\xi\rangle)
\end{array}$$

Ici, $c\langle\xi \to \neg.\xi\rangle$ désigne le chemin obtenu en remplaçant dans $c$ l'occurrence (éventuelle) de $\xi$ par le chemin $\neg.\xi$.

Soit $c$ un chemin et $X = \{\xi_1, \ldots, \xi_n\}$ un ensemble de variables de récursion. Nous notons par $c\langle X \to \neg X\rangle$ le chemin $c\langle\xi_1 \to \neg.\xi_1\rangle \cdots \langle\xi_n \to \neg.\xi_n\rangle$. Remarquons que le chemin $c$ ne peut contenir qu'une variable de récursion, en dernière position.

**Lemme 4.15** *Soit une formule close $\phi$.*

*(i) Pour tout chemin $c$ dans $\phi$, $\mathsf{pos}(c)$ est un chemin dans $\mathsf{pos}(\phi)$.*

*(ii) Pour tout chemin $c'.\xi$ dans $\mathsf{pos}(\phi)$, il existe un chemin $c$ dans $\phi$ tel que $c'.\xi = \mathsf{pos}(c)$ ;*

*(iii) Soit $c_1.c_2$ un chemin dans $\phi$ et soit $X$ l'ensemble de variables de récursion $\xi$ telles que $\mu\xi$ est un élément de $c_1$ et $\mu\xi$ apparaît dans $c_1$ sous un nombre impair de négations. Avec ces notations,*

*– si $c_1$ contient un nombre pair de négations, alors*

$$\mathsf{pos}(c_1.c_2) = \mathsf{pos}(c_1).\mathsf{pos}(c_2\langle X \to \neg X\rangle)$$

*et*

*– si $c_1$ contient un nombre impair de négations alors*

$$\mathsf{pos}(c_1.c_2) = \mathsf{pos}(c_1).\mathsf{neg}(c_2\langle X \to \neg X\rangle).$$

Remarquons que la deuxième propriété énoncée dans ce lemme ne peut être généralisée dans le style de la première : càd il n'est pas vrai que pour tout chemin $c'$ dans $\mathsf{pos}(\phi)$, il existe un chemin $c$ dans $\phi$ tel que $c' = \mathsf{pos}(c)$, ceci à cause de la règle $\mathsf{neg}(\alpha[\phi])$ qui génère les chemins $\vee.\mathbf{0}$, $\vee.|.\overline{\mathbf{0}}$ et $\vee.\overline{\alpha}.\top$ en plus des chemins $\vee.\alpha.\mathsf{neg}(c)$ pour $c$ étant un chemin dans $\phi$. C'est pourquoi nous limitons ici cette propriété aux chemins se terminant par une variable de récursion, puisque dans la suite la propriété sera appliquée uniquement à ce type de chemins.

*Preuve* La preuve de ce lemme est une récurrence fastidieuse mais facile sur la structure de la formule $\phi$.

□

Nous pouvons maintenant donner la preuve du lemme 4.13 en utilisant le lemme 4.15.

*Preuve du lemme 4.13* Pour montrer que la formule $\phi$ est équivalente à la formule $\mathsf{pos}(\phi)$, nous montrons par récurrence sur la structure de $\phi$ que pour toute valuation des variables $\delta$,

$[\![\mathsf{pos}(\phi)]\!]_\delta = [\![\phi]\!]_\delta$ et $[\![\mathsf{neg}(\phi)]\!] = \complement[\![\phi]\!]_\delta$. Cette preuve se fait très facilement en utilisant les propriétés de l'interprétation des formules.

Montrons maintenant que la construction de $\mathsf{pos}(\phi)$ et $\mathsf{neg}(\phi)$ termine. Remarquons d'abord que dans la plupart des règles définissant $\mathsf{pos}(\phi)$ et $\mathsf{neg}(\phi)$, une récursion $\mathsf{pos}(\phi')$ ou $\mathsf{neg}(\phi')$ apparaissant en partie droite de ces règles est telle que la formule $\phi'$ est une sous formule stricte de $\phi$. Les seule règle qui ne satisfait pas cette propriété est la règle

$$\mathsf{neg}(\mu\xi.\phi) = \nu\xi.\mathsf{neg}(\phi\langle\xi \to \neg\xi\rangle).$$

Nous allons donc définir une relation d'ordre bien fondée sur les formules qui est basée sur la relation de sous formule, mais ne prend pas en compte le nombre de négations qui peuvent se trouver immédiatement devant une variable de récursion. Pour toute formule $\phi$ de la logique spatiale, soit sa formule « témoin » $\mathsf{tem}(\phi)$ définie par :

$$\mathsf{tem}(\phi) = \begin{cases} \phi & \text{si } \phi = \mathbf{0} \text{ ou } \phi = \top \\ \xi & \text{si } \phi \text{ est de la forme } \underbrace{\neg\cdots\neg}_{n \text{ fois}}\xi \text{ pour } n \in \mathbb{N} \\ \alpha[\mathsf{tem}(\phi')] & \text{si } \phi = \alpha[\phi'] \\ \neg\mathsf{tem}(\phi') & \text{si } \phi = \neg\phi' \text{ et } \phi \text{ n'est pas de la forme } \underbrace{\neg\cdots\neg}_{n \text{ fois}}\xi \text{ pour } n \in \mathbb{N} \\ \mu\xi.\mathsf{tem}(\phi') & \text{si } \phi = \mu\xi.\phi' \\ \mathsf{tem}(\phi') \circ \mathsf{tem}(\phi'') & \text{si } \phi = \phi' \circ \phi'' \text{ et } \circ \text{ est un des opérateurs } \vee \text{ ou } | \end{cases}$$

Nous définissons alors la relation d'ordre $<$ entre formules : $\phi < \phi'$ si $\mathsf{tem}(\phi)$ est sous-formule stricte de $\mathsf{tem}(\phi')$. Il est facile de voir que cette relation définit un ordre bien fondé. Notons par $<_{\mathrm{sf}}$ l'ordre entre formules défini par la relation de sous formule stricte, et considérons l'ordre lexicographique entre formules $(<, <_{\mathrm{sf}})$. On peut voir que chacune des règles définissant $\mathsf{pos}(\phi)$ et $\mathsf{neg}(\phi)$ qui est définie de manière récursive, est définie en fonction de formules $\phi'$ inférieures pour l'ordre $(<, <_{\mathrm{sf}})$. Plus particulièrement, toutes les règles font décroître la première composante de cet ordre sauf les règles $\mathsf{pos}(\neg\phi) = \mathsf{neg}(\phi)$ et $\mathsf{neg}(\neg\phi) = \mathsf{pos}(\phi)$ lorsque $\phi$ est une formule de la forme $\neg\cdots\neg\xi$, dans quel cas l'appel récursif se fait sur des formules égales pour l'ordre $<$. (En effet, $\phi <_{\mathrm{sf}} \phi'$ implique $\phi < \phi'$ sauf dans le cas où $\phi$ et de la forme $\neg\cdots\neg\phi'$.) Dans le cas de ces deux règles, nous avons bien $\phi <_{\mathrm{sf}} \phi'$.

Il reste à montrer que $\mathsf{pos}(\phi)$ est effectivement une formule positive. D'après les règles de construction de la formule $\mathsf{pos}(\phi)$, on peut voir qu'un opérateur de négation peut apparaître dans $\mathsf{pos}(\phi)$ uniquement devant une variable de récursion $\xi$. Donc, la formule $\mathsf{pos}(\phi)$ contient une négation seulement si elle contient une négation sur un chemin se terminant par $\xi$. Dans le lemme 4.15 point (ii), nous avons vu que si $d.\xi$ est un chemin dans $\mathsf{pos}(\xi)$, alors il existe un chemin $c'$ dans $\phi$ tel que $d.\xi = \mathsf{pos}(c')$. D'après les règles de construction de $\mathsf{pos}(c')$, nous pouvons voir que nécessairement $c'$ est de la forme $c.\xi$. Donc, $\mathsf{pos}(\phi)$ contient une négation seulement s'il existe un chemin $c.\xi$ dans $\phi$ tel que $\mathsf{pos}(c.\xi)$ contient une négation. Nous allons monter que pour tout chemin de la forme $c.\xi$ dans $\phi$, $\mathsf{pos}(c.\xi)$ ne contient pas de négation. D'abord, puisque la formule $\phi$ est close et correcte, nous savons que $c.\xi = c_1.\mu\xi.c_2.\xi$ où $c_1$ et $c_2$ sont des chemins et le chemin $c_2$ contient un nombre pair de négations. Maintenant, d'après le lemme 4.15 point (iii), soit $X$ l'ensemble des variables de récursion $\xi'$ telles que $\mu\xi'$ apparaît dans $c_1.\mu\xi.c_2$ sous un nombre impair de négations. Alors, de deux choses l'une (rappelant que $c_2$ contient un nombre pair de négations) :

(i) $c_1$ contient un nombre pair de négations et $\mathsf{pos}(c_1.\mu\xi.c_2.\xi) = \mathsf{pos}(c_1.\mu\xi.c_2).\mathsf{pos}(\xi\langle X \to \neg X\rangle)$ ou

(ii) $c_1$ contient un nombre impair de négations et $\mathsf{pos}(c_1.\mu\xi.c_2.\xi) = \mathsf{pos}(c_1.\mu\xi.c_2).\mathsf{neg}(\xi\langle X \to \neg X\rangle)$.

Dans le premier cas, $\xi$ n'appartient pas à $X$ et donc $\mathsf{pos}(c_1.\mu\xi.c_2.\xi) = \mathsf{pos}(c_1.\mu\xi.c_2).\xi$, qui est un chemin ne contenant pas de négations. Dans le second cas, $\xi$ appartient à $X$ et donc $\mathsf{pos}(c_1.\mu\xi.c_2.\xi) = \mathsf{pos}(c_1.\mu\xi.c_2).\mathsf{neg}(\neg.\xi)$, ce qui est égal à $\mathsf{pos}(c_1.\mu\xi.c_2).\xi$ et est un chemin sans négation.

□

**Corollaire 4.16** *Pour toute formule close $\phi$ de la logique $LS_{|\exists}$, il existe une formule positive, notée* $\mathsf{pos}(\phi)$*, telle que* $[\![\phi]\!] = [\![\mathsf{pos}(\phi)]\!]$.

Suite à ce corollaire, à partir de maintenant jusque la fin de cette partie de la thèse et sauf mention du contraire, quand nous parlons de formule logique nous entendrons une formule positive. C'est à dire, à partir de maintenant nous identifions formules logiques et termes sur $\mathit{fixpt}(\tau, \mathcal{X}_\mathrm{r})$ et formules logiques sans récursion et termes sur $\mathit{fonct}(\tau, \mathcal{X}_\mathrm{r})$.

Soit maintenant $f$ une formule positive sans récursion close (ou, de façon équivalente, un terme clos dans $\mathit{fonct}(\tau, \mathcal{X}_\mathrm{r})$). Sans perte de généralité, nous supposons que $f$ est de la forme $\kappa\xi.f'$. A partir de cette formule nous allons construire le système d'équations sur $\mathit{fonct}(\tau, \mathcal{X}_\mathrm{r})$, noté $\mathsf{Eq}(f)$, tel que $\mathit{Sol}_\mathcal{T}(\mathsf{Eq}(f), \mathsf{dern}(\mathsf{Eq}(f))) = [\![f]\!]$. Pour chaque formule $g$, nous notons par $\mathsf{tr}(g)$ la formule obtenue en remplaçants dans $g$ chaque sous formule récursive de la forme $\kappa\xi.g'$ par la variable $\xi$. La définition de $\mathsf{tr}(g)$ récursivement sur la structure de $g$ est donnée sur la figure 2.

| $g$ | $\mathsf{tr}(g)$ |
|---|---|
| $\mathbf{0}, \overline{\mathbf{0}}, \bot, \top, \xi$ | $g$ |
| $\alpha[g']$ | $\alpha[\mathsf{tr}(g')]$ |
| $g' \circ g''$ (pour $\circ \in \{\mid, \|, \vee, \wedge\}$) | $\mathsf{tr}(g') \circ \mathsf{tr}(g'')$ |
| $\mu\xi.g$ | $\xi$ |

FIG. 2 – Définition de la formule $\mathsf{tr}(g)$.

Alors le système $\mathsf{Eq}(f)$ est défini récursivement sur la structure de $f$ comme suit :

$$\mathsf{Eq}(f) = \begin{cases} \mathsf{Eq}(f'), \xi' \overset{\kappa}{=} \mathsf{tr}(f') & \text{si } f = \kappa\xi'.f' \\ \mathsf{Eq}(f'), \mathsf{Eq}(f'') & \text{si } f = f' \circ f'' \text{ pour } \circ \in \{\vee, \wedge, \mid, \|\} \\ \mathsf{Eq}(f') & \text{si } f = \alpha[f'] \\ \epsilon & \text{sinon} \end{cases}$$

où $\epsilon$ désigne la séquence (d'équations) vide et , est la composition séquentielle de deux systèmes d'équations.

L'exemple ci-dessous illustre la construction de $\mathsf{Eq}(f)$.

**Exemple 4.3** Soit la formule $f'' = \mu\xi'.\alpha[\nu\xi.\xi \parallel \xi' \wedge \overline{\mathbf{0}}] \mid \xi' \vee \mathbf{0}$. En posant $f = \xi \parallel \xi' \wedge \overline{\mathbf{0}}$ et $f' = \alpha[\nu\xi.f] \mid \xi' \vee \mathbf{0}$, nous avons $f'' = \mu\xi'.f'$. Notons d'abord que $\mathsf{tr}(f') = \alpha[\xi] \mid \xi' \vee \mathbf{0}$ et $\mathsf{tr}(f) = f$.

Alors $\mathsf{Eq}(f'')$ est le système $\mathsf{Eq}(\mu\xi'.f') = \mathsf{Eq}(f'), \xi' \stackrel{\mu}{=} \alpha[\xi] \mid \xi' \vee \mathbf{0}$. Maintenant $\mathsf{Eq}(f')$ est égal à $\mathsf{Eq}(\alpha[\nu\xi.f] \mid \xi' \vee \mathbf{0})$ et se réduit en quelques pas en $\mathsf{Eq}(f)$ et $\mathsf{Eq}(f)$ se réduit en $\xi \stackrel{\nu}{=} \xi \parallel \xi' \wedge \overline{\mathbf{0}}$. Donc, le système d'équations associé à la formule $f''$ est

$$\xi \stackrel{\nu}{=} \xi \parallel \xi' \wedge \overline{\mathbf{0}}, \quad \xi' \stackrel{\mu}{=} \alpha[\xi] \mid \xi' \vee \mathbf{0}$$

Pour montrer que cette méthode de construction de $\mathsf{Eq}(\phi)$ est correcte, nous utilisons la propriété suivante :

**Lemme 4.17** *Pour toute formule close $f$,* $\mathsf{formule}(\mathsf{Eq}(f)) = f$.

*Preuve* Montrons d'abord que pour toute formule $f$, $\Gamma_{\mathsf{Eq}(f)}(\mathsf{tr}(f)) = f$. La preuve se fait par récurrence sur la structure de la formule $f$.

Pour le cas de base, $f$ est l'une des formules $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ ou $x$, alors par définition $\mathsf{Eq}(f)$ est le système à zéro équations, $\Gamma_{\mathsf{Eq}(f)}$ est la substitution vide et $\mathsf{tr}(f) = f$, d'où la propriété suit immédiatement.

Pour l'induction : si $f = \alpha[f']$, alors par définition
- $\mathsf{Eq}(f) = \mathsf{Eq}(f')$,
- $\Gamma_{\mathsf{Eq}(f)} = \Gamma_{\mathsf{Eq}(f')}$ et
- $\mathsf{tr}(f) = \alpha[\mathsf{tr}(f')]$.

Maintenant, $\Gamma_{\mathsf{Eq}(f')}(\alpha[\mathsf{tr}(f')]) = \alpha[\Gamma_{\mathsf{Eq}(f')}(\mathsf{tr}(f'))]$, ce qui par hypothèse de récurrence est égal à $\alpha[f']$, et donc à $f$. Si $f = f' \circ f''$, où $\circ$ est l'un des symboles $\vee, \wedge, \mid, \parallel$, la preuve est similaire au cas précédent. Finalement, si $f = \kappa\xi.f'$, alors par définition
- $\mathsf{Eq}(f)$ est le système $\mathsf{Eq}(f'), \xi \stackrel{\kappa}{=} \mathsf{tr}(f')$,
- $\Gamma_{\mathsf{Eq}(f)} = \Gamma_{\mathsf{Eq}(f')}[\xi \rightarrow \kappa\xi.\mathsf{tr}(f')]$ et
- $\mathsf{tr}(f) = \xi$.

Donc $\Gamma_{\mathsf{Eq}(f)}(\mathsf{tr}(f)) = \Gamma_{\mathsf{Eq}(f')}[\xi \rightarrow \kappa\xi.\mathsf{tr}(f')](\xi) = \Gamma_{\mathsf{Eq}(f')}(\kappa\xi.\mathsf{tr}(f'))$, et cette dernière expression est égale à $\kappa\xi.\Gamma_{\mathsf{Eq}(f')}(\mathsf{tr}(f'))$ et par hypothèse de récurrence est égal à $\kappa\xi.f'$, et donc à $f$.

Maintenant, comme nous avons supposé sans perte de généralité que $f$ est une formule de la forme $\kappa\xi.f'$, par définition $\mathsf{Eq}(f) = \mathsf{Eq}(f'), \xi \stackrel{\kappa}{=} \mathsf{tr}(f')$ et donc $\mathsf{formule}(\mathsf{Eq}(f)) = \Gamma_{\mathsf{Eq}(f)}(\xi)$. Or, par définition, $\xi = \mathsf{tr}(f)$ et donc par la propriété précédente $\Gamma_{\mathsf{Eq}(f)}(\xi) = f$, ce qui termine la preuve du lemme.

□

**Lemme 4.18** *Pour toute formule close $f$,* $Sol_{\mathcal{T}}(\mathsf{Eq}(f), \mathsf{dern}(\mathsf{Eq}(f))) = [\![f]\!]$.

*Preuve* Par le lemme 4.12, nous avons $Sol_{\mathcal{T}}(\mathsf{Eq}(f), \mathsf{dern}(\mathsf{Eq}(f))) = [\![\mathsf{formule}(\mathsf{Eq}(f))]\!]$ et par le lemme 4.17 nous avons $\mathsf{formule}(\mathsf{Eq}(f)) = f$.

□

## 4.3 Formes normales des systèmes d'équations

Jusque la fin de ce chapitre, et sauf mention du contraire, par « système d'équations » nous entendons un système d'équations sur $fonct(\tau, \mathcal{X}_\mathrm{r})$.

Les résultats présentés dans cette section sont essentiellement des résultats techniques qui seront utiles dans la suite. Nous présentons deux formes normales d'un système d'équations. La première est appelée « système simplifié » et est obtenue par des transformations syntaxiques. Les systèmes simplifiés sont introduits pour restreindre le type de termes qui peuvent apparaître en partie droite d'équation à des termes constants $(\mathbf{0}, \overline{\mathbf{0}}, \top, \bot)$, des termes de la forme $\alpha[\xi]$ ou de la forme $f \circ f'$, où $\circ$ est un opérateur binaire. Les systèmes simplifiés seront utilisés pour définir un automate à contraintes numériques correspondant à un système d'équations, et donc à une formule $\mathrm{LS}_{|\exists}$. La deuxième forme normale est appelée « système normalisé » et est un système sur $fonct(\tau', \mathcal{X}_\mathrm{r})$, où $\tau'$ est une extension de la signature $\tau$ avec deux nouveaux symboles de composition. Un système normalisé est obtenu par des transformations syntaxiques, mais en utilisant également des informations d'ordre sémantique sur les solutions du système d'équations.

La propriété principale d'un système d'équations simplifié et d'un système d'équations normalisés obtenus à partir d'un système quelconque est la préservation des solutions : la transformation se faisant en ajoutant de nouvelles variables, nous voulons garantir qu'après transformation, la solution du système reste la même pour les variables d'origine. Pour cela nous introduisons la notion d'équivalence de système d'équations.

**Définition 4.19 (Systèmes d'équations équivalents)** Soit $\sigma$ une signature et $\mathcal{I}$ une $\mu$-interprétation des symboles de $\sigma$. Les systèmes d'équations $\mathcal{S}$ et $\mathcal{S}'$ sur $fonct(\sigma, \mathcal{X}_\mathrm{r})$ sont dits *équivalents pour l'ensemble de variables* $\mathcal{V}$ si $\mathcal{V} \subseteq \{\mathsf{Vars}(\mathcal{S})\}$, $\mathcal{V} \subseteq \{\mathsf{Vars}(\mathcal{S}')\}$ et pour toute variable $\xi$ dans $\mathcal{V}$, $Sol_\mathcal{I}(\mathcal{S}, \xi) = Sol_\mathcal{I}(\mathcal{S}', \xi)$

Le chapitre est organisé comme suit : dans la section 4.3.1, nous définissons ce qu'est un système d'équations simplifié et nous montrons comment un système d'équations sur peut être transformé en un système simplifié équivalent. Dans la section 4.3.2 nous présentons la signature $\tau'$ et les systèmes d'équations normalisés. Nous présentons également une méthode de construction d'un système normalisé équivalent à partir d'un système quelconque. Finalement dans la section 4.4 nous utilisons les systèmes d'équations normalisés pour déduire certaines conditions pour l'unicité du plus petit et plus grand point fixe de la logique $\mathrm{LS}_{|\exists}$.

### 4.3.1 Système d'équations simplifié

Le système d'équations simplifié est la première forme normale d'un système d'équations que nous introduisons ici. L'utilité d'un système d'équations simplifié est de n'autoriser que certains types de termes en partie droite d'équation. Comme nous le verrons plus tard (dans le chapitre 6), cette restriction est nécessaire pour pouvoir définir l'automate à contraintes numériques reconnaissant le langage $\mathsf{formule}(\mathcal{S})$, où $\mathcal{S}$ est un système d'équations.

Ainsi, un système d'équations simplifié est défini de la façon suivante :

**Définition 4.20 (Terme simple et système simplifié)** Un terme fonctionnel sur $\tau$ est dit *terme simple* s'il est de l'une des formes suivantes

$$\mathbf{0} \quad \overline{\mathbf{0}} \quad \top \quad \bot \quad \alpha[\xi] \quad \xi \,|\, \xi' \quad \xi \,\|\, \xi' \quad \xi \vee \xi' \quad \xi \wedge \xi'$$

où $\xi$ et $\xi'$ sont des variables.

Un système d'équations $\mathcal{S}$ sur $fonct(\tau, \mathcal{X}_r)$ est dit *simplifié* si toute partie droite d'équation dans $\mathcal{S}$ est un terme simple.

Dans la suite de la section nous montrons comment un système d'équations $\mathcal{S}$ sur $fonct(\tau, \mathcal{X}_r)$ peut être transformé en un système simplifié équivalent à $\mathcal{S}$ sur l'ensemble des variables $\mathsf{Vars}(\mathcal{S})$. La méthode de transformation utilise le système de réécriture de systèmes d'équations présenté sur la figure 3 ci-dessous.

| | | $l$ | $r$ |
|---|---|---|---|
| (Red $\alpha$) | | $\xi \overset{\kappa}{=} \alpha[f]$ | $\xi' \overset{\kappa}{=} f,\ \xi \overset{\kappa}{=} \alpha[\xi']$ |
| (Red $\circ$ 1) | $\circ = \|, \|\|, \vee, \wedge$ | $\xi \overset{\kappa}{=} f \circ \xi''$ | $\xi' \overset{\kappa}{=} f,\ \xi \overset{\kappa}{=} \xi' \circ \xi''$ |
| (Red $\circ$ 2) | $\circ = \|, \|\|, \vee, \wedge$ | $\xi \overset{\kappa}{=} \xi'' \circ f$ | $\xi' \overset{\kappa}{=} f,\ \xi \overset{\kappa}{=} \xi'' \circ \xi'$ |
| (Red $\circ$ 3) | $\circ = \|, \|\|, \vee, \wedge$ | $\xi \overset{\kappa}{=} f \circ f'$ | $\xi' \overset{\kappa}{=} f,\ \xi'' \overset{\kappa}{=} f',\ \xi \overset{\kappa}{=} \xi' \circ \xi''$ |

FIG. 3 – Ensemble de règles de réécriture pour la simplification d'un système d'équations. Ici $\xi, \xi', \xi''$ sont des variables et $f', f', f''$ sont des termes qui ne sont pas des variables. Nous imposons également que si le système d'équations $\mathcal{S}, l, \mathcal{S}'$ se réécrit en le système d'équations $\mathcal{S}, r, \mathcal{S}'$, pour n'importe quel des couples $l, r$, alors les variables $\xi'$ et $\xi''$ apparaissant dans $r$ n'appartiennent pas à $\{\mathsf{Vars}(\mathcal{S})\} \cup \{\mathsf{Vars}(\mathcal{S}')\}$.

**Lemme 4.21** *La relation de réécriture $\rightarrow_s$ termine. De plus, si $\mathcal{S}_t$ est une forme normale de $\mathcal{S}_i$ pour $\rightarrow_s$, alors*

*(i) $\mathcal{S}_t$ est simplifié ;*

*(ii) $\mathcal{S}_t$ est équivalent à $\mathcal{S}_i$ sur $\{\mathsf{Vars}(\mathcal{S}_i)\}$ ;*

*Preuve* Voir la section A.2 de l'annexe A.

□

**Exemple 4.4** Comme nous l'avons vu dans l'exemple 4.3, le système d'équations associé à la formule $\mu\xi'.\alpha[\nu\xi.\xi \,\|\, \xi' \wedge \overline{\mathbf{0}}] \,|\, \xi' \vee \mathbf{0}$ est

$$\xi \overset{\nu}{=} \xi \,\|\, \xi' \wedge \overline{\mathbf{0}}, \quad \xi' \overset{\mu}{=} \alpha[\xi] \,|\, \xi' \vee \mathbf{0}$$

Avec la procédure de simplification présentée ici, nous obtenons le système d'équations simplifié suivant

$$\xi_1 \overset{\nu}{=} \xi \,\|\, \xi', \qquad \xi_2 \overset{\nu}{=} \overline{\mathbf{0}}, \qquad \xi \overset{\nu}{=} \xi_1 \wedge \xi_2,$$

$$\xi'_3 \overset{\mu}{=} \alpha[\xi], \qquad \xi'_1 \overset{\mu}{=} \xi'_3 \,|\, \xi', \qquad \xi'_2 \overset{\mu}{=} \mathbf{0}, \qquad \xi' \overset{\mu}{=} \xi'_1 \vee \xi'_2$$

### 4.3.2 Système d'équations normalisé

Soit la signature (infinie) $\tau' = \{\mathbf{0}, \overline{\mathbf{0}}, \top, \bot, \vee, \wedge, |, \|, \lceil, \rceil\lceil\} \cup \bigcup_{\alpha \subseteq \Sigma \text{ fini ou cofini}} \{\alpha\}$ avec les mêmes conventions que pour la signature $\tau$ définie dans la section 4.2.2 (page 72). Nous allons montrer que tout système d'équations $\mathcal{S}$ sur $fonct(\tau, \mathcal{X}_r)$ peut être transformé en un système d'équations $\mathcal{S}'$ sur $fonct(\tau', \mathcal{X}_r)$ n'utilisant pas les symboles $|$ et $\|$ et équivalent à $\mathcal{S}$ (pour une interprétation des symboles $\lceil$ et $\rceil\lceil$ donnée ci-dessous).

Nous considérons la $\mu$-interprétation $\mathcal{T}$ pour la signature $\tau'$ dans le treillis $\langle \wp(\mathcal{A}_\Sigma), \vee, \wedge \rangle$. Pour tous les symboles à l'exception de $\lceil$ et $\rceil\lceil$ cette interprétation coïncide avec l'interprétation $\mathcal{T}$ pour la signature $\tau$ présentée sur la figure 1 page 73. Pour les symboles $\lceil$ et $\rceil\lceil$, $\mathcal{T}$ est définie par

$$S \lceil^{\mathcal{T}} S' = \{t \uplus t' \mid t \in S \setminus \{\{\!|\}\!\}, t' \in S \setminus \{\{\!|\}\!\}\}$$
$$S \rceil\lceil^{\mathcal{T}} S' = \{t'' \mid \forall t, t'.\ t'' = t \uplus t' \Rightarrow (t \in S \cup \{\{\!|\}\!\} \text{ ou } t' \in S' \cup \{\{\!|\}\!\})\}$$

où $S, S'$ sont des ensembles d'arbres. Il est facile de voir que $\mathcal{T}$ est effectivement une $\mu$-interprétation.

**Remarque 4.22** *Comme conséquence immédiate des définitions, pour tous ensembles d'arbres $S, S'$, on a*

$$S \lceil^{\mathcal{T}} S' = (S \setminus \{\{\!|\}\!\}) \,|\, (S' \setminus \{\{\!|\}\!\})$$
$$S \rceil\lceil^{\mathcal{T}} S' = (S \cup \{\{\!|\}\!\}) \,\|\, (S' \cup \{\{\!|\}\!\})$$

*De plus, les deux opérateurs $\lceil$ et $\rceil\lceil$ sont duaux dans le sens où pour tous ensembles d'arbres $S, S'$,*

$$S \rceil\lceil^{\mathcal{T}} S' = \complement(\complement S \lceil^{\mathcal{T}} \complement S')$$

*où $\complement$ désigne le complémentaire dans l'ensemble $\mathcal{A}_\Sigma$.*

Pour un système d'équations $\mathcal{S}$ sur $fonct(\tau', \mathcal{X}_r)$, nous noterons $Sol_{\mathcal{T}}(\mathcal{S})$ sa solution obtenue par l'interprétation $\mathcal{T}$. Il est évident que tout système d'équations sur $fonct(\tau, \mathcal{X}_r)$ est également un système sur $fonct(\tau', \mathcal{X}_r)$.

**Définition 4.23 (Composition gardée)** Un système d'équations $\mathcal{S}$ sur $fonct(\tau', \mathcal{X}_r)$ est dit *à composition gardée* si aucune partie droite d'équation dans $\mathcal{S}$ n'utilise les symboles $|$ et $\|$.

Un terme de la forme $f \circ f'$ pour $f$ étant un des symboles $|$, $\|$, $\lceil$ ou $\rceil\lceil$ sera appelée *terme à composition*. Si $\circ$ est $|$ ou $\|$, nous dirons que c'est une *terme à composition non gardée*, et si $\circ$ est $\lceil$ ou $\rceil\lceil$, nous parlerons de *terme à composition gardée*.

Un *terme booléen* est un terme construit en utilisant uniquement des variables et les symboles de fonction $\top$, $\bot$, $\vee$ et $\wedge$. On dit que la variable $\xi$ apparaît en *position booléenne* dans le terme fonctionnel $f$ s'il existe un chemin $c$ dans $f$ tel que $f(c) = \xi$ et $c$ est composé des opérateurs $\vee$ et $\wedge$ uniquement (en plus de $\xi$ qui est nécessairement le dernier élément de $c$).

**Définition 4.24 (Dépendance booléenne)** Soit $\mathcal{S}$ un système d'équations simplifié. Soit la relation binaire $\mathcal{B}_{\mathcal{S}}(,)$ entre les variables de $\mathsf{Vars}(\mathcal{S})$ définie par : $\mathcal{B}_{\mathcal{S}}(\xi, \xi')$ si la partie droite de

l'équation pour $\xi$ dans $\mathcal{S}$ est un terme booléen et la variable $\xi'$ apparaît dans ce terme. Nous notons par $\mathcal{B}^*_{\mathcal{S}}(,)$ la clôture transitive de $\mathcal{B}_{\mathcal{S}}(,)$. Pour deux variables $\xi, \xi'$, on dit que $\xi$ *est en dépendance booléenne de* $\xi'$ *dans* $\mathcal{S}$ si $\mathcal{B}^*_{\mathcal{S}}(\xi, \xi')$. On dit que $\xi$ *est en auto-dépendance booléenne dans* $\mathcal{S}$ si $\mathcal{B}^*_{\mathcal{S}}(\xi, \xi)$.

Nous dirons que $\mathcal{S}$ est sans cycles de dépendance booléenne si aucune variable dans $\mathsf{Vars}(\mathcal{S})$ n'est en auto-dépendance booléenne.

Finalement, nous introduisons la notion de système semi-simplifié qui définit des restrictions sur la forme des parties droites d'équations du système à la manière des systèmes simplifiés.

**Définition 4.25 (Système semi-simplifié)** Le système d'équations $\mathcal{S}$ sur $fonct(\tau, \mathcal{X}_{\mathrm{r}})$ est dit *semi-simplifié* si toute partie droite d'équation dans $\mathcal{S}$ est ou bien un terme booléen, ou bien de la forme

$$\mathbf{0} \quad \overline{\mathbf{0}} \quad \top \quad \bot \quad \alpha[\xi] \quad \xi \mid \xi' \quad \xi \parallel \xi' \quad \xi \lceil \xi' \quad \xi \rfloor\!\lceil \xi'$$

où $\xi$ et $\xi'$ sont des variables.

La différence qui existe entre un système simplifié et un système semi-simplifié est dans la restriction plus ou moins forte pour les termes booléens. En vue de leur utilisation, dans le premier cas nous avons besoin d'imposer la forme restreinte pour les termes booléens apparaissant en partie droite d'équation ($\xi \vee \xi'$ ou $\xi \wedge \xi'$), tandis que dans le second cas, nous autorisons n'importe quel terme booléen en partie droite d'équation.[3]

Remarquons qu'un système d'équations sur $fonct(\tau', \mathcal{X}_{\mathrm{r}})$ peut être transformé en un système semi-simplifié par un système de réécriture très semblable à celui présenté dans la section 4.3.1. Plus précisément, il suffit d'ajouter à ce système de réécriture des règles pour les symboles $\lceil$ et $\rfloor\!\lceil$ de la même manière que les règles pour les symboles $\mid$ et $\parallel$ et de modifier les règles (Red $\vee$ 3) et (Red $\wedge$ 3) en ajoutant la condition que $f$ et $f'$ ne doivent pas être des termes booléens. D'autre part, nous pouvons également simplifier un système d'équations sur $fonct(\tau', \mathcal{X}_{\mathrm{r}})$. Il suffit pour cela d'ajouter au système de réécriture les règles pour les symboles $\lceil$ et $\rfloor\!\lceil$ sans affaiblir les règles pour les symboles $\vee$ et $\wedge$. Il n'est pas difficile de voir qu'avec ces deux modifications du système de réécriture servant à la simplifications d'un système d'équations, la préservation de la solution telle que est énoncée dans le lemme 4.21 reste valide.

Nous pouvons maintenant définir un système d'équations normalisé.

**Définition 4.26 (Système normalisé)** Le système d'équations $\mathcal{S}$ sur $fonct(\tau', \mathcal{X}_{\mathrm{r}})$ est dit *normalisé* si les trois conditions ci-dessous sont vérifiées

(i) $\mathcal{S}$ est à composition gardée ;

(ii) $\mathcal{S}$ est semi-simplifié ;

(iii) $\mathcal{S}$ est sans cycles de dépendance booléenne.

Le résultat principal de cette section est le suivant

[3]Plus précisément, les systèmes semi-simplifiés seront obtenus comme étape de la construction d'un système d'équations normalisé. Affaiblir les conditions sur les termes booléens en partie droite d'équation a comme principal but de faciliter la preuve d'équivalence entre le système initial et le système d'équations normalisé correspondant obtenus par la méthode que nous présentons.

**Lemme 4.27** *Tout système d'équations $\mathcal{S}$ sur $fonct(\tau', \mathcal{X}_r)$ peut être effectivement transformé en un système d'équations normalisé $\mathcal{S}'$ sur $fonct(\tau', \mathcal{X}_r)$ équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$.*

Le reste de la section est consacré à la preuve de ce lemme. Nous donnons une méthode effective de construction du système d'équations normalisé. Nous présentons d'abord une méthode d'élimination des compositions non gardées d'un système et ensuite une méthode d'élimination des cycles de dépendance booléenne.

### Élimination des compositions non gardées

L'élimination des compositions non gardées est possible grâce à deux propriétés des formules et des systèmes d'équations que nous présentons d'abord. D'une part, pour tout système d'équations $\mathcal{S}$ et pour toute variable $\xi$ apparaissant dans $\mathcal{S}$, on peut décider si l'arbre vide appartient à la solution de $\mathcal{S}$ pour $\xi$ (voir le lemme 4.29). D'autre part, pour tous ensembles d'arbres $S, S'$, les expressions $S \mid^{\mathcal{T}} S'$ et $S \parallel^{\mathcal{T}} S'$ sont équivalentes à des expressions utilisant uniquement des compositions gardée qui peuvent être trouvées à condition de savoir si les ensembles $S, S'$ contiennent ou non l'arbre vide (voir la figure 4).

$$T \mid^{\mathcal{T}} S = \begin{cases} T \lceil^{\mathcal{T}} S & \text{si } \{\!|\} \notin T, \{\!|\} \notin S \\ T \lceil^{\mathcal{T}} S \vee (S \wedge \overline{\mathbf{0}}) & \text{si } \{\!|\} \in T, \{\!|\} \notin S \\ T \lceil^{\mathcal{T}} S \vee (T \wedge \overline{\mathbf{0}}) & \text{si } \{\!|\} \notin T, \{\!|\} \in S \\ T \lceil^{\mathcal{T}} S \vee (S \wedge \overline{\mathbf{0}}) \vee (T \wedge \overline{\mathbf{0}}) \vee \mathbf{0} & \text{si } \{\!|\} \in T, \{\!|\} \in S \end{cases}$$

$$T \parallel^{\mathcal{T}} S = \begin{cases} T \rfloor\lceil^{\mathcal{T}} S \wedge (T \vee \mathbf{0}) \wedge (S \vee \mathbf{0}) \wedge \overline{\mathbf{0}} & \text{si } \{\!|\} \notin T, \{\!|\} \notin S \\ T \rfloor\lceil^{\mathcal{T}} S \wedge (T \vee \mathbf{0}) & \text{si } \{\!|\} \in T, \{\!|\} \notin S \\ T \rfloor\lceil^{\mathcal{T}} S \wedge (S \vee \mathbf{0}) & \text{si } \{\!|\} \notin T, \{\!|\} \in S \\ T \rfloor\lceil^{\mathcal{T}} S & \text{si } \{\!|\} \in T, \{\!|\} \in S \end{cases}$$

FIG. 4 – Relation entre la composition gardée et la composition non gardée. $T$ et $S$ sont des ensembles d'arbres.

**Lemme 4.28** *Les égalités de la figure 4 sont vérifiées pour tous ensembles d'arbres $S, T$.*

*Preuve* Voir la section A.2 de l'annexe A.

□

Montrons maintenant que pour tout système d'équations $\mathcal{S}$, on peut vérifier si $\{\!|\}$ appartient à chacune des composantes de la solution de $\mathcal{S}$.

**Lemme 4.29** *Soit $\mathcal{S}$ un système d'équations sur $fonct(\tau', \mathcal{X}_r)$. Alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, on peut décider si $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$.*

*Preuve* Pour la preuve complète, voir la section A.2 l'annexe A. L'idée est que $\{\!|\}$ appartient toujours à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ (respectivement n'appartient jamais à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$) lorsque la partie droite

de l'équation pour $\xi$ dans $\mathcal{S}$ est $\top$ ou $\mathbf{0}$ (respectivement $\bot$ ou $\overline{\mathbf{0}}$ ou de la forme $\alpha[\xi']$). De même, si la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est de la forme $\xi_1 \vee \xi_2$ ou bien $\xi_1 \parallel \xi_2$, alors $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ seulement si $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi_1)$ ou à $Sol_{\mathcal{T}}(\mathcal{S}, \xi_2)$. Finalement, si la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est de la forme $\xi_1 \wedge \xi_2$ ou bien $\xi_1 \mid \xi_2$, alors $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ seulement si $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi_1)$ et à $Sol_{\mathcal{T}}(\mathcal{S}, \xi_2)$.

□

La preuve du lemme 4.29 nous permet d'énoncer le corollaire ci-dessous, qui n'est pas directement relié avec l'élimination des compositions non gardées mais représente un résultat intéressant en soi.

**Corollaire 4.30** *Pour tout terme $f$ sur $fixpt(\tau', \mathcal{X}_r)$ et pour toute valuation $\delta$, $\{\!|\}$ appartient à $[\![\mu\xi.f]\!]$ si et seulement si $\{\!|\}$ appartient à $[\![f]\!]_{\delta[\xi \to \{\!|\}]}$ et $\{\!|\}$ appartient à $[\![\nu\xi.f]\!]$ si et seulement si $\{\!|\}$ appartient à $[\![f]\!]_{\delta[\xi \to \mathcal{A}_\Sigma]}$.*

Maintenant en utilisant le lemme 4.29 et les égalités de la figure 4, nous pouvons définir le système d'équations $\mathsf{CG}(\mathcal{S})$ équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$. Considérons $x$ une variable dans $\mathsf{Vars}(\mathcal{S})$ et soient $\mathcal{S}', \mathcal{S}''$ les systèmes, $f$ le terme et $\kappa$ l'opérateur de point fixe tels que $\mathcal{S} = \mathcal{S}', x \overset{\kappa}{=} f, \mathcal{S}''$. Le système d'équations $\mathsf{CG}'(x, \mathcal{S})$ est défini par

$$\mathsf{CG}'(x,\ \mathcal{S}', x \overset{\kappa}{=} f, \mathcal{S}'') = \mathcal{S}', \mathsf{cg}(x \overset{\kappa}{=} f), \mathcal{S}'',$$

où le système d'équations $\mathsf{cg}(x \overset{\kappa}{=} f)$ est défini par :

$$\mathsf{cg}(x \overset{\kappa}{=} f) \ = \ x \overset{\kappa}{=} f \qquad \text{si } f \text{ n'est pas un terme à composition non gardée}$$

$$\mathsf{cg}(x \overset{\kappa}{=} y \,|\, z) = \begin{cases} x \overset{\kappa}{=} y \lceil z & \text{si } \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, z) \\ x \overset{\kappa}{=} y \lceil z \vee z & \text{si } \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, z) \\ x \overset{\kappa}{=} y \lceil z \vee y & \text{si } \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, z) \\ x \overset{\kappa}{=} y \lceil z \vee y \vee z \vee \mathbf{0} & \text{si } \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, z) \end{cases}$$

$$\mathsf{cg}(x \overset{\kappa}{=} y \,\|\, z) = \begin{cases} x \overset{\kappa}{=} y \rfloor\lceil z \wedge y \wedge z \wedge \overline{\mathbf{0}} & \text{si } \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, z) \\ x \overset{\kappa}{=} y \rfloor\lceil z \wedge y & \text{si } \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, z) \\ x \overset{\kappa}{=} y \rfloor\lceil z \wedge z & \text{si } \{\!|\} \notin Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, z) \\ x \overset{\kappa}{=} y \rfloor\lceil z & \text{si } \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, y), \{\!|\} \in Sol_{\mathcal{T}}(\mathcal{S}, z) \end{cases}$$

Soit $\mathsf{Vars}(\mathcal{S}) = x_1, \ldots, x_n$ et $x_1, \ldots, x_n \in X$. Alors $\mathsf{CG}(\mathcal{S})$ est défini par :

$$\mathsf{CG}(\mathcal{S}) = \mathsf{CG}'(x_n, \mathsf{CG}'(x_{n-1}, \ldots \mathsf{CG}'(x_1, \mathcal{S}) \ldots)).$$

**Lemme 4.31** *Pour tout système d'équations normalisé $\mathcal{S}$, le système d'équations $\mathsf{CG}(\mathcal{S})$ est un système d'équations à composition gardée équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$.*

*Preuve* Voir la section A.2 de l'annexe A.

□

### Élimination des auto dépendances booléennes

Considérons un système d'équations $\mathcal{S}$ semi simplifié et à composition gardée. Nous allons montrer comment éliminer les auto dépendances booléennes de $\mathcal{S}$ par un système de réécriture de systèmes d'équations.

Annonçons d'abord un résultat connu pour le $\mu$-calcul propositionnel qui permettra d'éliminer les cycles de dépendances booléennes.

**Proposition 4.32 ([Arnold and Niwinski, 2001], Proposition 9.1.2)** *Soit $t$ un terme propositionnel (càd sur la signature $\{\bot, \top, \vee, \wedge\}$ où $\bot$ et $\top$ sont des symboles d'arité zéro et $\vee$ et $\wedge$ sont des symboles binaires) et soit $\mathcal{D}$ un treillis distributif dans lequel $\bot$ et $\top$ sont interprétés comme la borne inférieure et la borne supérieure du domaine de $\mathcal{D}$ respectivement et $\vee$ et $\wedge$ sont interprétés comme la plus petite borne supérieure et la plus grande borne inférieure respectivement. Alors avec cette interprétation, pour toute variable $x$,*

$$\mu x.t = t\langle x \to \bot\rangle \qquad\qquad \nu x.t = t\langle x \to \top\rangle.$$

Un treillis est distributif s'il satisfait les lois de distributivité $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ et $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ pour tous $x, y, z$ éléments du domaine du treillis. Clairement, $(\mathcal{A}_\Sigma, \vee, \wedge)$ est un treillis distributif.

Pour tout système d'équations $\mathcal{S}$ sur $fonct(\tau', \{\xi_1, \ldots, \xi_n\})$ et pour tout $i$ dans $1..n$, nous définissons les systèmes d'équations $\mathsf{transforme}(\mathcal{S}, \xi_i)$ et $\mathsf{élimine}(\mathcal{S}, \xi_i)$ par des transformations syntaxiques de $\mathcal{S}$. Pour la suite, supposons que $\mathcal{S}$ est le système

$$\xi_1 \overset{\kappa_1}{=} f_1\ , \ldots,\ \xi_i \overset{\kappa_i}{=} f_i,\ \ldots,\ \xi_n \overset{\kappa_n}{=} f_n.$$

Le système $\mathsf{transforme}(\mathcal{S}, \xi_i)$ est défini par :

$$\mathsf{transforme}(\mathcal{S}, \xi_i) = \mathcal{S} \qquad \text{si } f_i \text{ n'est pas un terme booléen,}$$

$$\mathsf{transforme}(\mathcal{S}, \xi_i) = \begin{cases} \xi_1 \overset{\kappa_1}{=} f_1,\ \ldots,\ \xi_i \overset{\kappa_i}{=} f_i, \\ \xi_{i+1} \overset{\kappa_{i+1}}{=} g_{i+1},\ \ldots,\ \xi_n \overset{\kappa_n}{=} g_n \end{cases} \qquad \text{sinon,}$$

où pour tout $j$ dans $i+1..n$, $g_j = f_j$ si $f_j$ n'est pas un terme booléen et $g_j = f_j\langle \xi_i \to f_i\rangle$ si $f_j$ est un terme booléen.

Le système $\mathsf{élimine}(\mathcal{S}, \xi_i)$ est défini par :

$$\mathsf{élimine}(\mathcal{S}, \xi_i) = \mathcal{S} \qquad \text{si } f_i \text{ n'est pas un terme booléen ;}$$

$$\mathsf{élimine}(\mathcal{S}, \xi_i) = \begin{cases} \xi_1 \overset{\kappa_1}{=} f_1,\ \ldots,\ \xi_{i-1} \overset{\kappa_{i-1}}{=} f_{i-1}, \\ \xi_i \overset{\kappa_i}{=} g, \\ \xi_{i+1} \overset{\kappa_{i+1}}{=} f_{i+1},\ \ldots,\ \xi_n \overset{\kappa_n}{=} f_n \end{cases} \qquad \text{sinon.}$$

où $g = f_i\langle \xi_i \to \bot\rangle$ si $\kappa_i = \mu$ et $g = f_i\langle \xi_i \to \top\rangle$ si $\kappa_i = \nu$.

**Lemme 4.33** *Pour tout système d'équations $\mathcal{S}$ sur $fonct(\tau', \{\xi_1, \ldots, \xi_n\})$ et pour tout $i$ dans $1..n$ :*

*(i) les trois systèmes* $\mathcal{S}$, $\mathsf{transforme}(\mathcal{S}, \xi_i)$ *et* $\mathsf{élimine}(\mathcal{S}, \xi_i)$ *ont la même solution lorsque les parties droites d'équations sont interprétées par* $\mathcal{T}$ *;*

*(ii) si* $\mathcal{S}$ *est un système d'équations semi-simplifié et à composition gardée, alors les systèmes* $\mathsf{transforme}(\mathcal{S}, \xi_i)$ *et* $\mathsf{élimine}(\mathcal{S}, \xi_i)$ *le sont aussi. De plus, pour tout* $j$ *dans* $1..n$*, si la partie droite de l'équation pour la variable* $\xi_j$ *dans* $\mathcal{S}$ *est un terme booléen, alors les parties droites des équations pour* $\xi_j$ *dans* $\mathsf{transforme}(\mathcal{S}, \xi_i)$ *et* $\mathsf{élimine}(\mathcal{S}, \xi_i)$ *sont aussi des termes booléens.*

*Preuve* Pour (i), le fait que $\mathsf{transforme}(\mathcal{S}, \xi_i)$ a la même solution que $\mathcal{S}$ est simple conséquence de la proposition 4.5 et le fait que $\mathsf{élimine}(\mathcal{S}, \xi_i)$ a la même solution que $\mathcal{S}$ est conséquence de la proposition 4.32 : en effet l'interprétation $\mathcal{T}$ de $\tau'$ satisfait les conditions de cette proposition.

Pour (ii), par définition $\mathsf{transforme}(\mathcal{S}, \xi_i)$ diffère $\mathcal{S}$ uniquement par la partie droite de l'équation pour $\xi_i$ si celle-ci est un terme booléen dans $\mathcal{S}$. De plus, si c'est le cas, la partie droite de l'équation pour $\xi_i$ dans $\mathsf{transforme}(\mathcal{S}, \xi_i)$ est obtenue en remplaçant dans un terme booléen les occurrences de la variable $\xi_j$ par un terme booléen dans un terme booléen, ce qui nous donne un terme booléen comme résultat et donc préserve le caractère simplifié du système d'équations. Des arguments similaires sont valables pour $\mathsf{élimine}(\mathcal{S}, \xi_i)$, qui est obtenu à partir de $\mathcal{S}$ en modifiant certaines parties droites d'équations qui sont des termes booléens par d'autres termes booléens.

□

Les deux transformations ci-dessus vont être utilisées pour transformer un système d'équations $\mathcal{S}$ en un système d'équations équivalent mais sans cycles de dépendances booléennes. Soit comme précédemment $\mathcal{S}$ un système d'équations sur les variables $\xi_1, \ldots, \xi_n$, et soit $(\mathcal{S}_i)_{i \in 0..n}$ la suite de systèmes d'équations suivante : $\mathcal{S}_0 = \mathcal{S}$ et pour tout $i$ dans $1..n$,

$$\mathcal{S}_i = \mathsf{transforme}(\mathsf{élimine}(\mathcal{S}_{i-1}, \xi_i), \xi_i).$$

Le lemme qui suit établit que pour tout $i$, dans le système d'équations $\mathcal{S}_i$, les variables $\xi_1, \ldots, \xi_i$ ne sont en dépendance booléenne que de variables les succédant dans la séquence $\mathsf{Vars}(\mathcal{S})$.

**Lemme 4.34** *Si* $\mathcal{S}$ *est un système d'équations semi-simplifié, alors pour tout* $i$ *dans* $0..n$ *et pour tous* $j, k$ *dans* $1..n$ *tels que* $j \leq i$*, si* $\mathcal{B}_{\mathcal{S}_i}(\xi_j, \xi_k)$*, alors* $k > j$*.*

*Preuve* Voir la section A.2 de l'annexe A.

□

Il apparaît du lemme précédent que dans le système d'équations $\mathcal{S}_n$, toute variables $\xi_j$ est en dépendance booléenne uniquement de variables $\xi_k$ de rang strictement supérieur, càd $\mathcal{B}^*_{\mathcal{S}_i}(\xi_j, \xi_k)$ implique $k > j$, ce qui donne naturellement corollaire suivant.

**Corollaire 4.35** *Le système d'équations* $\mathcal{S}_n$ *est sans cycles de dépendance booléenne.*

De même, une conséquence immédiate du lemme 4.33 est le suivant

**Corollaire 4.36** *Le système d'équations* $\mathcal{S}_n$ *est équivalent au système d'équations* $\mathcal{S}$*.*

Nous avons donc défini, à l'aide des transformations de systèmes d'équations purement syntaxiques transforme$(\mathcal{S}, \xi_i)$ et élimine$(\mathcal{S}, \xi_i)$, une méthode pour transformer un système d'équations semi-simplifié et à composition gardée $\mathcal{S}$ en un système équivalent qui est de plus sans cycles de dépendances booléennes, càd en un système équivalent normalisé.

Revenons maintenant à la preuve du lemme 4.27, càd montrer que tout système d'équations peut effectivement être transformé en un système d'équations normalisé. Étant donné le système d'équations $\mathcal{S}$, nous commençons par simplifier $\mathcal{S}$ en utilisant la procédure décrite dans la section 4.3.1 ; soit $\mathcal{S}_1$ le système obtenu. Par le lemme 4.21, $\mathcal{S}_1$ est simplifié et équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$. Ensuite, nous transformons $\mathcal{S}_1$ en utilisant la procédure décrite dans la section 4.3.2 ; soit $\mathcal{S}_2$ le système obtenu. Par le lemme 4.31, $\mathcal{S}_2$ est un système à composition gardée équivalent à $\mathcal{S}_1$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S}_1)\}$ ; donc par transitivité de la relation d'équivalence, $\mathcal{S}_2$ est équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$. Par contre, $\mathcal{S}_2$ n'est pas forcément un système semi-simplifié. Nous devons appliquer à $\mathcal{S}_2$ la procédure de semi-simplification ; soit $\mathcal{S}_3$ le système obtenu. Alors $\mathcal{S}_3$ est un système semi-simplifié et à composition gardée et est équivalent à $\mathcal{S}_2$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S}_2)\}$, donc équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$. Finalement, nous appliquons à $\mathcal{S}_3$ la procédure décrite dans le section 4.3.2 ; soit $\mathcal{S}'$ le système obtenu. Par le lemme 4.33, $\mathcal{S}'$ est semi-simplifié et à composition gardée ; par le corollaire 4.35, $\mathcal{S}'$ est sans cycles de dépendances booléennes et, par le corollaire 4.36, $\mathcal{S}'$ est équivalent à $\mathcal{S}_3$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S}_3)\}$, donc aussi à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$. Ceci termine la preuve du lemme 4.27.

**Exemple 4.5** Considérons la formule $f = \mu\xi.(\alpha[\xi] \vee \mathbf{0}) \,|\, (\beta[\mathbf{0}] \vee \xi)$. Nous allons montrer les différentes étapes de la construction d'un système d'équations pour cette formule et sa transformation en un système d'équations normalisé.

La formule $f$ étant positive, nous pouvons directement construire le système d'équations $\mathsf{Eq}(f)$. Il est facile de voir que c'est le système à une seule équation

$$\xi \stackrel{\mu}{=} (\alpha[\xi] \vee \mathbf{0}) \,|\, (\beta[\mathbf{0}] \vee \xi)$$

que nous noterons $\mathcal{S}$ pour la suite.

Le système d'équations simplifié obtenu à partir de $\mathcal{S}$ par la procédure de simplification décrite dans la section 4.3.1 est le système $\mathcal{S}_{\text{simpl}}$ suivant :

$$\xi_4 \stackrel{\mu}{=} \alpha[\xi], \qquad \xi_5 \stackrel{\mu}{=} \mathbf{0}, \qquad \xi_1 \stackrel{\mu}{=} \xi_4 \vee \xi_5, \qquad \xi_6 \stackrel{\mu}{=} \mathbf{0},$$

$$\xi_3 \stackrel{\mu}{=} \beta[\xi_6], \qquad \xi_2 \stackrel{\mu}{=} \xi_3 \vee \xi, \qquad \xi \stackrel{\mu}{=} \xi_1 \,|\, \xi_2$$

Nous devons d'abord éliminer les compositions gardées du système d'équations $\mathcal{S}_{\text{simpl}}$. Pour cela, nous commencons par déterminer, pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_{\text{simpl}})$, si l'arbre $\{\!|\}$ appartient à la solution de $\mathcal{S}_{\text{simpl}}$ pour cette variable. Nous obtenons que $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}_{\text{simpl}}, \xi)$ lorsque $\xi$ est l'une des variables $\xi_1, \xi_5, \xi_6$ et $\{\!|\}$ n'appartient pas à $Sol_{\mathcal{T}}(\mathcal{S}_{\text{simpl}}, \xi)$ lorsque $\xi$ est l'une des variables $\xi, \xi_2, \xi_3, \xi_4$.

Avec ces informations, nous pouvons éliminer les compositions non gardées de $\mathcal{S}_{\text{simpl}}$ pour obtenir le système à composition gardée $\mathcal{S}_{\text{cg}}$ équivalent à $\mathcal{S}_{\text{simpl}}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S}_{\text{simpl}})\}$. La seule équation à composition (non gardée) de $\mathcal{S}_{\text{simpl}}$ est la dernière équation.

Donc, $\mathcal{S}_{\text{cg}}$ est identique à $\mathcal{S}_{\text{simpl}}$ à part pour la dernière équation qui devient $\xi \stackrel{\mu}{=} \xi_1 \rceil \xi_2 \vee \xi_2$. Remarquons qu'alors le système d'équations $\mathcal{S}_{\text{cg}}$ n'est pas un système semi-simplifié. Or, pour appliquer la prochaine étape de normalisation, nous avons besoin d'un système semi-simplifié. En appliquant la procédure de simplification décrite dans la section 4.3.1, nous obtenons le système d'équations $\mathcal{S}'_{\text{cg}}$ équivalent à $\mathcal{S}_{\text{simpl}}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S}_{\text{simpl}})\}$ suivant :

$$\xi_4 \stackrel{\mu}{=} \alpha[\xi], \qquad \xi_5 \stackrel{\mu}{=} \mathbf{0}, \qquad \xi_1 \stackrel{\mu}{=} \xi_4 \vee \xi_5, \qquad \xi_6 \stackrel{\mu}{=} \mathbf{0},$$

$$\xi_3 \stackrel{\mu}{=} \beta[\xi_6], \qquad \xi_2 \stackrel{\mu}{=} \xi_3 \vee \xi, \qquad \xi_7 \stackrel{\mu}{=} \xi_1 \rceil \xi_2, \qquad \xi \stackrel{\mu}{=} \xi_7 \vee \xi_2$$

Pour obtenir un système d'équations normalisé, il nous reste maintenant à éliminer les cycles de dépendance booléenne de $\mathcal{S}'_{\text{cg}}$. Plus précisément, le système $\mathcal{S}_{\text{norm}}$ est obtenu par une succession d'étapes de transformation, une par variable du système $\mathcal{S}'_{\text{cg}}$, et qui sont effectués dans l'ordre des variables dans la séquence $\mathsf{Vars}(\mathcal{S}'_{\text{cg}})$. Ainsi, le système d'équations $\mathcal{S}_{\xi'}$, obtenu à l'étape correspondant à la variable $\xi'$, est $\mathsf{transforme}(\mathsf{élimine}(\mathcal{S}_{\xi''}, \xi'), \xi')$, où $\xi''$ est la variable qui précède immédiatement $\xi'$ dans la séquence des variables du système $\mathcal{S}'_{\text{cg}}$.

Pour nous éviter de dérouler toutes les huit étapes sur l'exemple, remarquons d'abord quelques propriétés des transformations $\mathsf{élimine}(\mathcal{S}', \xi')$ et $\mathsf{transforme}(\mathcal{S}', \xi')$, valables pour tout système $\mathcal{S}'$ et toute variable $\xi'$ dans $\mathsf{Vars}(\mathcal{S}')$, et qui découlent directement des définitions :

(i) si la partie droite de l'équation pour $\xi'$ dans $\mathcal{S}'$ n'est pas un terme booléen, alors les systèmes $\mathsf{élimine}(\mathcal{S}', \xi')$ et $\mathsf{transforme}(\mathcal{S}', \xi')$ sont tous deux égaux à $\mathcal{S}'$ ;

(ii) si la partie droite de l'équation pour $\xi'$ est un terme booléen qui ne contient pas d'occurrence de la variable $\xi'$, alors $\mathsf{élimine}(\mathcal{S}', \xi')$ est égal à $\mathcal{S}'$.

Revenant à notre exemple ; par la remarque précédente nous obtenons que $\mathcal{S}_{\xi_5} = \mathcal{S}_{\xi_4} = \mathcal{S}'_{\text{cg}}$ et $\mathsf{élimine}(\mathcal{S}_{\xi_5}, \xi_1) = \mathcal{S}_{\xi_5}$. Il n'est pas difficile de voir que $\mathsf{transforme}(\mathsf{élimine}(\mathcal{S}_{\xi_5}, \xi_1), \xi_1)$ est aussi égal à $\mathcal{S}_{\xi_5}$, ceci à cause du fait que la variable $\xi_1$ n'apparaît pas en partie droite des équations pour les variables $\xi_2$ et $\xi$ (qui sont les variables après $\xi_1$ pour lesquelles la partie droite d'équation est un terme booléen). Nous en déduisons que $\mathcal{S}_{\xi_1} = \mathcal{S}'_{\text{cg}}$. Encore une fois par les remarques précédentes, nous déduisons que $\mathcal{S}_{\xi_3}$ est aussi égal au système de départ $\mathcal{S}_{\text{cg}}$. Ensuite, pour le système $\mathcal{S}_{\xi_2}$ nous obtenons

$$\xi_4 \stackrel{\mu}{=} \alpha[\xi], \qquad \xi_5 \stackrel{\mu}{=} \mathbf{0}, \qquad \xi_1 \stackrel{\mu}{=} \xi_4 \vee \xi_5, \qquad \xi_6 \stackrel{\mu}{=} \mathbf{0},$$

$$\xi_3 \stackrel{\mu}{=} \beta[\xi_6], \qquad \xi_2 \stackrel{\mu}{=} \xi_3 \vee \xi, \qquad \xi_7 \stackrel{\mu}{=} \xi_1 \rceil \xi_2, \qquad \xi \stackrel{\mu}{=} \xi_7 \vee (\xi_3 \vee \xi)$$

$\mathcal{S}_{\xi_1}$ a été obtenu en remplaçant $\xi_3 \vee \xi$ pour $\xi_2$ dans la dernière équation. Encore une fois par la remarque précédente, nous obtenons que $\mathcal{S}_{\xi_7}$ est égal au système $\mathcal{S}_{\xi_2}$. Finalement, le système $\mathsf{élimine}(\mathcal{S}_{\xi_7}, \xi)$ est égal au système $\mathcal{S}_{\xi_2}$ sauf pour la dernière équation qui devient

$$\xi \stackrel{\mu}{=} \xi_7 \vee (\xi_3 \vee \bot)$$

et le système $\mathsf{transforme}(\mathsf{élimine}(\mathcal{S}_{\xi_7}, \xi), \xi)$ est donc

$$\xi_4 \stackrel{\mu}{=} \alpha[\xi], \qquad \xi_5 \stackrel{\mu}{=} \mathbf{0}, \qquad \xi_1 \stackrel{\mu}{=} \xi_4 \vee \xi_5, \qquad \xi_6 \stackrel{\mu}{=} \mathbf{0},$$

$$\xi_3 \stackrel{\mu}{=} \beta[\xi_6], \qquad \xi_2 \stackrel{\mu}{=} \xi_3 \vee \xi, \qquad \xi_7 \stackrel{\mu}{=} \xi_1 \rceil \xi_2, \qquad \xi \stackrel{\mu}{=} \xi_7 \vee (\xi_3 \vee \bot)$$

Par définition, $\mathcal{S}_{\text{norm}}$ est égal à $\mathcal{S}_{\xi}$, et donc $\mathcal{S}_{\text{norm}}$ est le système ci-dessus.

## 4.4 Unicité du point fixe

Nous allons montrer que dans un système d'équations normalisé, les opérateurs de plus petit et plus grand point fixe peuvent être inter-changés sans que cela ne modifie la solution du système. Ce résultat peut être vu comme une généralisation de l'unicité du point fixe pour les formules à récursion descendante (voir ci-dessous) établi dans [Dal Zilio, 2001]. Nous commençons par présenter brièvement le résultat de [Dal Zilio, 2001] dans les termes utilisés plus tard dans cette section.

Une formule $\phi$ est dite *à récursion descendante pour* $\xi$ si dans $\phi$, toute occurrence de la variable $\xi$ apparaît sous un opérateur d'emboîtement. [4] On montre alors que si $\phi$ est une formule gardée dans la variable $\xi$, alors pour toute valuation des variables $\delta$, $[\![\mu\xi.\phi]\!]_\delta = [\![\nu\xi.\phi]\!]_\delta$. Pour la preuve, l'auteur utilise la propriété suivante : pour tous ensembles d'arbres $S, T$ et pour tout arbre $t$ appartenant à $[\![\phi]\!]_{\delta[\xi\to S]}$ mais n'appartenant pas à $[\![\phi]\!]_{\delta[\xi\to T]}$, la hauteur de $t$ est supérieure ou égale à $1 + h(t')$, où $t'$ est un plus petit arbre (pour sa hauteur) appartenant à $S$ mais n'appartenant pas à $T$. De cette propriété découle le fait que la fonction qui, à l'ensemble d'arbres $S$ associe $[\![\phi]\!]_{\delta[\xi\to S]}$ est strictement contractante, et donc a un unique point fixe (voir plus bas pour la définition d'une fonction contractante).

Dans ce qui suit, nous élargissons ce résultat en montrant que, intuitivement, si $f$ est un terme à composition gardée et sans dépendance booléenne (càd $f$ satisfait les conditions pour les termes en partie droite d'une équation dans un système normalisé), alors la fonction $f^{\mathcal{T}}$ est contractante et donc à unique point fixe.

Nous commençons par identifier une condition suffisante permettant d'inter-changer les opérateurs de point fixe dans un système d'équations.

**Lemme 4.37** *Soient $\langle E, \leq\rangle$ un treillis complet et $\mathcal{S}$ le système d'équations*

$$x_1 \stackrel{\kappa_1}{=} f_1(x_1, \ldots, x_n), \ldots, x_n \stackrel{\kappa_n}{=} f_n(x_1, \ldots, x_n),$$

*où pour tout $i$ dans $1..k$, $f_i(x_1, \ldots, x_n)$ est une fonction monotone de $E^n$ dans $E$. Supposons que pour tout $i$ dans $1..k$ et pour tous $h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_n$ éléments de $E$ on a*

$$\mu x_i.f_i(h_1, \ldots, h_{i-1}, x, h_{i+1}, \ldots, h_n) = \nu x_i.f_i(h_1, \ldots, h_{i-1}, x, h_{i+1}, \ldots, h_n) \qquad (4.2)$$

*Alors le système d'équations $\mathcal{S}$ est équivalent à tout système $\mathcal{S}'$ de la forme*

$$x_1 \stackrel{\kappa'_1}{=} f_1(x_1, \ldots, x_n), \ldots, x_n \stackrel{\kappa'_n}{=} f_n(x_1, \ldots, x_n),$$

*où les $\kappa'_i$ sont des opérateurs de points fixes arbitraires.*

*Preuve* Il suffit de montrer que ($\star$) si (4.2) est vérifié pour un certain $i$ dans $1..n$, alors on peut changer l'opérateur de point fixe $\kappa_i$. La preuve du lemme est alors une récurrence facile sur le nombre d'opérateurs de point fixe ayant changé. Pour montrer ($\star$), il suffit d'utiliser la proposition 4.4 en coupant le système d'équations entre les variables $x_{i-1}$ et $x_i$.

[4] Dans [Dal Zilio, 2001], les formules à récursion descendante sont appelées « formules gardées ». Comme nous allons le voir bientôt, l'utilisation du terme « gardé » pour la composition gardée n'est pas un hasard. En effet, c'est cette « garde » qui garantit l'unicité du point fixe.

□

Dans la suite de cette section nous allons montrer que les conditions du lemme 4.37 sont satisfaites par tout système d'équations sur $fonct(\tau', \mathcal{X}_r)$ normalisé en considérant sa solution pour l'interprétation $\mathcal{T}$. Plus précisément, nous allons montrer que si $f \in fonct(\tau', \mathcal{X}_r)$ est la partie droite de la variable $\xi$ dans un système d'équations normalisé, alors $f$ admet un et un seul point fixe (par rapport à $\xi$). Pour cela, nous utilisons le théorème suivant.

**Théorème 4.38 (Théorème du point fixe des applications contractantes)** *Si $(E, d)$ est un espace métrique complet (pour la distance $d$) et $f : E \to E$ est une application strictement contractante (pour la distance $d$), alors $f$ admet un unique point fixe.*

Rappelons d'abord les éléments de topologie nécessaires pour la manipulation de ce théorème. Un *espace métrique* $(E, d)$ est un espace $E$ muni d'une distance $d$. Un espace métrique $(E, d)$ est dit *complet pour la distance $d$* si toute suite de Cauchy d'éléments de $E$ converge dans $E$. Une suite $(S_n)_{n\in\mathbb{N}}$ est dite *suite de Cauchy* si pour tout $\varepsilon > 0$, il existe $N \in \mathbb{N}$ tel que pour tous $n, m \geq N$, $d(S_n, S_m) < \varepsilon$. Une application $f$ d'un espace métrique $(E, d)$ dans lui même est dite *application strictement contractante* s'il existe une constante $0 \leq c < 1$ telle que pour tous $x, y$ appartenant à $E$, $d(f(x), f(y)) \leq c\, d(x, y)$.

Dans ce qui suit, nous allons définir une distance $d$ pour l'ensemble $\wp(\mathcal{A}_\Sigma)$, nous allons montrer que cet ensemble est complet pour la distance $d$ et finalement nous allons montrer que toute fonction apparaissant en partie droite d'un système d'équations sur $fonct(\tau', \mathcal{X}_r)$ normalisé est strictement contractante.

Soit l'application $d : \wp(\mathcal{A}_\Sigma), \wp(\mathcal{A}_\Sigma) \to \mathbb{R}$ définie par : pour tous $S, T \subseteq \mathcal{A}_\Sigma$,

$$\begin{cases} d(S, S) = 0 \\ d(S, T) = 2^{-x} \text{ où } x = \inf\{|t| \mid t \in (S \cup T) \setminus (S \cap T)\} \text{ pour } S \neq T \end{cases}$$

Autrement dit, la distance $d(S, T)$ est $2^{|t|}$ où $t$ est un plus petit arbre, du point de vue de la taille, qui appartient seulement à l'un des ensembles $S, T$.

**Lemme 4.39** *L'application $d$ est une distance pour $\wp(\mathcal{A}_\Sigma)$.*

*Preuve* Voir la section A.2 de l'annexe A.

□

Montrons à présent que $\wp(\mathcal{A}_\Sigma)$ est un espace complet pour la distance $d$.

**Lemme 4.40** *$\wp(\mathcal{A}_\Sigma)$ est un espace complet pour la distance $d$.*

*Preuve* Il nous faut montrer que toute suite de Cauchy d'éléments de $\wp(\mathcal{A}_\Sigma)$ converge dans $\wp(\mathcal{A}_\Sigma)$.

Soit $(U_n)_{n\in\mathbb{N}}$ une suite de Cauchy d'éléments de $\wp(\mathcal{A}_\Sigma)$. Pour tout $k \in \mathbb{N}$, soit $N_k$ le plus petit entier naturel vérifiant que pour tous $n, m \geq N_k$, $d(U_n, U_m) < 2^{-k}$. Il est évident de cette définition que

($\star$) pour tout $k$, $N_{k+1} \geq N_k$.

Soit $k$ un entier naturel quelconque et soit $t$ un arbre de taille $k$. Supposons que $t$ appartient à $U_{N_k}$ et supposons que pour un certain $n > N_k$, $t$ n'appartient pas à $U_n$. Donc $t \in (U_{N_k} \cup U_n) \setminus (U_{N_k} \cap U_n)$ et nous en déduisons que $d(U_{N_k}, U_n) \geq 2^{-k}$. Or, nous savons qu'au contraire $d(U_{N_k}, U_n) < 2^{-k}$, donc pour tout arbre $t$ de taille $k$, si $t$ appartient à $U_{N_k}$, alors $t$ appartient à $U_n$, ceci pour tout $n > N_k$. De même, si nous supposons que $t$ n'appartient pas à $U_{N_k}$ mais $t$ appartient à $U_n$ pour un certain $n > N_k$, nous arrivons à une contradiction. En utilisant ($\star$), nous résumons ces deux constats dans la propriété suivante :

$$\text{pour tout } k \in \mathbb{N} \text{ et pour tout arbre } t \text{ de taille inférieure à } k \text{ et pour tout } n \geq N_k, \quad t \in U_{N_k} \text{ si et seulement si } t \in U_n. \tag{4.3}$$

Soit maintenant $U$ l'ensemble d'arbres défini par :

pour tout $k \in \mathbb{N}$ et pour tout arbre $t$ de taille $k$, $t$ appartient à $U$ si $t$ appartient à $U_{N_k}$.

Nous allons montrer que la suite $(U_n)_{n\in\mathbb{N}}$ converge vers $U$. Plus précisément, nous montrons que pour tout $\varepsilon > 0$, si $k$ est l'entier naturel tel que $2^{-(k+1)} < \varepsilon \leq 2^{-k}$, alors l'indice $N_k$ satisfait que pour tout $n \geq N_k$, $d(U_n, U) < \varepsilon$. En effet, soit $n \geq N_k$. Par définition de $U$ nous savons que les ensembles $U_{N_k}$ et $U$ contiennent les mêmes arbres de taille inférieure à $k$. Par (4.3) nous savons que les ensembles $U_n$ et $U_{N_k}$ contiennent les mêmes arbres de taille inférieure à $k$. Donc les ensembles $U_n$ et $U$ contiennent les mêmes arbres de taille inférieure à $k$. Par définition de la distance $d$, nous concluons $d(U_n, U) \leq 2^{-(k+1)} < \varepsilon$.

□

Pour pouvoir appliquer le théorème du point fixe des fonctions contractantes et montrer l'unicité du point fixe des fonctions en partie droite d'équation d'un système normalisé, il nous reste à montrer que ces fonctions sont contractantes.

**Lemme 4.41** *Soit $\mathcal{S}$ le système d'équations normalisé*

$$x_1 \overset{\kappa_1}{=} f_1, \ldots, x_n \overset{\kappa_n}{=} f_n.$$

*Pour tout $i$ dans $1..n$ et pour tous ensembles d'arbres $S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$, la fonction $g(x_i) = f_i^{\mathcal{T}}(S_1, \ldots, S_{i-1}, x_i, S_{i+1}, \ldots, S_n)$ est strictement contractante.*

*Preuve* Voir la section A.2 de l'annexe A.

□

**Corollaire 4.42** *Supposons les hypothèses du lemme 4.41. Alors pour tout $i$ dans $1..n$ et pour tous ensembles d'arbres $S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$, $\mu x.g(x) = \nu x.g(x)$.*

*Preuve* Conséquence immédiate du lemme 4.41 et du théorème du point fixe des applications contractantes (théorème 4.38).

□

Maintenant nous avons tous les ingrédients pour énoncer le résultat principal de cette section.

**Théorème 4.43** *Soit $\mathcal{S}$ un système d'équations normalisé. Alors pour tout système d'équations $\mathcal{S}'$ obtenu à partir de $\mathcal{S}$ en changeant un ou plusieurs opérateurs de point fixe, les solutions de $\mathcal{S}$ et $\mathcal{S}'$ coïncident.*

*Preuve* Il suffit de montrer que les hypothèses du lemme 4.37 sont satisfaites, ce qui est vrai par le corollaire 4.42.

□

### Comment transférer ce résultat sur les formules de la logique spatiale

Les résultats de cette section nous permettent de transformer toute formule positive close $\phi$ en une formule positive équivalente $\phi'$ et dans laquelle tous les opérateurs de point fixe sont des opérateurs de plus petit point fixe. Pour cela, il faut construire le système d'équations $\mathsf{Eq}(\phi)$, transformer $\mathsf{Eq}(\phi)$ en un système d'équations normalisé $\mathcal{S}$ et transformer tous les opérateurs de point fixe dans $\mathcal{S}$ en $\mu$. Rappelons que $\mathcal{S}$ est un système d'équations sur $fonct(\tau', \mathcal{X}_{\mathrm{r}})$. Mais $\mathcal{S}$ peut être transformé, de façon syntaxique, en un système d'équations $\mathcal{S}'$ sur $fonct(\tau, \mathcal{X}_{\mathrm{r}})$ (càd éliminer les opérateurs $\lceil$ et $\rfloor\!\lceil$) simplement en utilisant les correspondances qui existent entre les opérateurs de composition gardée et les opérateurs de composition initiaux (voir la remarque 4.22). Il suffit ensuite de construire la formule $\mathsf{formule}(\mathcal{S}')$ qui est équivalente à $\phi$ et ne contient que des opérateurs de plus petit point fixe.

Il serait intéressant de voir comment le résultat de cette section peut être utilisé pour caractériser les formules logiques sur la syntaxe d'origine (sans opérateurs dérivés), pour lesquelles nous avons unicité du point fixe. Ou mieux encore, se poser la question s'il existe une caractérisation syntaxique de ces formules (comme dans [Dal Zilio, 2001]).

Remarquons d'abord que si la logique spatiale avait été définie avec l'opérateur de composition gardée $\lceil$ et non avec la composition $|$, alors toute formule $\phi$ dans laquelle la variable $\xi$ n'est pas en position booléenne définirait une formule à unique point fixe. [5] Donc, pour une formule $\phi$ sur la vraie syntaxe de la logique (opérateur $|$), la condition pour l'unicité du point fixe serait que $\phi$ ne contienne pas d'occurrence de $\xi$ en position booléenne, et que toutes les compositions dans $\phi$ soient équivalentes à des compositions gardées. Intuitivement, cela veut dire que pour toute valuation $\delta$, on aurait $[\![\phi]\!]_\delta = [\![\phi']\!]_\delta$, où la formule (ou plutôt le terme) $\phi'$ est obtenue à partir de $\phi$ en remplaçant toute occurrence de l'opérateur de composition $|$ par l'opérateur de composition gardée $\lceil$. Nous ne voyons pas, à ce jour, une caractérisation syntaxique des formules satisfaisant cette propriété.

## Conclusion

Le premier résultat de ce chapitre est que les formules $\mathrm{LS}_{|\exists}$ peuvent être représentées par des systèmes d'équations dont les parties droites d'équations sont des fonctions ayant une forme particulière. A tout système d'équations $\mathcal{S}$ ayant cette forme correspond une formule close $\phi$ et à toute formule close $\phi$ correspond un système d'équations $\mathcal{S}$ tels que $[\![\phi]\!]$ est égal à la solution

[5] Il faut évidemment que $\phi$ soit monotone dans $\xi$ pour que le point fixe existe, mais nous supposons que c'est toujours le cas.

de $\mathcal{S}$ pour la dernière variable du système. La construction de $\mathcal{S}$ à partir de $\phi$ et vice versa est purement syntaxique. Cette équivalence entre systèmes d'équations et formules logiques nous permettra d'utiliser dans nombreuses situation des systèmes à la place de formules, ce qui facilite certaines preuves notamment grâce au fait qu'elle permet de raisonner sur les sous formules à point fixe indépendemment les unes des autres, même si celles-ci sont imbriquées.

Un autre résultat de ce chapitre est que la négation peut être éliminée des formules logiques grâce aux opérateurs duaux $(\bot, \wedge, \overline{\mathbf{0}}, ||, \nu)$. Plus précisément, pour toute formule close $\phi$ on peut construire la formule close sans négation $\mathsf{pos}(\phi)$ utilisant les opérateurs duaux telle que $[\![\phi]\!] = [\![\mathsf{pos}(\phi)]\!]$. Encore une fois, la construction est purement syntaxique. Grâce à ce résultat, dans la suite nous considérons uniquement les formules sans négation.

Finalement, pour tout système d'équation nous avons défini deux formes normales. La principale utilité de ces formes normales sera de permettre la définition dans le chapitre 6 d'un automate à contraintes numériques reconnaissant le langage d'arbres défini par une formule. Une première utilisation des formes normales d'un système d'équations est la possibilité d'éliminer les plus grands points fixes d'une formule positive.

# Chapitre 5

# Fragments indécidables de la logique spatiale

Il est connu que le problème de satisfiabilité est indécidable pour la logique spatiale avec quantification [Charatonik and Talbot, 2001, Charatonik et al., 2003] (voir la section 5.1). C'est pourquoi nous nous intéressons ici à la décidabilité de la logique spatiale sans quantification. Nous montrons dans ce chapitre que la satisfiabilité est indécidable pour le fragment de la logique spatiale sans quantification. Nous commençons par rappeler brièvement la preuve d'indécidabilité pour le cas avec quantification dans la section 5.1. Nous présentons ensuite les machines à deux compteurs (section 5.2) qui seront utilisées dans la section 5.3 pour montrer l'indécidabilité de la satisfiabilité de la logique sans quantification. Nous terminons le chapitre par une comparaison de la logique PMSO et le fragment sans quantification de la logique spatiale, montrant notamment que ce dernier est strictement plus expressif que PMSO.

## 5.1 Indécidabilité de la logique en présence de quantification

Dans [Charatonik et al., 2003], les auteurs prouvent l'indécidabilité de la satisfiabilité de la logique des ambients sans l'opérateur de noms nouveaux et sans l'opérateur de garantie. La preuve utilise une réduction du problème de l'existence d'un modèle fini pour les formule du premier ordre en le problème de satisfiabilité de la logique. Cet encodage est directement adaptable à la logique spatiale (comme nous le verrons, seul un fragment de $\mathrm{LS}_{|\mu}$ est utilisé dans cette réduction). Dans ce qui suit, nous donnons les grandes lignes de la preuve de [Charatonik et al., 2003]. Ce rappel est intéressant puisqu'il donne un exemple supplémentaire de l'expressivité de l'opérateur de composition.

Considérons l'ensemble $\mathcal{F}$ des formules de la logique du premier ordre construites sur un ensemble dénombrable de variables $x, y, z, \ldots$ et un ensemble fini de symboles de relation $\{R_1, \ldots, R_k\}$ d'arité strictement positive. Formellement, $\mathcal{F}$ est le plus petit ensemble tel que

- pour tout $R_i$ d'arité $l$, $R_i(x_1, \ldots, x_l)$ est une formule dans $\mathcal{F}$ ;
- pour tous $f, g$ formules dans $\mathcal{F}$ et toute variable $u$, $f \wedge g$, $\neg f$ et $\exists u.f$ sont aussi des formules dans $\mathcal{F}$.

Les notions de variable libre, variable liée et formule close sont comme d'habitude ; nous supposons que les variables liées d'une formule sont toutes distinctes.

Les formules de $\mathcal{F}$ sont interprétés sur des structures rationnelles ; une structure $S$ sur un domaine $\mathcal{D}$ est un ensemble d'objets de la forme $R_i(a_1, \ldots, a_l)$, où $R_i$ est un symbole de relation et $a_1, \ldots, a_l$ sont des éléments de $\mathcal{D}$. La structure $S$ est dite finie si son domaine $\mathcal{D}$ est fini.

Pour une formule $f$ et une structure $S$ de domaine $\mathcal{D}$, une valuation $\gamma$ est une application des variables libres de $f$ dans $\mathcal{D}$. La structure $S$ est *modèle* de la formule $f$ sous la valuation $\gamma$, écrit $S, \gamma \models f$, si

- $R_i(\gamma(x_1), \ldots, \gamma(x_l)) \in S$ pour $f = R_i(x_1, \ldots, x_l)$ ;
- $S, \gamma \models f'$ et $S, \gamma \models f''$ pour $f = f' \wedge f''$ ;
- $S, \gamma \not\models f'$ pour $f = \neg f'$ ;
- il existe $a \in \mathcal{D}$ tel que $S, \gamma\langle u \rightarrow a\rangle \models f'$ si $f = \exists u.f'$.

D'après le résultat bien connu de Trakhtenbrot [Trakhtenbrot, 1950], l'existence de modèle fini pour une formule du premier ordre $f$ arbitraire est indécidable.

À toute formule $f$ dans $\mathcal{F}$ on associe la formule de la logique spatiale $(\!|f|\!)$, définie récursivement sur la structure de $f$ par :

- $(\!|R(x_1, \ldots, x_l)|\!) = r_i[x_1[x_2[\ldots[x_l[\mathbf{0}]]\ldots]]] \mid \top$,
- $(\!|f \wedge f'|\!) = (\!|f|\!) \wedge (\!|f|\!)'$,
- $(\!|\neg f|\!) = \neg(\!|f|\!)$,
- $(\!|\exists x.f|\!) = \exists x.(d[x[\mathbf{0}]] \mid \top) \wedge (\!|f|\!)$.

Remarquons que les variables utilisées dans les formules de $\mathcal{F}$ sont les mêmes que les variables utilisée dans les formules logiques. Donc, $f$ et $(\!|f|\!)$ ont les mêmes variables libres.

L'idée est d'utiliser le fait que l'opérateur de composition de la logique est interprété comme une union (de multiensembles). Ainsi, un domaine fini $\mathcal{D}$ et une structure $S$ sur $\mathcal{D}$ peuvent être représentés par l'union des éléments de $\mathcal{D}$ et les relations qui sont vraies dans $S$ : chaque élément $a$ de $\mathcal{D}$ est représenté par l'élément d'arbre $d[\{\!|a[\{\!|\}\!]|\}\!]$ (l'étiquette $d$ étant réservée à cet usage) et la relation $R(a_1, \ldots, a_l)$ vraie dans $S$ est représentée par l'élément d'arbre $r[\{\!|a_1[\{\!|a_2[\ldots a_l[\{\!|\}\!]]|\}\!]]|\}\!]$ ; la structure $S$ et le domaine $\mathcal{D}$ sont représentés par l'arbre composé des éléments d'arbres correspondant aux éléments de $\mathcal{D}$ et aux relations vraies dans $S$. Alors la formule $f$ admet un modèle fini si et seulement si la formule $(\!|f|\!)$ est satisfiable.

## 5.2 Machines à deux compteurs

Une machine à deux compteurs possède deux cases mémoire $C_0$ et $C_1$ appelées compteurs qui peuvent prendre des valeurs naturelles positives et un contrôle fini qui définit la façon dont la machine peut changer d'état en incrémentant ou décrémentant un des deux compteurs.

**Définition 5.1 (Machine à deux compteurs)** Une *machine à deux compteurs* $\mathcal{M} = \langle Q, q_\mathrm{i}, q_\mathrm{f}, \Delta\rangle$ [Minsky, 1961] est un système de transitions fini où $Q$ est l'ensemble des états, $q_\mathrm{i}$ et $q_\mathrm{f}$ sont l'état initial et l'état final respectivement et $\Delta$ est un ensemble de transitions de la forme $(q, r, q')$, où $q, q' \in Q$ et $r$ est une étiquette qui appartenant à l'ensemble

$$\mathsf{Trans} = \{Inc_0, Inc_1, Dec_0, Dec_1, Zero_0, Zero_1\}.$$

Une *configuration* de la machine $\mathcal{M}$ est un triplet $(q, n_0, n_1)$, où $q$ est un état de la machine et les entiers $n_0$ et $n_1$ sont les valeurs des compteurs $C_0$ et $C_1$ respectivement.

L'ensemble de transitions $\Delta$ définit une relation de transition sur les états de la machine, notée $\vdash_{\mathcal{M}}$. On écrira $(q, n_0, n_1) \vdash_{\mathcal{M}} (q', n_0', n_1')$ si les configurations $(q, n_0, n_1)$ et $(q', n_0', n_1')$ sont en relation par $\vdash_{\mathcal{M}}$.La machine peut changer d'état suivant une transition $(q, r, q')$ dans $\Delta$ en incrémentant le compteur $C_j$ si $r = Inc_j$, en décrémentant le compteur $C_j$ si la valeur de celui-ci est positive et $r = Dec_j$ et en testant que la valeur du compteur $C_j$ est nulle si $r = Zero_j$. Formellement,

**Définition 5.2 (Relation de transition)** Pour toute machine à deux compteurs $\mathcal{M} = \langle Q, q_{\mathrm{i}}, q_{\mathrm{f}}, \Delta \rangle$, la relation de transition de $\mathcal{M}$, notée $\vdash_{\mathcal{M}}$, est la plus petite relation satisfaisant (pour tous entiers naturels $n_0, n_1$) :

$$
\begin{array}{ll}
(q, n_0, n_1) \vdash_{\mathcal{M}} (q', n_0 + 1, n_1) & \text{si } (q, Inc_0, q') \in \Delta \\
(q, n_0, n_1) \vdash_{\mathcal{M}} (q', n_0, n_1 + 1) & \text{si } (q, Inc_1, q') \in \Delta \\
(q, n_0, n_1) \vdash_{\mathcal{M}} (q', n_0 - 1, n_1) & \text{si } (q, Dec_0, q') \in \Delta \text{ et } n_0 > 0 \\
(q, n_0, n_1) \vdash_{\mathcal{M}} (q', n_0, n_1 - 1) & \text{si } (q, Dec_1, q') \in \Delta \text{ et } n_1 > 0 \\
(q, 0, n_1) \vdash_{\mathcal{M}} (q', 0, n_1) & \text{si } (q, Zero_0, q') \in \Delta \\
(q, n_0, 0) \vdash_{\mathcal{M}} (q', n_0, 0) & \text{si } (q, Zero_1, q') \in \Delta
\end{array}
$$

Nous notons $\vdash_{\mathcal{M}}^*$ la clôture réflexive et transitive de la transition $\vdash_{\mathcal{M}}$.

**Définition 5.3 (Langage)** Le *langage* de la machine à deux compteurs $\mathcal{M} = \langle Q, q_{\mathrm{i}}, q_{\mathrm{f}}, \Delta \rangle$, noté $\mathcal{L}(\mathcal{M})$, est l'ensemble d'entiers naturels $n$ pour lesquels ils existent deux entiers naturels $m_0, m_1$ tels que $(q_{\mathrm{i}}, n, 0) \vdash_{\mathcal{M}}^* (q_{\mathrm{f}}, m_0, m_1)$.

Il est bien connu que le vide du langage d'une machine à deux compteurs est indécidable, puisque tout ensemble récursivement énumérable d'entiers naturels peut être représenté comme le langage d'une machine à deux compteurs [Minsky, 1961].

## Machine à compteurs étendue

Soit $\mathcal{M} = \langle Q, q_{\mathrm{i}}, q_{\mathrm{f}}, \Delta \rangle$ une machine à deux compteurs. Nous allons montrer comment les calculs de $\mathcal{M}$ peuvent être simulés par une sorte de machine à quatre compteurs pour laquelle les seules opérations autorisés sont la décrémentation d'un compteur et le test d'égalité de deux compteurs. Intuitivement, la configuration $(q, m_0, m_1)$ de $\mathcal{M}$ est représentée par un élément de $(q, N)$ de $Q \times \mathbb{N}^4$ où le quadruplet $N = (n_0, x_0, n_1, x_1)$ est tel que $m_0 = n_0 - x_0$ et $m_1 = n_1 - x_1$. Une incrémentation du compteur $C_j$ ($j = 0, 1$) de $\mathcal{M}$ est effectuée en décrémentant $x_j$ ; une décrémentation de $C_j$ est effectuée en décrémentant $n_j$ et un test d'égalité à zéro sur le compteur $C_j$ consiste à tester si $n_j = x_j$.

Soit $\mathbb{N}_{\mathrm{c}}^4 \subseteq \mathbb{N}^4$ l'ensemble de quadruplets $(n_0, x_0, n_1, x_1)$ tels que $n_0 \geq x_0$ and $n_1 \geq x_1$. C'est à dire, $\mathbb{N}_{\mathrm{c}}^4$ est l'ensemble des quadruplets qui sont des configurations possibles d'une

machine à compteurs étendue. Les relations binaires $\mathcal{R}_{r,\Delta}$ pour tout $r$ dans Trans sont les plus petites relations sur $Q \times \mathbb{N}_c^4$ qui vérifient :

$$
\begin{array}{ll}
(q,(n_0,x_0,n_1,x_1))\ \mathcal{R}_{Inc_0,\Delta}\ (q',(n_0,x_0-1,n_1,x_1)) & \text{si } (q,Inc_0,q') \in \Delta \\
(q,(n_0,x_0,n_1,x_1))\ \mathcal{R}_{Inc_1,\Delta}\ (q',(n_0,x_0,n_1,x_1-1)) & \text{si } (q,Inc_1,q') \in \Delta \\
(q,(n_0,x_0,n_1,x_1))\ \mathcal{R}_{Dec_0,\Delta}\ (q',(n_0-1,x_0,n_1,x_1)) & \text{si } (q,Dec_0,q') \in \Delta \text{ et } n_0-x_0>0 \\
(q,(n_0,x_0,n_1,x_1))\ \mathcal{R}_{Dec_1,\Delta}\ (q',(n_0,x_0,n_1-1,x_1)) & \text{si } (q,Dec_1,q') \in \Delta \text{ et } n_1-x_1>0 \\
(q,(n_0,n_0,n_1,x_1))\ \mathcal{R}_{Zero_0,\Delta}\ (q',(n_0,n_0,n_1,x_1)) & \text{si } (q,Zero_0,q') \in \Delta \\
(q,(n_0,x_0,n_1,n_1))\ \mathcal{R}_{Zero_1,\Delta}\ (q',(n_0,x_0,n_1,n_1)) & \text{si } (q,Zero_1,q') \in \Delta
\end{array}
$$

La relation binaire $\mathcal{R}_\Delta$ est l'union des relations $\mathcal{R}_{r,\Delta}$ pour tout $r$ dans Trans et $\mathcal{R}_\Delta^*$ est sa clôture transitive et réflexive.

**Lemme 5.4** *Soit $\mathcal{M} = \langle Q, q_i, q_f, \Delta\rangle$ une machine à deux compteurs. Pour tous états $q, q'$ de $\mathcal{M}$ et pour tous entiers naturels $n_0, n_1, n_0', n_1'$, $(q,n_0,n_1) \vdash_{\mathcal{M}}^* (q',n_0',n_1')$ si et seulement s'il existe quatre entiers naturels $x_0, x_1, x_0', x_1'$ tels que $(q,(n_0+x_0,x_0,n_1+x_1,x_1))\ \mathcal{R}_\Delta^*\ (q',(n_0'+x_0',x_0',n_1'+x_1',x_1'))$.*

*Preuve* Supposons d'abord $(q,n_0,n_1) \vdash_{\mathcal{M}}^* (q',n_0',n_1')$. Alors ils existent un entier naturel $p$ et une suite $(q^0,n_0^0,n_1^0),\cdots,(q^p,n_0^p,n_1^p)$ de configurations de la machine $\mathcal{M}$ tels que pour tout $i$ dans $1..p$, $(q^{i-1},n_0^{i-1},n_1^{i-1}) \vdash_{\mathcal{M}} (q^i,n_0^i,n_1^i)$ et $q_0 = q, q^p = q', n_0^0 = n_0, n_0^p = n_0'$ et $n_1^0 = n_1, n_1^p = n_1'$. Pour tout $r$ dans Trans, soit $p_r$ le nombre de fois où la transition $r$ a été utilisée pour passer de la configuration $(q,n_0,n_1)$ à la configuration $(q',n_0',n_1')$ de la machine $\mathcal{M}$. Alors il est facile de voir que $n_0' = n_0 + p_{Inc_0} - p_{Dec_0}$ et $n_1' = n_1 + p_{Inc_1} - p_{Dec_1}$. D'autre part, par définition de $\mathcal{R}_\Delta$ il est facile de voir que $(q,(n_0+p_{Inc_0}+p_{Dec_0},p_{Inc_0}+p_{Dec_0},n_1+p_{Inc_1}+p_{Dec_1},p_{Inc_1}+p_{Dec_1}))\ \mathcal{R}_\Delta\ (q'(n_0+p_{Inc_0},p_{Dec_0},n_1+p_{Inc_1},p_{Dec_1}))$. Alors il suffit de prendre $x_0 = p_{Inc_0}+p_{Dec_0}$, $x_1 = p_{Inc_1}+p_{Dec_1}$, $x_0' = p_{Dec_0}$ et $x_1' = p_{Dec_1}$.

Supposons maintenant que $(q,(n_0+x_0,x_0,n_1+x_1,x_1))\ \mathcal{R}_\Delta^*\ (q',(n_0'+x_0',x_0',n_1'+x_1',x_1'))$. Par définition de $\mathcal{R}_\Delta$ il est facile de voir que pour tous états $q_1, q_2$ de la machine et pour tous entiers $n_0, x_0, n_1, x_1$ et $m_0, y_0, m_1, y_1$, si $(q_1,(n_0,x_0,n_1,x_1))\ \mathcal{R}_\Delta\ (q_2,(m_0,y_0,m_1,y_1))$, alors $(q_1, n_0-x_0, n_1-x_1) \vdash_{\mathcal{M}} (q_2, m_0-y_0, m_1-y_1)$. La preuve est alors une récurrence facile sur la longueur du nombre de pas pour mettre en relation $(q,(n_0+x_0,x_0,n_1+x_1,x_1))$ et $(q',(n_0'+x_0',x_0',n_1'+x_1',x_1'))$ par la clôture transitive de $\mathcal{R}_\Delta$.

□

Le quadruplet $\mathcal{M}_{\text{et}} = \langle Q, q_{\text{i}}, q_{\text{f}}, \mathcal{R}_\Delta\rangle$ est appelé machine à compteurs étendue de $\mathcal{M}$. Nous étendons le vocabulaire lié aux machines à deux compteurs aux machines à compteurs étendues correspondantes. Une *configuration* de $\mathcal{M}_{\text{et}}$ est un élément de $Q \times \mathbb{N}_c^4$ ; le *langage* de $\mathcal{M}_{\text{et}}$ est l'ensemble de quadruplets $(n+x,x,y,y) \in \mathbb{N}_c^4$ tels que ils existent des entiers $n', x', m', y'$ pour lesquels $(q_{\text{i}},(n+x,x,y,y))\ \mathcal{R}_\Delta^*\ (q_{\text{f}},(n'+x',x',m'+y',y'))$.

Comme conséquence facile du lemme 5.4, nous avons les deux corollaires suivant.

**Corollaire 5.5** *Pour toute machine à deux compteurs $\mathcal{M}$ et pour tout entier naturel $n$, $n$ appartient au langage de $\mathcal{M}$ si et seulement s'il existent deux entiers naturels $x, y$ tels que $(n+x,x,y,y)$ appartient au langage de $\mathcal{M}_{et}$.*

**Corollaire 5.6** *Pour toute machine à deux compteurs $\mathcal{M}$, le langage de $\mathcal{M}$ est vide si et seulement si le langage de $\mathcal{M}_{et}$ est vide.*

**Remarque 5.7** *De la façon dont une machine à deux compteurs a été définie, il est facile de voir que si le quadruplet $(n+x, x, y, y)$ appartient au langage d'une machine $\mathcal{M}_{et}$, alors $(n+x+1, x+1, y, y)$ et $(n+x, x, y+1, y+1)$ appartiennent également à ce langage.*

## 5.3 Preuve d'indécidabilité

Cette section est consacrée à la preuve d'indécidabilité de la satisfiabilité du fragment de la logique sans quantification. Plus précisément, nous allons construire un système d'équations $\mathcal{S}_{\mathcal{M}}$ tel que l'ensemble $Sol_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \mathsf{dern}(\mathcal{S}_{\mathcal{M}}))$ est vide si et seulement si le langage d'une machine à deux compteurs $\mathcal{M}$ est vide. A proprement parler, nous allons utiliser une machine à deux compteurs étendue. L'idée de cette preuve s'inspire de la preuve d'indécidabilité du vide des automates ACU alternants [Goubault-Larrecq and Verma, 2002], [Verma, 2003].

Soit $\mathcal{M} = \langle Q, q_{\mathrm{i}}, q_{\mathrm{f}}, \Delta \rangle$ une machine à deux compteurs que nous considérons fixée pour la suite de la section. Pour tout état $q$ de la machine, nous notons par $Acc_{\mathcal{M}}(q)$ l'ensemble des quadruplets $s \in \mathbb{N}_{\mathrm{c}}^4$ qui correspondent aux configurations de $\mathcal{M}_{\mathrm{et}}$ « acceptables » à partir de l'état $q$, càd

$$Acc_{\mathcal{M}}(q) = \{s \in \mathbb{N}_{\mathrm{c}}^4 \mid \exists s' \in \mathbb{N}_{\mathrm{c}}^4,\ (q,s)\ \mathcal{R}_{\Delta}^{*}\ (q_{\mathrm{f}}, s')\}.$$

**Lemme 5.8** *Pour toute machine à deux compteurs $\mathcal{M} = \langle Q, q_i, q_f, \Delta \rangle$, le langage de la machine à compteurs étendue $\mathcal{M}_{et}$ est $Acc_{\mathcal{M}}(q_i) \cap \{(n, x, m, m) \mid n, x, m \in \mathbb{N}\}$.*

*Preuve* Conséquence immédiate des définitions.

□

Les éléments de $\mathbb{N}_{\mathrm{c}}^4$ peuvent être encodés par des arbres de façon assez naturelle. Considérons les quatre étiquettes distinctes $a_0, b_0, a_1, b_1 \in \Sigma$. Pour tout quadruplet $\mathbf{n} = (n_0, x_0, n_1, x_1)$ dans $\mathbb{N}_{\mathrm{c}}^4$, soit $(\!|\mathbf{n}|\!)$ l'arbre

$$(\!|(n_0, x_0, n_1, x_1)|\!) = n_0 \cdot a_0[\{\!\}] \uplus x_0 \cdot b_0[\{\!\}] \uplus n_1 \cdot a_1[\{\!\}] \uplus x_1 \cdot b_1[\{\!\}].$$

Pour un sous-ensemble $N$ de $\mathbb{N}_{\mathrm{c}}^4$, $(\!|N|\!)$ est l'ensemble d'arbres $\{(\!|\mathbf{n}|\!) \mid \mathbf{n} \in N\}$.

Soit le système d'équations $\mathcal{S}_{\mathcal{M}}$ sur l'ensemble de variables

$$\{\xi_{a0*}, \xi_{a1*}, \xi_{\mathrm{ok0}}, \xi_{\mathrm{ok1}}, \xi_{\mathrm{ok}}, \xi_{\mathrm{zero0}}, \xi_{\mathrm{zero1}}, \xi\} \cup \{\xi_q \mid q \in Q\}.$$

Les sept premières équations de $\mathcal{S}_{\mathcal{M}}$ sont :

$$\begin{aligned}
\xi_{a0*} &\stackrel{\mu}{=} a_0[\mathbf{0}] \mid \xi_{a0*} \vee \mathbf{0} \\
\xi_{a1*} &\stackrel{\mu}{=} a_1[\mathbf{0}] \mid \xi_{a1*} \vee \mathbf{0} \\
\xi_{\mathrm{ok}0} &\stackrel{\mu}{=} a_0[\mathbf{0}] \mid b_0[\mathbf{0}] \mid \xi_{\mathrm{ok}0} \vee \xi_{a0*} \\
\xi_{\mathrm{ok}1} &\stackrel{\mu}{=} a_1[\mathbf{0}] \mid b_1[\mathbf{0}] \mid \xi_{\mathrm{ok}1} \vee \xi_{a1*} \\
\xi_{\mathrm{ok}} &\stackrel{\mu}{=} \xi_{\mathrm{ok}0} \mid \xi_{\mathrm{ok}1} \\
\xi_{\mathrm{zero}0} &\stackrel{\mu}{=} a_0[\mathbf{0}] \mid b_0[\mathbf{0}] \mid \xi_{\mathrm{zero}0} \vee \xi_{\mathrm{ok}1} \\
\xi_{\mathrm{zero}1} &\stackrel{\mu}{=} a_1[\mathbf{0}] \mid b_1[\mathbf{0}] \mid \xi_{\mathrm{zero}1} \vee \xi_{\mathrm{ok}0}
\end{aligned}$$

Les équations suivantes sont pour les variables $\xi_q$ pour $q$ dans $Q$. Pour chaque état $q$, l'équation pour la variable $\xi_q$ est :

$$\xi_q \stackrel{\mu}{=} \bigvee_{(q,r,q') \in \Delta} \mathsf{Pred}(\xi_{q'}, r)$$

où pour tout $r$ dans Trans, $\mathsf{Pred}(\xi', r)$ est donné par

$$\begin{aligned}
\mathsf{Pred}(\xi', \mathit{Inc}_j) &= (\xi' \mid b_j[\mathbf{0}]) \wedge \xi_{\mathrm{ok}} \\
\mathsf{Pred}(\xi', \mathit{Dec}_j) &= (\xi' \mid a_j[\mathbf{0}]) \wedge \xi_{\mathrm{ok}} \\
\mathsf{Pred}(\xi', \mathit{Zero}_j) &= \xi' \wedge \xi_{\mathrm{zero}j}
\end{aligned}$$

La dernière équation du système $\mathcal{S}_{\mathcal{M}}$ est

$$\xi \stackrel{\mu}{=} \xi_{q_{\mathrm{i}}} \wedge \xi_{\mathrm{zero}1}.$$

Le lemme qui suit précise le rôle des sept premières équations de $\mathcal{S}_{\mathcal{M}}$.

**Lemme 5.9** *Dans le système d'équations $\mathcal{S}_{\mathcal{M}}$,*
- $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi_{ok}) = (\!|\mathbb{N}^4_c|\!)$ ;
- $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi_{zero0}) = \{(\!|(n, n, m, y)|\!) \mid n, m, y \in \mathbb{N}, m \geq y\}$ ;
- $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi_{zero1}) = \{(\!|(n, x, m, m)|\!) \mid n, x, m \in \mathbb{N}, n \geq x\}$.

C'est à dire, $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi_{\mathrm{ok}})$ est l'ensemble des arbres qui sont des encodages d'une configuration correcte de la machine étendue et $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi_{\mathrm{zero}i})$ est l'ensemble des arbres qui sont des encodages corrects de la machine étendue et dans laquelle la valeur du $i^{\text{ème}}$ compteur (de $\mathcal{M}$) est zéro, pour $i$ étant 0 ou 1.

*Preuve* Considérons le système $\mathcal{S}'_{\mathcal{M}}$ composé par les sept premières équations de $\mathcal{S}_{\mathcal{M}}$ et $\mathcal{S}''_{\mathcal{M}}$ le système composé par les équations restantes. D'après la proposition 4.4, et en remarquant qu'aucune des variables de $\mathcal{S}''_{\mathcal{M}}$ n'apparaît libre dans $\mathcal{S}'_{\mathcal{M}}$, nous déduisons que les solutions de $\mathcal{S}'_{\mathcal{M}}$ et $\mathcal{S}_{\mathcal{M}}$ coïncident pour les variables $\xi_{\mathrm{ok}0}, \xi_{\mathrm{ok}1}, \xi_{\mathrm{ok}}, \xi_{\mathrm{zero}0}, \xi_{\mathrm{zero}1}, \xi$. Donc, il suffit de montrer la propriété pour $\mathcal{S}'_{\mathcal{M}}$.

Il est facile de voir que pour les deux premières équations de $\mathcal{S}'_{\mathcal{M}}$, les solutions correspondants sont $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}, \xi_{a0*}) = \{n.a_0[\{\!|\!\}] \mid n \in \mathbb{N}\}$ et $\mathit{Sol}_{\mathcal{T}}(\mathcal{S}, \xi_{a1*}) = \{n.a_1[\{\!|\!\}] \mid n \in \mathbb{N}\}$. De ce

fait on déduit facilement que pour les deux équations qui suivent, les solutions correspondantes sont $Sol_{\mathcal{T}}(\mathcal{S}, \xi_{\text{ok0}}) = \{n.a_0[\{\!|\}] \uplus (n+x).b_0[\{\!|\}] \mid n, x \in \mathbb{N}\}$ et $Sol_{\mathcal{T}}(\mathcal{S}, \xi_{\text{ok1}}) = \{n.a_1[\{\!|\}] \uplus (n+x).b_1[\{\!|\}] \mid n, x \in \mathbb{N}\}$. Le lemme suit immédiatement de ces deux propriétés et de la définition de $\mathbb{N}^4_c$.

□

Les équations pour les variables $\xi_q$ (pour $q$ un état de la machine $\mathcal{M}$) définissent les ensembles de configurations acceptables dans l'état correspondant.

**Lemme 5.10** *Pour tout état $q$ de la machine $\mathcal{M}$, dans le système d'équations $\mathcal{S}_{\mathcal{M}}$,*

$$Sol_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi_q) = (\!|Acc_{\mathcal{M}}(q)|\!).$$

*Preuve* Soient $S_{a0*}$, $S_{a1*}$, $S_{\text{ok0}}$, $S_{\text{ok1}}$, $S_{\text{ok}}$, $S_{\text{zero0}}$ et $S_{\text{zero1}}$ les solutions des sept premières équations du système $\mathcal{S}_{\mathcal{M}}$. Supposons que l'ensemble d'états de la machine $\mathcal{M}$ est $\{q_1, \ldots, q_n\}$. Soit $\mathcal{S}'$ le système d'équations à points fixes obtenu comme la restrictions de $\mathcal{S}_{\mathcal{M}}$ aux équations pour les variables $\xi_{q_1}, \ldots, \xi_{q_n}$ et dans lesquelles les variables $\xi_{a0*}$, $\xi_{a1*}$, $\xi_{\text{ok0}}$, $\xi_{\text{ok1}}$, $\xi_{\text{ok}}$, $\xi_{\text{zero0}}$ et $\xi_{\text{zero1}}$ ont été remplacées par la solution correspondante. Remarquons que $\mathcal{S}'$ est un système clos dans le sens que toutes les variables apparaissant en partie droite d'équation sont définies par une équation de $\mathcal{S}'$. Pour tout $q_i$ état de la machine, notons $f_{q_i}$ la partie droite de l'équation pour $\xi_{q_i}$ dans $\mathcal{S}'$. Considérons le système d'équations

$$\begin{aligned} \xi_{q_1} &= f_{q_1}(\xi_{q_1}, \ldots, \xi_{q_n}) \\ &\ldots \\ \xi_{q_n} &= f_{q_n}(\xi_{q_1}, \ldots, \xi_{q_n}) \end{aligned} \tag{5.1}$$

Soit $\langle S_{q_1}, \ldots, S_{q_n} \rangle$ la plus petite solution de (5.1).[1] Il est clair par construction de $\mathcal{S}_{\mathcal{M}}$ et de $\mathcal{S}'$ et du lemme 5.9 que pour tout $q$ état de la machine, $S_q = (\!|Acc_{\mathcal{M}}(q)|\!)$. La conclusion suit du fait que les solutions des systèmes (5.1) et $\mathcal{S}'$ coïncident (voir la section 4.1.1).

□

Comme conséquence immédiate des deux lemmes précédents, nous avons

**Lemme 5.11** *Dans le système d'équations $\mathcal{S}_{\mathcal{M}}$,*

$$Sol_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi) = (\!|Acc_{\mathcal{M}}(q_i) \cap \{(n, x, m, m) \mid n, x, m \in \mathbb{N}\}|\!).$$

*Preuve* Par le lemme 5.4, nous savons que $\{(n, x, m, m) \mid n, x, m \in \mathbb{N}\}$ est la solution de $\mathcal{S}_{\mathcal{M}}$ pour la variable $\xi_{\text{zero1}}$ et par le lemme 5.10 nous savons que $(\!|Acc_{\mathcal{M}}(q_i)|\!)$ est la solution de $\mathcal{S}_{\mathcal{M}}$ pour la variable $\xi_{q_i}$, d'où la conclusion se déduit immédiatement.

□

Rappelant que, par le lemme 5.8, $Acc_{\mathcal{M}}(q) \cap \{(n, x, m, m) \mid n, x, m \in \mathbb{N}\}$ est exactement le langage de la machine à compteurs étendue $\mathcal{M}_{\text{et}}$, et que par le corollaire 5.6, la langage de $\mathcal{M}_{\text{et}}$ est vide si et seulement si le langage de $\mathcal{M}$ est vide, nous déduisons du précédent

[1] Il s'agit ici de la solution extrême d'un système d'équations sur un treillis produit et non de la solution d'un système d'équations à points fixes ; voir la section 4.1.1 pour la distinctions entre les deux.

lemme que dans le système d'équations $\mathcal{S}_{\mathcal{M}}$, $Sol_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi)$ est l'ensemble vide si et seulement si $\mathcal{L}(\mathcal{M})$ est vide. Or, $\xi$ étant la dernière variable de $\mathcal{S}_{\mathcal{M}}$, nous savons que $Sol_{\mathcal{T}}(\mathcal{S}_{\mathcal{M}}, \xi)$ est égal à l'interprétation de la formule $\mathsf{formule}(\mathcal{S}_{\mathcal{M}})$, d'où le principal théorème de cette section :

**Théorème 5.12** *La satisfiabilité de la logique spatiale sans quantification est indécidable.*

Ce résultat auxiliaire qui découle de la preuve du théorème 5.12 sera utilisé dans la section suivante.

**Lemme 5.13** *Pour toute machine à deux compteurs $\mathcal{M}$, il existe une formule $\phi_{\mathcal{M}}$ telle que*

$$[\![\phi_{\mathcal{M}}]\!] = \left\{ (n+x) \cdot a_0[\{\!\}] \uplus x \cdot b_0[\{\!\}] \uplus y \cdot a_1[\{\!\}] \uplus y \cdot b_1[\{\!\}] \;\middle|\; \begin{array}{c} (n+x, x, y, y) \text{ dans} \\ \text{le langage de } \mathcal{M}_{et} \end{array} \right\},$$

*où $a_0, b_0, a_1, b_1$ sont quatre étiquettes distinctes.*

## 5.4 Comparaison de l'expressivité des logiques $\mathrm{LS}_{|\exists}$ et PMSO

Nous avons montré dans la section précédente que la logique $\mathrm{LS}_{|\exists}$ n'est pas décidable. Nous savons par ailleurs que la logique PMSO est décidable. Dans ce qui suit, nous allons montrer que $\mathrm{LS}_{|\exists}$ permet effectivement de définir des ensembles d'arbres qui ne sont pas PMSO-définissables. Plus tard, le chapitre 7 nous allons montrer que la logique PMSO est « incluse » dans $\mathrm{LS}_{|\exists}$, de part son expressivité ; PMSO est équivalente à un fragment de $\mathrm{LS}_{|\exists}$. Combinant ces deux résultats, nous pourrons conclure que la logique $\mathrm{LS}_{|\exists}$ est strictement plus expressive que PMSO.

Nous montrons dans la proposition 5.14 qu'il existe une formule $\phi$ telle que l'ensemble d'arbres $[\![\phi]\!]$ n'est pas PMSO définissable. Plus précisément, cette formule est la formule $\phi_{\mathcal{M}}$ définie dans le lemme 5.13 ci-dessus ; càd $[\![\phi_{\mathcal{M}}]\!]$ est en quelque sorte une représentation du langage de la machine à deux compteurs $\mathcal{M}$. L'idée est la suivante : nous allons montrer qu'il existe une machine à deux compteurs $\mathcal{M}$ telle que $[\![\phi_{\mathcal{M}}]\!]$ n'est le langage d'aucun automate automates à contraintes numériques semilinéaire. Nous utiliserons ensuite le fait que les ensembles d'arbres reconnaissables par un ACN semilinéaire sont exactement les ensembles d'arbres PMSO définissables.

**Proposition 5.14** *Il existe une formule $\phi$ de la logique $LS_{|\exists}$ telle que $[\![\phi]\!]$ n'est pas un ensemble d'arbres PMSO définissable.*

*Preuve* La preuve se fait par contradiction. Supposons que pour toute formule $\phi$, l'ensemble d'arbres $[\![\phi]\!]$ est PMSO définissable. Donc, par le théorème 3.2 nous savons que $[\![\phi]\!]$ est le langage d'un ACN semilinéaire. Considérons maintenant une machine à deux compteurs arbitraire $\mathcal{M}$ et soit $\phi_{\mathcal{M}}$ la formule telle que, d'après le lemme 5.13, $[\![\phi_{\mathcal{M}}]\!]$ est l'ensemble d'arbres

$$\{(n+x).a_0[\{\!\}] \uplus x.b_0[\{\!\}] \uplus y.a_1[\{\!\}] \uplus y.b_1[\{\!\}] \mid (n+x, x, y, y) \text{ dans le langage de } \mathcal{M}_{\text{et}}\},$$

où $a_0, b_0, a_1, b_1$ sont quatre étiquettes distinctes. D'après les hypothèses, nous savons que $[\![\phi_{\mathcal{M}}]\!]$ est le langage d'un ACN semilinéaire. Soit donc $A$ un ACN semilinéaire déterministe tel que $\mathcal{L}(A) = [\![\phi_{\mathcal{M}}]\!]$ ; par le lemme 2.26 nous savons que cet automate déterministe existe toujours.

Nous commençons par montrer qu'on peut considérer que l'automate $A$ a une forme particulière, telle que décrite dans le fait qui suit.

**Fait 5.1** L'ensemble d'arbres $[\![\phi_{\mathcal{M}}]\!]$ est reconnu par un ACN semilinéaire si et seulement s'il est reconnu par un ACN semilinéaire déterministe $A = (\Sigma, Q, \Delta, F)$ tel que :

(i) toute transition de $A$ est de la forme $(q, \alpha, 0^Q)$ (pour un certain $q \in Q$ et $\alpha \subseteq \Sigma$) ;

(ii) toute transition de $A$ est de la forme $(q, \alpha, 0^Q)$, avec $\alpha$ étant un ensemble d'étiquettes non vide et inclus dans $\{a_0, b_0, a_1, b_1\}$ (pour un certain $q \in Q$) ;

(iii) pour tout $q$ état de l'automate, l'ensemble de transitions $\Delta$ contient au plus une transition de la forme $(q, \alpha, 0^Q)$ ;

(iv) pour toutes transitions $(q, \alpha, 0^Q)$ et $(q', \alpha', 0^Q)$ dans $\Delta$, si $q \neq q'$, alors les ensembles d'étiquettes $\alpha$ et $\alpha'$ sont disjoints.

La direction de la droite vers la gauche est triviale.

Pour la direction de la gauche vers la droite, soit $A' = (\Sigma, Q, \Delta', F)$ un ACN semilinéaire tel que $\mathcal{L}(A') = [\![\phi_{\mathcal{M}}]\!]$. Par le lemme 2.26, nous pouvons supposer sans perte de généralité que $A'$ est déterministe.

Pour la preuve du point (i), nous montrons simplement que si $(q, \alpha, D)$ est une transition dans $\Delta'$ et $D \neq 0^Q$, alors cette transition peut être enlevée de $\Delta'$ sans que cela ne modifie le langage de l'automate $A'$. Soit donc $(q, \alpha, D)$ dans $\Delta'$ et $D \neq 0^Q$. Si $D = \varnothing$ ou, de façon plus générale $\mathcal{L}_{A'}(D)$ est vide, alors cette transition peut être enlevée puisqu'elle ne contribue en rien à $\mathcal{L}_{A'}(q)$. Si $\mathcal{L}_{A'}(D)$ n'est pas vide, alors $\mathcal{L}_{A'}(q)$ contient des arbres de hauteur supérieure strictement à un. Or, nous savons par définition que $\mathcal{L}(A')$ contient uniquement des arbres de hauteur un. Alors l'état $q$ ne contribue pas à la définition du langage de l'automate et toute transition le concernant peut être enlevée. [2] Donc, l'automate $A = (\Sigma, Q, \Delta, F)$ satisfait la condition du point (i), où $\Delta$ est la relation de transition définie par :

$$\Delta = \Delta' \setminus \{(q, \alpha, D) \mid D \neq 0^Q\}.$$

Il est facile de voir que l'automate $A$ est à contraintes semilinéaires et déterministe. Ceci termine la preuve du point (i).

Pour le point (ii), soit $A' = (\Sigma, Q, \Delta', F)$ un ACN semilinéaire déterministe tel que $\mathcal{L}(A') = [\![\phi_{\mathcal{M}}]\!]$ et tel que toute transition de $A'$ est de la forme $(q, \alpha, 0^Q)$. Par le point (i), nous savons que l'automate $A'$ existe toujours. Nous allons montrer que si $(q, \alpha, 0^Q)$ est une transition de $A'$ et $\alpha$ n'est pas inclus dans $\{a_0, b_0, a_1, b_1\}$ ou $\alpha$ est l'ensemble vide, alors cette transition peut être enlevée sans que cela ne modifie le langage de l'automate $A'$. Si $\alpha = \varnothing$, alors cette transition ne contribue en rien à $\mathcal{L}_{A'}(q)$ et peut donc être enlevée. Si $\alpha \neq \varnothing$ mais $\alpha$ contient une étiquette $c$ qui n'est pas parmi $\{a_0, b_0, a_1, b_1\}$, alors nécessairement $\mathcal{L}(q)$ contient l'arbre $c[\{\!\}]$. Or, par définition nous savons que les arbres dans $\mathcal{L}(A')$ contiennent uniquement les étiquettes parmi $\{a_0, b_0, a_1, b_1\}$, ce qui signifie que l'état $q$ ne contribue pas à la définition du langage de l'automate et toute transition le concernant peut être enlevée. Donc, l'automate $A = (\Sigma, Q, \Delta, F)$ satisfait la condition du point (i), où $\Delta$ est la relation de transition définie par :

$$\Delta = \Delta' \setminus \{(q, \alpha, 0^Q) \mid \alpha = \varnothing \text{ ou } \alpha \not\subseteq \{a_0, b_0, a_1, b_1\}\}.$$

[2] Plus précisément, pour tout $\mathbf{d}$ dans $F$, $\mathbf{d}(q) = 0$, ce qui signifie que l'état $q$ de l'automate est inutile.

Il est facile de voir que l'automate $A$ est à contraintes semilinéaires et déterministe. Ceci termine la preuve du point (ii).

Pour le point (iii), si $(q, \alpha, 0^Q)$ et $(q, \alpha', 0^Q)$ sont deux transitions différentes de $\Delta$, il suffit de les remplacer par l'unique transition $(q, \alpha \cup \alpha', 0^Q)$. Il est facile de voir que ce remplacement ne modifie pas le langage de $A$ et que $A$ reste un automate à contraintes semilinéaires et déterministe.

Le point (iv) est conséquence directe du fait que l'automate $A$ est déterministe. En effet, si $(q, \alpha, 0^Q)$ et $(q', \alpha', 0^Q)$ sont deux transitions de l'automate et $q \neq q'$ et $c \in \alpha \cap \alpha'$, alors l'élément d'arbre $c[\{\!\}]$ appartient à $\mathcal{L}_A(q)$ et à $\mathcal{L}_A(q')$, et donc $A$ n'est pas un automate déterministe. Ceci termine la preuve du point (iv) et du fait 5.1.

Soit maintenant $A' = (\Sigma, Q', \Delta', F')$ un automate à contraintes numériques déterministe tel que $\mathcal{L}(A') = [\![\phi_{\mathcal{M}}]\!]$ et qui satisfait les contraintes annoncées dans le fait 5.1. Considérons l'automate $A = (\Sigma, Q, \Delta, F)$ défini par : l'ensemble d'états $Q$ est $\{q_{a_0}, q_{b_0}, q_{a_1}, q_{b_1}\}$ ; la relation de transition $\Delta$ est

$$\Delta = \{(q_{a_0}, \{a_0\}, 0^Q), (q_{b_0}, \{b_0\}, 0^Q), (q_{a_1}, \{a_0\}, 0^Q), (q_{a_1}, \{b_1\}, 0^Q)\}. \tag{5.2}$$

Finalement, la condition d'acceptation $F$ est définie par :

$$F = \bigcup_{\mathbf{d} \in F'} H_{\mathbf{d}}$$

où pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^{Q'}$, l'ensemble $H_{\mathbf{d}} \subseteq \mathbb{N}^Q$ est défini par :

$$H_{\mathbf{d}} = \left\{ \mathbf{h} \;\middle|\; \begin{array}{l} \text{pour tout } (q, \alpha, 0^{Q'}) \text{ dans } \Delta', \left(\sum_{c \in \alpha} \mathbf{h}(c)\right) = \mathbf{d}(q) \text{ et} \\ \text{pour tout } c \text{ dans } \beta, \mathbf{h}(q_c) = 0 \end{array} \right\}$$

où $\beta \subseteq \{a_0, b_0, a_1, b_1\}$ désigne l'ensemble d'étiquettes qui n'apparaissent dans aucune transition de l'automate $A'$. Formellement, $\beta$ est l'ensemble des étiquettes $c$ telles qu'ils n'existent pas $q \in Q'$ et $\alpha$ avec $c \in \alpha$ et $(q, \alpha, 0^{Q'})$ transition dans $\Delta'$.

Il n'est pas difficile de voir que l'ensemble $F$ est semilinéaire (en utilisant que, par hypothèse, $F'$ est un ensemble semilinéaire). En effet, étant donné une formule de l'arithmétique de Presburger définissant $F'$, on peut construire une formule de l'arithmétique de Presburger définissant $F$. Il n'est pas non plus difficile de voir que le langage reconnu par l'automate $A$ est le même que le langage reconnu par l'automate $A'$.

Nous en déduisons que pour toute machine à deux compteurs $\mathcal{M}$, il existe un automate à contraintes numériques semilinéaires $A = (\Sigma, Q, \Delta, F)$ tel que l'ensemble d'états $Q$ est $\{q_{a_0}, q_{b_0}, q_{a_1}, q_{b_1}\}$ et la relation de transition $\Delta$ est celle définie dans (5.2) et tel que $\mathcal{L}(A) = [\![\phi_{\mathcal{M}}]\!]$. Rappelons que, par définition, $[\![\phi_{\mathcal{M}}]\!]$ est l'ensemble d'arbres

$$\{(n+x)\cdot a_0[\{\!\}] \uplus x \cdot b_0[\{\!\}] \uplus y \cdot a_1[\{\!\}] \uplus y \cdot b_1[\{\!\}] \mid (n+x, x, y, y) \text{ dans le langage de } \mathcal{M}_{\text{et}}\}.$$

Par définition de l'automate $A$, on peut voir qu'alors l'ensemble de multiensemble $F$ est

$$\{(q_{a_0} \to n,\ q_{b_0} \to x,\ q_{a_1} \to y,\ q_{b_1} \to y) \mid (n + x, x, y, y) \text{ dans le langage de } \mathcal{M}_{\text{et}}\}.$$

Maintenant, comme l'automate $A$ est semilinéaire, nous savons que l'ensemble $F$ est semilinéaire. D'autre part, nous savons que les ensembles semilinéaires sont clos par projection.

Considérons donc $F_1$, l'ensemble de multiensembles sur $\{q_{a_0}, q_{b_0}\}$ obtenu en projetant $F$ sur les composantes correspondant à $q_{a_0}$ et $q_{b_0}$ :

$$F_1 = \{(q_{a_0} \to n,\ q_{b_0} \to x) \mid (q_{a_0} \to n,\ q_{b_0} \to x,\ q_{a_1} \to y,\ q_{b_1} \to y) \text{ dans } F\}.$$

Alors $F_1$ est un ensembles de multiensembles semilinéaire. Nous savons également que les ensembles semilinéaires sont clos par somme et soustraction des composantes. [3] Considérons $F_2$, l'ensemble de multiensembles sur $\{(q_{a_0}, q_{b_0})\}$, obtenu de $F_1$ en soustrayant la composante correspondant à $q_{b_0}$ de la composante correspondant à $q_{a_0}$ :

$$F_2 = \{((q_{a_0}, q_{b_0}) \to n - x) \mid (q_{a_0} \to n,\ q_{b_0} \to x) \mid \text{ dans } F_1\}.$$

Donc l'ensemble $F_2$ est semilinéaire et, par conséquent, l'ensemble

$$N = \{n \mid ((q_{a_0}, q_{b_0}) \to n) \text{ dans } F_2\}$$

est un ensemble d'entiers naturels semilinéaire. Finalement, par le corollaire 5.5 et par définition de $F_2$, nous pouvons facilement déduire $N$ est exactement le langage de la machine à deux compteurs $\mathcal{M}$. Rappelons que $\mathcal{M}$ est une machine à deux compteurs arbitraire ; nous déduisons donc que pour toute machine à deux compteurs $\mathcal{M}$, le langage de $\mathcal{M}$ est un ensemble d'entiers naturels semilinéaire, ce qui est évidemment faux. Cette contradiction vient de l'hypothèse initiale que pour toute machine à deux compteurs $\mathcal{M}$, l'ensemble d'arbres $[\![\phi_{\mathcal{M}}]\!]$ est PMSO-définissable. Nous en concluons qu'il existe une formule $\phi$ de la logique $LS_{|\exists}$ telle que l'ensemble d'arbres $[\![\phi]\!]$ n'est pas PMSO-définissable.

□

# Conclusion

Nous avons établi dans cette section deux résultats importants. Tout d'abord, nous avons montré que la logique spatiale est indécidable même en absence de quantification. La preuve se fait en encodant le vide du langage d'une machine à deux compteurs, suivant des idées présentées dans [Goubault-Larrecq and Verma, 2002], [Verma, 2003] utilisés pour montrer l'indécidabilité du vide des automates ACU alternats. Il est à remarquer ici que cet encodage n'utilise pas toute la puissance de la logique : notamment, la conjonction est utilisée, mais pas la négation, et la récursion se fait uniquement en « largeur », càd les variables de récursion n'apparaissent jamais sous un opérateur d'emboîtement. D'ailleurs, les arbres utilisés pour l'encodage des états de la machine à deux compteurs sont des arbres plats, de hauteur un. Ensuite nous avons utilisé le résultat d'indécidabilité pour montrer que la logique $LS_{|\exists}$ est strictement plus expressive que la logique PMSO qui, elle, a un problème de satisfiabilité décidable.

[3] Dans le cas général, la soustraction de composantes peut générer des nombres négatifs. Sans rentrer dans les détails, un ensemble semilinéaire de vecteurs peut être défini sur les vecteurs de nombres rationnels. La restriction d'un tel ensemble aux vecteurs d'entiers naturels est alors un ensemble semilinéaire. De ce fait, la soustraction ne nous fait pas vraiment sortir du cadre des ensembles semilinéaires. Notons pour finir que dans notre cas de figure, la soustraction faite $n - x$ satisfait toujours que $n \geq x$, et donc les ensembles générés sont bien des ensembles semilinéaires sur les entiers naturels tels que nous les considérons partout dans cet ouvrage.

# Chapitre 6

# Automates d'arbres pour la logique sans quantification

Dans ce chapitre nous montrons que pour toute formule de la logique $\mathrm{LS}_{|\exists}$ on peut construire un automate à contraintes numériques équivalent à cette formule, c'est à dire un automate dont le langage est équivalent à l'interprétation de cette formule. Pour définir un tel automate, nous allons utiliser les systèmes d'équations simplifiés. En effet, comme nous l'avons vu dans le chapitre précédent, toute formule logique peut être représentée par un système d'équations simplifié. Nous commençons par présenter brièvement dans la section 6.1 comment dans [Dal Zilio et al., 2004] des automates d'arbres semblables aux ACN ont été définis pour une logique qui s'avère être un fragment de la logique spatiale. Ensuite, dans la section 6.2 nous définissons l'automate $A_{\mathcal{S}}$ associé au système d'équations simplifié $\mathcal{S}$. Dans le section 6.3 nous montrons que cette définition est correcte, càd que l'automate $A_{\mathcal{S}}$ défini dans la section 6.2 est effectivement équivalent à la formule $\phi$ telle que $\mathcal{S}$ est un système d'équations simplifié pour $\phi$. Dans la dernière section 6.4 nous présentons un algorithme pour le test d'appartenance d'un arbre au langage d'un automate lorsque celui-ci est l'automate équivalent à une formule de la logique $\mathrm{LS}_{|\exists}$.

## 6.1 La logique et les automates à faisceaux

Une première définition d'automates d'arbres pour une logique basée sur la logique des ambients (avec opérateur de composition) a été faite dans une suite de travaux par Dal Zilio, Lugiez et Meyssonnier [Lugiez and Dal Zilio, 2002], [Dal Zilio and Lugiez, 2003], [Dal Zilio et al., 2004]. L'approche proposée dans [Dal Zilio et al., 2004] a fortement inspiré la mise en correspondance entre la logique spatiale et les automates d'arbres à contraintes numériques que nous faisons ici. Dans ce qui suit, nous présentons brièvement l'approche et les résultats de ce travail.

L'idée initiale est qu'un arbre (un multiensemble) est un produit de la forme

$$\langle \text{vecteur de cardinalités} \rangle \cdot \langle \text{vecteur d'éléments d'arbre} \rangle.$$

Par exemple, l'arbre plat ayant quatre arêtes étiquetées par $a$ et deux arêtes étiquetées par $b$ est représenté par le produit $(4,2) \cdot (a[\{\!|\}], b[\{\!|\}])$. Ce type de produit est étendu aux ensembles

d'arbres par

$$\langle \text{ensemble de vecteurs de cardinalités} \rangle \cdot \langle \text{vecteur d'ensembles d'éléments d'arbres} \rangle$$

Par exemple, l'ensemble d'arbres plats ayant deux fois plus d'arêtes étiquetées par $a$ que d'étiquettes étiquetées par $b$ est représenté par le produit $\{(2n, n) \mid n \in \mathbb{N}\} \cdot (\{a[\{\!\}]\}, \{b[\{\!\}]\})$. L'ensemble d'arbres ayant deux fois plus d'arêtes étiquetées par $a$ que d'arêtes étiquetées par $b$ sous la racine est représenté par le produit $\{(2n, n) \mid n \in \mathbb{N}\} \cdot (\{a\}[\mathcal{A}_\Sigma], \{b\}[\mathcal{A}_\Sigma])$. (Rappelons que $\alpha[T]$ est l'ensemble d'arbres $\{a[t] \mid a \in \alpha, t \in T\}$.) On voit immédiatement le lien avec les automates à contraintes numériques : dans un automate à contraintes numériques qui reconnaîtrait ce même ensemble d'arbres. Dans ce cas, les ensembles d'éléments d'arbres $\{a\}[\mathcal{A}_\Sigma]$ et $\{b\}[\mathcal{A}_\Sigma]$ seraient les langages des états de l'automate, et l'ensemble de vecteurs $\{(2n, n) \mid n \in \mathbb{N}\}$ correspondrait à la condition d'acceptation.[1]

Dans [Dal Zilio et al., 2004], les auteurs s'intéressent à un fragment de la logique des ambients, appelé TL (pour *tree logic*), et introduisent une classe d'automates qui capture cette logique. Une formule de la logique TL, que nous notons $\psi$, est définie par la syntaxe suivante :

| | | |
|---|---|---|
| $\psi ::=$ | $\mathbf{0}$ | arbre vide |
| | $a[\psi]$ | imbrication ou emboîtement |
| | $\psi \mid \psi$ | composition |
| | $\psi \rhd \psi$ | adjonction |
| | $\psi*$ | itération |
| | $\top$ | vérité |
| | $\neg\psi$ | négation |
| | $\psi \vee \psi$ | disjonction |

Les formules de TL sont interprétées sur les arbres non ordonnés d'arité non bornée, l'interprétation de la formule $\psi$ est notée $[\![\psi]\!]$ et est un ensemble d'arbres défini par :

$$\begin{array}{lcl}
[\![\mathbf{0}]\!] & = & \{\{\!\}\} \\
[\![a[\psi]]\!] & = & \{a\}[[\![\psi]\!]] \\
[\![\psi \mid \psi']\!] & = & [\![\psi]\!] \mid [\![\psi']\!] \\
[\![\psi \rhd \psi']\!] & = & \{t \mid \forall t' \in [\![\psi]\!].(t \mid t') \in [\![\psi']\!]\} \\
[\![\psi*]\!] & = & \bigcup_{n \in \mathbb{N}} n \cdot [\![\psi]\!] \\
[\![\top]\!] & = & \mathsf{Arbres}_\Sigma \\
[\![\neg\psi]\!] & = & \mathsf{Arbres}_\Sigma \setminus [\![\psi]\!] \\
[\![\psi \vee \psi']\!] & = & [\![\psi]\!] \cup [\![\psi']\!]
\end{array}$$

Pour passer de la logique TL à une représentation sous la forme de produit (un couple) $\langle$ vecteur de cardinalités $\rangle \cdot \langle$ vecteur d'éléments d'arbre $\rangle$, et de celle-ci à un automate, les auteurs proposent dans [Dal Zilio et al., 2004] un formalisme intermédiaire, qui est la *logique à faisceaux*, que nous notons ShL (pour *sheaves logic*). Une formule de cette logique sera notée

[1] Le lien existant entre un ensemble de vecteurs et un ensemble de multiensembles est facile à voir.

$\Psi$ et est définie par la syntaxe suivante :

| | | | |
|---|---|---|---|
| $\alpha$ | ::= | | expression d'étiquette |
| | | $\{a_1, \ldots, a_k\}$ | ensemble fini d'étiquettes |
| | | $\overline{\alpha}$ | l'ensemble complémentaire à $\alpha$ |
| $E$ | ::= | $\alpha[\Psi]$ | emboîtement |
| $\Psi$ | ::= | | formule ShL |
| | | $\top$ | vrai |
| | | $\exists \mathbf{z}.p(\mathbf{z}).\mathbf{E}$ | composition à faisceaux |

Ici $\mathbf{z}$ est un vecteur de variables entières positives, $p$ est une formule de l'arithmétique de Presburger dont les variables libres sont toutes dans $\mathbf{z}$ et $\mathbf{E}$ est un vecteur de formules d'emboîtement. Sans donner la sémantique exacte de ShL, expliquons juste l'interprétation des formules d'emboîtement et des formules ShL ; comme pour les logiques TL et la logique spatiale, l'interprétation de la formule ShL $\Psi$ est un ensemble d'arbres noté $[\![\Psi]\!]$. Tout d'abord, une formule d'emboîtement $E = \alpha[\Psi]$ est interprétée comme l'ensemble d'arbres

$$[\![\alpha[\Psi]]\!] = \{a[t] \mid a \in \alpha, t \in [\![\Psi]\!]\}.$$

Alors l'interprétation de la formule ShL $\Psi = \exists z_1, \ldots, z_k.p(z_1, \ldots, z_k).(E_1, \ldots, E_k)$ est l'ensemble d'arbres

$$\{(n_1, \ldots, n_k) \mid (z_1 \to n_1, \ldots, z_k \to n_k) \text{ satisfait la formule } p\} \cdot ([\![E_1]\!], \ldots, [\![E_k]\!]).$$

défini comme le produit de l'ensemble des vecteurs modèles de la formule de l'arithmétique de Presburger $p$ et le vecteur des ensembles d'éléments d'arbres modèles de $E_1, \ldots, E_k$.

Dans [Dal Zilio et al., 2004] il est établi qu'à toute formule TL correspond une formule ShL équivalente (définissant le même ensemble d'arbres).

Le dernier résultat présenté dans [Dal Zilio et al., 2004] qui nous intéresse ici est la définition d'une classe d'automates, les automates à faisceaux, qui capture la logique ShL, et donc aussi la logique TL. Sans rentrer dans les détails, ces automates sont très proches des automates de Presburger. La seule différence est l'existence de deux types d'états dans les automates à faisceaux : des états pour étiqueter les nœuds de l'arbre et des états pour étiqueter les arêtes de l'arbre. Dans une exécution ascendante d'un automate à faisceaux, l'état à associer à un nœud de l'arbre est déterminé à partir des états associés aux arrêtes successeurs de celui-ci et le nombre d'occurrences de chacun de ces états. La contrainte sur le nombre d'occurrences est exprimée à l'aide d'une formule de Presburger. L'état à associer à une arête est déterminé à partir de l'étiquette de cette arête et de l'état associé ou nœud successeur de celle-ci. Il n'est pas difficile à voir que ces automates sont équivalents aux automates de Presburger.

Dans ce chapitre et le chapitre suivant, nous complétons cette correspondance entre automates et logique à composition, en définissant une classe d'automates qui capture la logique $\mathrm{LS}_{|\exists}$. Nous parlons de « complétion » pour deux raisons. D'abord, la logique TL s'avère être un fragment de $\mathrm{LS}_{|\exists}$. Même si ceci n'est pas immédiat pour l'instant (à cause de l'opérateur d'adjonction qui est absent dans $\mathrm{LS}_{|\exists}$), nous montrons dans le chapitre 7 que PMSO, équivalente aux automates de Presburger, est « incluse » dans $\mathrm{LS}_{|\exists}$, et nous venons de voir que TL peut être encodée dans des automates équivalents aux automates de Presburger. Ceci signifie, entre autres, que l'opérateur d'adjonction n'ajouterait pas d'expressivité à $\mathrm{LS}_{|\exists}$. Par ailleurs, ce n'est pas un

opérateur monotone, et son utilisation avec l'opérateur de point fixe ne pourrait se faire sans restrictions. La deuxième raison pour laquelle nous parlons de complétion est que nous allons établir une équivalence entre des fragments (syntaxiques) de $\mathrm{LS}_{|\exists}$ et des classes d'automates à contraintes numériques. Dans le cas de TL, la logique TL est strictement moins expressive que les automates à faisceaux. [2] Pour s'en convaincre, il suffit de remarquer que TL ne propose pas de mécanisme de récursion en profondeur, qui vient naturellement avec les automates de par la possibilité de « boucler » sur un état.

## 6.2 Automate pour un système d'équations

Nous présentons ici comment à partir d'un système d'équations simplifié $\mathcal{S}$ l'on peut définir un automate à contraintes numériques $A_{\mathcal{S}}$ tel que $\mathcal{L}(A_{\mathcal{S}}) = Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$ (rappelons que $Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$ est la solution de $\mathcal{S}$ pour sa dernière variable). Comme toute formule logique close $\phi$ peut être effectivement représentée par un système d'équations simplifié (voir le chapitre 4), nous pourrons en déduire une construction d'un automate à contraintes numériques $A$ à partir de la formule $\phi$ tel que le langage de $A$ est $[\![\phi]\!]$.

Soit donc $\mathcal{S}$ un système d'équations simplifié que nous considérons fixe jusque la fin de la section ; nous allons définir l'automate $A_{\mathcal{S}} = (\Sigma, Q_{\mathcal{S}}, \Delta_{\mathcal{S}}, F_{\mathcal{S}})$. Dans la section 6.2.1 nous montrons comment l'ensemble d'états $Q_{\mathcal{S}}$ est construit à partir de $\mathcal{S}$. Dans la section 6.2.2 nous introduisons une nouvelle interprétation du système d'équations $\mathcal{S}$, dont nous déduirons dans la section 6.2.3 la relation de transition de l'automate $A_{\mathcal{S}}$.

### 6.2.1 États de l'automate

Nous définissons ici l'ensemble $Q_{\mathcal{S}}$ des états de l'automate $A_{\mathcal{S}}$.

**Définition 6.1 (Variables élémentaires)** Soit $\mathcal{S}'$ un système d'équations simplifié. Une variable $\xi$ de $\mathsf{Vars}(\mathcal{S}')$ est dite *variable élémentaire* si la partie droite de l'équation pour $\xi$ dans $\mathcal{S}'$ est de la forme $\alpha[\xi']$. La sous séquence de $\mathsf{Vars}(\mathcal{S}')$ constituée des variables élémentaires est notée $\mathsf{EVars}(\mathcal{S}')$, et l'ensemble sous-jacent est noté $\{\mathsf{EVars}(\mathcal{S}')\}$.

Pour toute variable élémentaire $\xi$ dans $\mathcal{S}$, soient les formules $\mathsf{Pos}(\xi)$, $\mathsf{Neg}(\xi)$ et $\mathsf{Compl}(\xi)$ définies comme suit :

$$\mathsf{Pos}(\xi) = \alpha[\xi'] \qquad \mathsf{Neg}(\xi) = \alpha[\neg\xi'] \qquad \mathsf{Compl}(\xi) = \overline{\alpha}[\top]$$

où $\xi \overset{\kappa}{=} \alpha[\xi']$ est l'équation pour la variable $\xi$ dans $\mathcal{S}$. $Q_{\mathcal{S}}$, l'ensemble d'états de l'automate $A_{\mathcal{S}}$ est alors défini comme le produit cartésien des ensembles $\{\mathsf{Pos}(\xi), \mathsf{Neg}(\xi), \mathsf{Compl}(\xi)\}$ pour toute variable élémentaire $\xi$ :

$$Q_{\mathcal{S}} = \prod_{\xi \text{ dans } \mathsf{EVars}(\mathcal{S})} \{\mathsf{Pos}(\xi), \mathsf{Neg}(\xi), \mathsf{Compl}(\xi)\}.$$

[2] Dans [Dal Zilio et al., 2004], les auteurs définissent également une extension de la logique à faisceaux par un mécanisme de récursion qui permet d'atteindre la même expressivité que les automates à faisceaux.

Par conséquent, tout état $q$ dans $Q_{\mathcal{S}}$ est un $n$-uplet de la forme $(y_1, \ldots, y_n)$, où pour tout $i$ dans $1..n$, $y_i$ est un parmi $\mathsf{Pos}(\xi_i)$, $\mathsf{Neg}(\xi_i)$ et $\mathsf{Compl}(\xi_i)$ et $\xi_1, \ldots, \xi_n$ sont les variables élémentaires de $\mathcal{S}$. Pour tout état $q$ dans $Q_{\mathcal{S}}$ et toute variable $\xi$ dans $\mathsf{EVars}(\mathcal{S})$, nous notons par $q(\xi)$ l'élément de $q$ correspondant à la variable $\xi$, càd $q(\xi)$ est un parmi $\mathsf{Pos}(\xi)$, $\mathsf{Neg}(\xi)$, et $\mathsf{Compl}(\xi)$.

Notons que si le système d'équations $\mathcal{S}$ n'a pas de variables élémentaires, alors $Q_{\mathcal{S}}$ contient un unique état, le 0-uplet.

### 6.2.2 Systèmes d'équations numériques

Pour définir l'ensemble des transitions de l'automate $A_{\mathcal{S}}$, nous allons transformer le système d'équations $\mathcal{S}$ en un système d'équations appelé système d'équations numériques, et noté $\mathcal{S}^{\text{num}}$, le système $\mathcal{S}^{\text{num}}$ sera appelé système d'équations numériques. Ce système d'équation numérique est interprété sur le treillis de l'ensemble des multiensembles, et à partir de ses solutions nous allons définir les transitions de l'automate $A_{\mathcal{S}}$.

Soit $\rho$ la signature (infinie) $\{\mathbf{0}, \overline{\mathbf{0}}, \top, \bot, \vee, \wedge, \mid, \parallel\} \cup \{C_D\}_{D \in \wp\mathbb{N}^{Q_{\mathcal{S}}}}$. Comme d'habitude, $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ sont des symboles d'arité zéro et $\vee, \wedge, \mid, \parallel$ des symboles binaires utilisés en notation infixe. Les symboles $\{C_D\}_{D \in \wp(\mathbb{N}^{Q_{\mathcal{S}}})}$ sont des symboles d'arité zéro. Considérons la $\mu$-interprétation $\mathcal{R}$ des symboles de $\rho$ dans le treillis $\langle \wp(\mathbb{N}^{Q_{\mathcal{S}}}), \vee, \wedge \rangle$ présentée sur la figure 1. Il est facile de voir que $\mathcal{R}$ est effectivement une $\mu$-interprétation.

$$
\begin{array}{lcl}
C_D^{\mathcal{R}} & = & D \\
\mathbf{0}^{\mathcal{R}} & = & \{0^{Q_{\mathcal{S}}}\} \\
\overline{\mathbf{0}}^{\mathcal{R}} & = & \mathcal{A}_{\Sigma} \setminus \{0^{Q_{\mathcal{S}}}\} \\
\top^{\mathcal{R}} & = & \mathbb{N}^{Q_{\mathcal{S}}} \\
\bot^{\mathcal{R}} & = & \varnothing \\
D \mid^{\mathcal{R}} D' & = & D + D' \\
D \parallel^{\mathcal{R}} D' & = & D +\!\!+ D' \\
D \vee^{\mathcal{R}} D' & = & D \cup D' \\
D \wedge^{\mathcal{R}} D' & = & D \cap D'
\end{array}
$$

FIG. 1 – Définition de la $\mu$-Interprétation $\mathcal{R}$. Ici $D, D'$ sont des sous ensembles de $\mathbb{N}^{Q_{\mathcal{S}}}$.

Comme pour la signature $\tau$, la signature $\rho$ est étendue avec deux nouveaux symboles binaires, $\lceil$ et $\rfloor\!\lceil$, pour obtenir la signature $\rho'$ : $\{\mathbf{0}, \overline{\mathbf{0}}, \top, \bot, \vee, \wedge, \mid, \parallel, \lceil, \rfloor\!\lceil\} \cup \{C_D\}_{D \in \mathbb{N}^{Q_{\mathcal{S}}}}$. L'interprétation $\mathcal{R}$ est étendue à $\rho'$ par :

$$
\begin{aligned}
D \lceil^{\mathcal{R}} D' &= (D \setminus \{0^{Q_{\mathcal{S}}}\}) + (D' \setminus \{0^{Q_{\mathcal{S}}}\}) \\
D \rfloor\!\lceil^{\mathcal{R}} D' &= (D \cup \{0^{Q_{\mathcal{S}}}\}) +\!\!+ (D' \cup \{0^{Q_{\mathcal{S}}}\}).
\end{aligned}
$$

L'opérateur $+\!\!+$ est le dual de la somme de multiensembles et a été défini dans le chapitre 1.

Pour améliorer la lisibilité, dans la suite nous allons confondre le symbole constant $C_D$ et son interprétation, l'ensemble de multiensembles $D$.

Remarquons maintenant que les notion de système d'équations simplifié et système d'équations normalisé peuvent être étendues aux systèmes sur $fonct(\rho, \mathcal{X}_{\mathrm{r}})$ et $fonct(\rho', \mathcal{X}_{\mathrm{r}})$. Plus pré-

cisément, un système sur $fonct(\rho, \mathcal{X}_\mathrm{r})$ sera dit simplifié si les parties droites d'équations sont de la forme

$$\mathbf{0} \quad \overline{\mathbf{0}} \quad \top \quad \bot \quad C_D \quad \xi|\xi' \quad \xi\|\xi' \quad \xi \vee \xi' \quad \xi \wedge \xi'$$

et un système sur $fonct(\rho', \mathcal{X}_\mathrm{r})$ sera dit semi-simplifié si les parties droites d'équations sont soit des termes booléens, soit de la forme

$$\mathbf{0} \quad \overline{\mathbf{0}} \quad \top \quad \bot \quad C_D \quad \xi|\xi' \quad \xi\|\xi' \quad \xi\rceil\xi' \quad \xi\rfloor\xi'.$$

Un système d'équations normalisé sur $fonct(\rho', \mathcal{X}_\mathrm{r})$ est comme précédemment un système semi-simplifié, sans cycles de dépendance booléenne et à composition gardée. De plus, si $\mathcal{S}'$ est un système d'équations semi-simplifié sur $fonct(\rho, \mathcal{X}_\mathrm{r})$, la procédure de normalisation décrite dans la section 4.3.2 peut être appliquée telle quelle pour obtenir un système d'équations sur $fonct(\rho', \mathcal{X}_\mathrm{r})$ normalisé et équivalent à $\mathcal{S}'$.

Soit maintenant le système d'équations numérique $\mathcal{S}^{\mathrm{num}}$ construit à partir de $\mathcal{S}$ de la manière suivante : $\mathsf{Vars}(\mathcal{S}^{\mathrm{num}}) = \mathsf{Vars}(\mathcal{S})$ et pour tout $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, l'équation pour $\xi$ dans $\mathcal{S}^{\mathrm{num}}$ est

$$\begin{cases} \xi \stackrel{\kappa}{=} C_{\mathsf{prj}(\xi)} & \text{si } \xi \in \{\mathsf{EVars}(\mathcal{S})\} \\ \xi \stackrel{\kappa}{=} f & \text{si } \xi \notin \{\mathsf{EVars}(\mathcal{S})\} \text{ et } \xi \stackrel{\kappa}{=} f \text{ est l'équation pour } \xi \text{ dans } \mathcal{S} \end{cases}$$

où pour toute variable $\xi$ dans $\mathsf{EVars}(\mathcal{S})$, $\mathsf{prj}(\xi)$ est l'ensemble de multiensembles dans $\mathbb{N}^{Q_\mathcal{S}}$ défini par (rappelons que $1_q$ est le multiensemble unitaire contenant une seule occurrence de $q$) :

$$\mathsf{prj}(\xi) = \bigcup_{q \in Q_\mathcal{S} | q(\xi) = \mathsf{Pos}(\xi)} \{1_q\}.$$

Soit également le système d'équations normalisé $\mathcal{S}''$ sur $fonct(\tau', \mathcal{X}_\mathrm{r})$ équivalent à $\mathcal{S}'$ et obtenu par la méthode de normalisation décrite dans la section 4.3.2. Comme cette méthode n'affecte pas les variables élémentaires, il est facile de voir que $\mathcal{S}''^{\mathrm{num}}$ est exactement le système d'équations sur $fonct(\rho', \mathcal{X}_\mathrm{r})$ obtenu en normalisant $\mathcal{S}'^{\mathrm{num}}$ par cette même méthode de normalisation.

### 6.2.3 Transitions de l'automate

Nous pouvons maintenant définir l'ensemble des transitions de l'automate $A_\mathcal{S}$.

Pour tout $\xi$ variable élémentaire de $\mathcal{S}$, soient $\mathsf{etq}(\xi)$ l'étiquette et $\mathsf{var}(\xi)$ la variable dans $\mathsf{Vars}(\mathcal{S})$ tels que l'équation pour $\xi$ dans $\mathcal{S}$ est $\xi \stackrel{\kappa}{=} \mathsf{etq}(\xi)[\mathsf{var}(\xi)]$. Par exemple, si l'équation pour $\xi$ dans $\mathcal{S}$ est $\xi \stackrel{\kappa}{=} \alpha[\xi']$, alors $\mathsf{etq}(\xi) = \alpha$ et $\mathsf{var}(\xi) = \xi'$.

Pour tout état $q$ dans $Q_\mathcal{S}$, soient $\alpha_q$ l'ensemble d'étiquettes et $D_q$ l'ensemble de multiensembles dans $\mathbb{N}^{Q_\mathcal{S}}$ définis par :

$$\alpha_q = \Sigma \cap \bigcap_{\xi \mid q(\xi) = \mathsf{Pos}(\xi) \text{ ou } q(\xi) = \mathsf{Neg}(\xi)} \mathsf{etq}(\xi) \cap \bigcap_{\xi \mid q(\xi) = \mathsf{Compl}(\xi)} \overline{\mathsf{etq}(\xi)} \tag{6.1}$$

$$D_q = \mathbb{N}^{Q_\mathcal{S}} \cap \bigcap_{\xi \mid q(\xi) = \mathsf{Pos}(\xi)} Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \mathsf{var}(\xi)) \cap \bigcap_{\xi \mid q(\xi) = \mathsf{Neg}(\xi)} \mathbb{N}^{Q_\mathcal{S}} \setminus Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \mathsf{var}(\xi)) \tag{6.2}$$

Maintenant, $\Delta_\mathcal{S}$, la relation de transition de l'automate $A_\mathcal{S}$, est donnée par :

$$\Delta_\mathcal{S} = \{(q, \alpha_q, D_q) \mid q \in Q_\mathcal{S}\}$$

et $F_\mathcal{S}$, la condition d'acceptation de $A_\mathcal{S}$, est l'ensemble $Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \mathsf{dern}(\mathcal{S}^{\mathrm{num}}))$.

### 6.2.4 Exemple

Dans cette section, nous donnons un exemple de la construction d'un automate à contraintes numériques à partir d'un système d'équations simplifié.

Soit $\phi$ la formule $\mu\xi.\{a\}[\xi] \mid \{b\}[\xi] \mid \xi \vee \mathbf{0}$. Cette formule est satisfaite par l'ensemble des arbres ayant autant d'arêtes successeurs étiquetées par $a$ que d'arêtes successeurs étiquetées par $b$ pour chaque nœud de l'arbre.

Dans la suite, nous écrirons $a[\xi]$ et $b[\xi]$ à la place de $\{a\}[\xi]$ et $\{b\}[\xi]$ et $\overline{a}$ et $\overline{b}$ pour désigner les ensembles d'étiquettes $\overline{\{a\}}$ et $\overline{\{b\}}$.

Soit $\mathcal{S}$ le système d'équations simplifié correspondant à la formule $\phi$, $\mathcal{S}$ est alors le système suivant :

$$\begin{aligned}
\xi_1 &\stackrel{\mu}{=} \mathbf{0} \\
\xi_2 &\stackrel{\mu}{=} a[\xi] \\
\xi_3 &\stackrel{\mu}{=} b[\xi] \\
\xi_4 &\stackrel{\mu}{=} \xi_2 \,|\, \xi_3 \\
\xi_5 &\stackrel{\mu}{=} \xi_4 \,|\, \xi \\
\xi &\stackrel{\mu}{=} \xi_5 \vee \xi_1
\end{aligned}$$

Nous allons construire l'automate $A_{\mathcal{S}} = (\Sigma, Q_{\mathcal{S}}, \Delta_{\mathcal{S}}, F_{\mathcal{S}})$.

Les variables élémentaires du système $\mathcal{S}$ sont $\xi_2$ et $\xi_3$. Nous savons que $Q_{\mathcal{S}}$, l'ensemble des états de l'automate $A_{\mathcal{S}}$ est

$$Q_{\mathcal{S}} = \{\mathsf{Pos}(\xi_2),\ \mathsf{Neg}(\xi_2),\ \mathsf{Compl}(\xi_2)\} \times \{\mathsf{Pos}(\xi_3),\ \mathsf{Neg}(\xi_3),\ \mathsf{Compl}(\xi_3)\}$$

où $\mathsf{Pos}(\xi_i)$, $\mathsf{Neg}(\xi_i)$ et $\mathsf{Compl}(\xi_i)$, pour $i = 2, 3$, sont les formules

$$\begin{array}{lll}
\mathsf{Pos}(\xi_2) = a[\xi] & \mathsf{Neg}(\xi_2) = a[\neg\xi] & \mathsf{Compl}(\xi_2) = \overline{a}[\top] \\
\mathsf{Pos}(\xi_3) = b[\xi] & \mathsf{Neg}(\xi_3) = b[\neg\xi] & \mathsf{Compl}(\xi_3) = \overline{b}[\top]
\end{array}$$

Donc, $Q_{\mathcal{S}}$ est un ensemble de neuf éléments. Pour la suite, nous adoptons une écriture plus brève pour ces neuf éléments

$$Q_{\mathcal{S}} = \{q\mathsf{pp}, q\mathsf{pn}, q\mathsf{pc}, q\mathsf{np}, q\mathsf{nn}, q\mathsf{nc}, q\mathsf{cp}, q\mathsf{cn}, q\mathsf{cc}\}$$

où $q\mathsf{pp} = (\mathsf{Pos}(\xi_2), \mathsf{Pos}(\xi_3))$, ce qui par définition est égal à $(a[\xi], b[\xi])$, $q\mathsf{pn} = (\mathsf{Pos}(\xi_2), \mathsf{Neg}(\xi_3))$, ce qui est égal à $(a[\xi], b[\neg\xi])$, $q\mathsf{pc} = (\mathsf{Pos}(\xi_2), \mathsf{Compl}(\xi_3))$, ce qui est égal à $(a[\xi], \overline{b}[\top])$, etc.

Pour construire l'ensemble des transitions de l'automate $A_{\mathcal{S}}$, nous devons d'abord construire le système d'équations numérique correspondant à $\mathcal{S}$, noté $\mathcal{S}^{\text{num}}$. Par définition, $\mathcal{S}^{\text{num}}$ est :

$$\begin{aligned}
\xi_1 &\stackrel{\mu}{=} \mathbf{0} \\
\xi_2 &\stackrel{\mu}{=} C_{\mathsf{prj}(\xi_2)} \\
\xi_3 &\stackrel{\mu}{=} C_{\mathsf{prj}(\xi_3)} \\
\xi_4 &\stackrel{\mu}{=} \xi_2 \,|\, \xi_3 \\
\xi_5 &\stackrel{\mu}{=} \xi_4 \,|\, \xi \\
\xi &\stackrel{\mu}{=} \xi_5 \vee \xi_1
\end{aligned}$$

où $\mathsf{prj}(\xi_2) = 1_{q\mathsf{pp}} \cup 1_{q\mathsf{pn}} \cup 1_{q\mathsf{pc}}$ et $\mathsf{prj}(\xi_3) = 1_{q\mathsf{pp}} \cup 1_{q\mathsf{np}} \cup 1_{q\mathsf{cp}}$.

Comme nous l'avons vu, il y a une transition par état de l'automate. Pour chaque état $q\mathsf{xy}$, soit $(q\mathsf{xy}, \alpha_{\mathsf{xy}}, D_{\mathsf{xy}})$ cette transition, où $\alpha_{\mathsf{xy}}$ est un ensemble d'étiquettes et $D_{\mathsf{xy}}$ est un ensemble de multiensembles dans $\mathbb{N}^Q$. Nous savons que les ensembles $\alpha_{\mathsf{xy}}$ sont définis et effectivement calculables (en tant qu'ensembles d'étiquettes finis ou cofinis) à partir du système d'équations $\mathcal{S}$, et que les $D_{\mathsf{xy}}$ sont définis de façon symbolique à partir de la solution du système d'équations $\mathcal{S}^{\mathrm{num}}$.

Regardons la définition de $\alpha_{\mathsf{xy}}$ et $D_{\mathsf{xy}}$ juste pour les états $q\mathsf{xy} = q\mathsf{pp}, q\mathsf{nc}, q\mathsf{cc}$. Par définition,

$$\alpha_{\mathsf{xy}} = \Sigma \cap \bigcap_{\xi' \mid q\mathsf{xy}(\xi')=\mathsf{Pos}(\xi') \text{ ou } q\mathsf{xy}(\xi')=\mathsf{Neg}(\xi')} \mathsf{etq}(\xi') \cap \bigcap_{\xi' \mid q\mathsf{xy}(\xi')=\mathsf{Compl}(\xi')} \overline{\mathsf{etq}(\xi')}$$

où, rappelons le, $\mathsf{etq}(\xi')$ est l'ensemble de variables apparaissant en partie droite de l'équation dans $\mathcal{S}$ pour la variable $\xi'$. Dans la formule ci-dessus, $\xi'$ est une variable élémentaire, et donc dans notre cas peut prendre la valeur $\xi_2$ ou $\xi_3$.

Maintenant, pour l'état $q\mathsf{pp}$, $q\mathsf{pp}(\xi_2) = \mathsf{Pos}(\xi_2)$ et $q\mathsf{pp}(\xi_3) = \mathsf{Pos}(\xi_3)$ et donc $\alpha_{\mathsf{pp}} = \mathsf{etq}(\xi_2) \cap \mathsf{etq}(\xi_3)$, ce qui est égal à $\{a\} \cap \{b\}$ et donc à l'ensemble vide.

Pour l'état $q\mathsf{nc}$, nous avons $q\mathsf{nc}(\xi_2) = \mathsf{Neg}(\xi_2)$ et $q\mathsf{nc}(\xi_3) = \mathsf{Compl}(\xi_3)$, et donc $\alpha_{\mathsf{nc}} = \mathsf{etq}(\xi_2) \cap \overline{\mathsf{etq}(\xi_3)}$, ce qui est égal à $\{a\} \cap \bar{b}$, et donc à $\{a\}$.

Finalement, pour l'état $q\mathsf{cc}$, nous obtenons que $\alpha_{\mathsf{cc}} = \overline{\mathsf{etq}(\xi_2)} \cap \overline{\mathsf{etq}(\xi_3)}$, ce qui est égal à $\overline{\{a, b\}}$.

Pour l'ensembles d'étiquettes $D_{\mathsf{xy}}$, nous savons que par définition

$$D_{\mathsf{xy}} = \mathbb{N}^{Q_S} \cap \bigcap_{\xi' \mid q(\xi')=\mathsf{Pos}(\xi')} Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \mathsf{var}(\xi')) \cap \bigcap_{\xi' \mid q(\xi')=\mathsf{Neg}(\xi')} \mathbb{N}^{Q_S} \setminus Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \mathsf{var}(\xi'))$$

Encore une fois, $\xi'$ peut prendre comme valeur $\xi_2$ ou $\xi_3$. Rappelons que $\mathsf{var}(\xi')$ est la variable apparaissant en partie droite de l'équation pour $\xi'$. Dans notre cas, $\mathsf{var}(\xi_2)$ et $\mathsf{var}(\xi_3)$ sont toutes les deux égales à la variable $\xi$. Notons donc par $D$ la solution de $\mathcal{S}^{\mathrm{num}}$ pour la variable $\xi$ : $D = Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$.

Maintenant, d'après cette définition, nous obtenons que $D_{\mathsf{pp}} = D$. Pour $D_{\mathsf{nc}}$, nous avons $D_{\mathsf{nc}} = \mathbb{N}^{Q_S} \setminus D$. Finalement, $D_{\mathsf{cc}} = \mathbb{N}^{Q_S}$.

Pour finir avec la définition de l'automate $A_{\mathcal{S}}$, notons que $F_{\mathcal{S}}$, la condition d'acceptation de cet automate est l'ensemble $D$.

Ce qui est intéressant à voir, c'est comment la définition syntaxique des états de l'automate $A_{\mathcal{S}}$ est reliée avec le langage de l'automate pour chacun de ces états. Pour voir cette correspondance, nous listons ci-dessous toutes les transitions de l'automate $A_{\mathcal{S}}$, en utilisant cette fois-ci non pas la notation courte des états de la forme $q\mathsf{xy}$, mais leur définition sous la forme de couples

de formules.

$$\begin{aligned}\Delta_{\mathcal{S}} = \{\ & (\,(a[\xi], b[\xi]), && \{a\} \cap \{b\}, && D \cap D\ ),\\ & (\,(a[\xi], b[\neg\xi]), && \{a\} \cap \{b\}, && D \cap \overline{D}\ ),\\ & (\,(a[\xi], \overline{b}[\top]), && \{a\} \cap \overline{b}, && D\ ),\\ & (\,(a[\neg\xi], b[\xi]), && \{a\} \cap \{b\}, && \overline{D} \cap D\ ),\\ & (\,(a[\neg\xi], b[\neg\xi]), && \{a\} \cap \{b\}, && \overline{D} \cap \overline{D}\ ),\\ & (\,(a[\neg\xi], \overline{b}[\top]), && \{a\} \cap \overline{b}, && \overline{D}\ ),\\ & (\,(\overline{a}[\top], b[\xi]), && \overline{a} \cap \{b\}, && D\ ),\\ & (\,(\overline{a}[\top], b[\neg\xi]), && \overline{a} \cap \{b\}, && \overline{D}\ ),\\ & (\,(\overline{a}[\top], \overline{b}[\top]), && \overline{a} \cap \overline{b}, && \mathbb{N}^{Q_{\mathcal{S}}}\ )\ \}\end{aligned}$$

Ce sont ces correspondances qui sont à l'origine de la définition des états de l'automate sous la forme de couples de formules. Pour être plus précis, chaque état de l'automate $A_{\mathcal{S}}$ est un tuple de formules logiques faisant intervenir des variables libres ; la taille de ces tuples est égale au nombre de variables élémentaires dans le système $\mathcal{S}$. La définition de l'automate $A_{\mathcal{S}}$ garantit que pour chaque état $q$ de l'automate, le langage associé à cet état est égal à l'intersection des interprétations des formules qui le forment, sous la valuation qui à toute variable libre $\xi$ associe la solution de $\mathcal{S}$ pour cette variable. Retournant à notre exemple, nous avons par exemple pour l'état $q\mathsf{pp} = (a[\xi], b[\xi])$ :

$$\mathcal{L}_{A_{\mathcal{S}}}((a[\xi], b[\xi])) = [\![a[\xi]]\!]_\delta \cap [\![b[\xi]]\!]_\delta$$

où $\delta$ est la valuation qui à $\xi$ associe $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$. Pour cet état particulier, $\mathcal{L}_{A_{\mathcal{S}}}(q\mathsf{pp}) = \varnothing$.

Pour terminer avec cet exemple, nous allons donner une définition effective de l'automate $A_{\mathcal{S}}$ ; il nous manque uniquement une représentation effective de l'ensemble $D$. On peut montrer que l'ensemble $D$ est égal à

$$D = \left\{ \begin{array}{lll} q\mathsf{pp} \mapsto n_{\mathsf{pp}}, & q\mathsf{pn} \mapsto n_{\mathsf{pn}}, & q\mathsf{pc} \mapsto n_{\mathsf{pc}}, \\ q\mathsf{np} \mapsto n_{\mathsf{np}}, & q\mathsf{nn} \mapsto 0, & q\mathsf{nc} \mapsto 0, \\ q\mathsf{cp} \mapsto n_{\mathsf{cp}}, & q\mathsf{cn} \mapsto 0, & q\mathsf{cc} \mapsto 0 \end{array} \middle| \; n_{\mathsf{pp}} + n_{\mathsf{pn}} + n_{\mathsf{pc}} = n_{\mathsf{pp}} + n_{\mathsf{np}} + n_{\mathsf{cp}} \right\}$$

Comme nous l'avons déjà noté, c'est également la condition d'acceptation de l'automate. On voit que l'automate $A_{\mathcal{S}}$ ainsi construit correspond peu à l'intuition qu'on peut se faire d'un automate reconnaissant l'ensemble d'arbres $[\![\phi]\!]$. Par ailleurs, un automate reconnaissant cet ensemble d'arbres a été présenté dans l'exemple 2.1. Cependant, si nous enlevons dans $A_{\mathcal{S}}$ les états dont le langage est vide, on peut voir que l'automate $A_{\mathcal{S}}$ est proche de l'intuition. En effet, $\mathcal{L}_{A_{\mathcal{S}}}(q\mathsf{xy})$ est vide pour $q\mathsf{xy} \in \{q\mathsf{pp}, q\mathsf{pn}, q\mathsf{np}, q\mathsf{nn}\}$, ce qui signifie que ces états ne seront jamais utilisés dans une exécution de l'automate. Donc, l'ensemble $D$ peut être remplacé (dans les transitions de l'automate $A_{\mathcal{S}}$) par l'ensemble $D'$ :

$$D' = D \cap \left\{ \begin{array}{lll} q\mathsf{pp} \mapsto 0, & q\mathsf{pn} \mapsto 0, & q\mathsf{pc} \mapsto n_{\mathsf{pc}}, \\ q\mathsf{np} \mapsto 0, & q\mathsf{nn} \mapsto 0, & q\mathsf{nc} \mapsto n_{\mathsf{nc}}, \\ q\mathsf{cp} \mapsto n_{\mathsf{cp}}, & q\mathsf{cn} \mapsto n_{\mathsf{cn}}, & q\mathsf{cc} \mapsto n_{\mathsf{cc}} \end{array} \middle| \; n_{\mathsf{pc}}, n_{\mathsf{nc}}, n_{\mathsf{cn}}, n_{\mathsf{cp}}, n_{\mathsf{cn}}, n_{\mathsf{cc}} \in \mathbb{N} \right\}$$

C'est à dire, l'ensemble $D'$ est égal à

$$D' = \left\{ \begin{array}{lll} q\mathsf{pp} \mapsto 0, & q\mathsf{pn} \mapsto 0, & q\mathsf{pc} \mapsto n_{\mathsf{pc}}, \\ q\mathsf{np} \mapsto 0, & q\mathsf{nn} \mapsto 0, & q\mathsf{nc} \mapsto 0, \\ q\mathsf{cp} \mapsto n_{\mathsf{cp}}, & q\mathsf{cn} \mapsto 0, & q\mathsf{cc} \mapsto 0 \end{array} \middle| \; n_{\mathsf{pc}} = n_{\mathsf{cp}} \right\}$$

**Remarque sur les systèmes sans variables élémentaires**

Comme la définition de l'automate $A_{\mathcal{S}}$ utilise l'ensemble de variables élémentaires du système $\mathcal{S}$, nous discutons ici de la définition de cet automate lorsque $\mathcal{S}$ est un système d'équations sans variables élémentaires. D'abord, l'ensemble $Q_{\mathcal{S}}$ des états de l'automate est défini comme le produit cartésien de $n$ ensembles, où $n$ est le nombre de variables élémentaires du système $\mathcal{S}$. Lorsque $n$ est égal à zéro, $Q_{\mathcal{S}}$ est le produit cartésien de zéro ensembles, et contient donc un unique élément. Nous en déduisons que si $\mathcal{S}$ est un système d'équations sans variables élémentaires, alors l'automate $A_{\mathcal{S}}$ a un unique état.

Par définition, l'automate $A_{\mathcal{S}}$ a autant de transitions que d'états, soit $(q, \alpha, D)$ son unique transition. Il n'est pas difficile de voir par leur définitions que $\alpha = \Sigma$ et $D = \mathbb{N}^{Q_{\mathcal{S}}}$.

Regardons maintenant quel est le type de langages qui peuvent être reconnus par cet automate. C'est un automate déterministe et complet, donc pour tout arbre $t$, il existe une exécution de $A_{\mathcal{S}}$ sur cet arbre. Nécessairement, toute arête de $t$ est étiquetée par $q$ dans cette exécution. En ce qui concerne la condition d'acceptation de l'automate, c'est un sous ensemble de $\mathbb{N}$, soit $F$. L'arbre $t$ est alors accepté par $A_{\mathcal{S}}$ si le nombre de successeurs de la racine dans $t$ appartient à $F$. Donc, dans ce cas, le langage de $A_{\mathcal{S}}$ est défini uniquement par le nombre d'arêtes successeurs de la racine.

Pour finir, faisons le lien avec la formule $\phi$ qui aurait servi à construire le système $\mathcal{S}$. $\phi$ est une formule qui n'utilise pas l'opérateur d'emboîtement, et on peut voir que l'ensemble d'arbres $[\![\phi]\!]$ peut effectivement être défini par le nombre d'arêtes successeurs de la racine. Voici quelques exemples de telles formules :

- plus de trois successeurs de la racine

$$\overline{\mathbf{0}} \,|\, \overline{\mathbf{0}} \,|\, \overline{\mathbf{0}};$$

- au plus trois successeurs de la racine

$$\neg(\overline{\mathbf{0}} \,|\, \overline{\mathbf{0}} \,|\, \overline{\mathbf{0}} \,|\, \overline{\mathbf{0}});$$

- un nombre pair de successeurs de la racine

$$\mu\xi.((\overline{\mathbf{0}} \,|\, \overline{\mathbf{0}}) \wedge \neg(\overline{\mathbf{0}} \,|\, \overline{\mathbf{0}} \,|\, \overline{\mathbf{0}})) \,|\, \xi \vee \mathbf{0};$$

- toute formule n'utilisant que les opérateurs booléens a pour interprétation l'ensemble de tous les arbres ou l'ensemble vide.

### 6.2.5 Remarques sur la taille de l'automate et le calcul d'une exécution

Nous ne pouvons pas vraiment parler de complexité de la construction pour l'automate $A_{\mathcal{S}}$, puisqu'il est défini uniquement de façon syntaxique ; les ensembles $D_q$ tels que $(q, \alpha_q, D_q)$ est

une transition de l'automate n'ont pas d'autre représentation qu'une expression faisant intervenir des opérations ensemblistes sur les composantes de la solution du système d'équations $\mathcal{S}^{\text{num}}$. Nous pouvons tout de même donner une borne inférieure de la taille de l'automate $A_{\mathcal{S}}$ : le nombre d'états de $A_{\mathcal{S}}$ est égal à $3^k$, où $k$ est le nombre de variables élémentaires dans $\mathcal{S}$, et donc $k$ est limité par le nombre d'équations dans le système $\mathcal{S}$ et par la taille de la formule $\phi$ telle que $\mathcal{S}$ est un système d'équations simplifié équivalent à $\phi$.

En ce qui concerne le calcul d'une exécution, rappelons qu'il suffit de

(i) pouvoir énumérer en un temps fini les triplets $(q, \alpha, D)$ appartenant à $\Delta$,

(ii) pour toute étiquette $a$ et tout ensemble d'étiquettes $\alpha$ apparaissant dans une transition de l'automate, pouvoir décider si $a$ appartient à $\alpha$ et

(iii) pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^{Q_{\mathcal{S}}}$ et pour tout ensemble $D \subseteq \mathbb{N}^{Q_{\mathcal{S}}}$ apparaissant dans une transition de l'automate, pouvoir décider si $\mathbf{d}$ appartient à $D$.

La première condition est satisfaite par construction, puisque l'automate a un nombre fini de transitions. Pour la deuxième condition, les ensembles d'étiquettes apparaissant dans les transitions de l'automate sont exactement les ensembles $\alpha_q$ définis ci-dessus. Rappelons que les $\alpha_q$ sont définis par des opérations ensemblistes sur des ensembles apparaissant dans le système d'équations $\mathcal{S}$, et donc sont des ensembles finis ou co-finis d'étiquettes. Le test d'appartenance pour un ensemble fini ou co-fini est décidable pour sa représentation canonique. Finalement, pour la troisième condition, étant donné la définition des ensembles $D_q$, qui sont exactement les ensembles apparaissant dans une transition de l'automate $A_{\mathcal{S}}$, et par (6.2), pour tester l'appartenance de $\mathbf{d}$ à $D_q$ (pour un certain $q$), il suffit de pouvoir tester l'appartenance de $\mathbf{d}$ à $Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}, \xi)$ pour chaque variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{\text{num}})$. Nous en déduisons le lemme suivant :

**Lemme 6.2** *Si pour toute variable $\xi$ dans $\mathsf{var}(\mathcal{S}^{num})$ et pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^{Q_{\mathcal{S}}}$ on peut décider si $\mathbf{d}$ appartient à $Sol_{\mathcal{R}}(\mathcal{S}^{num}, \xi)$, alors l'automate $A_{\mathcal{S}}$ est à exécution calculable.*

## 6.3 Preuve de correction de l'automate

Nous montrons ici que la construction de l'automate $A_{\mathcal{S}}$ est correcte, càd $\mathcal{L}(A_{\mathcal{S}})$ est égal à $Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$, ceci pour tout système d'équations simplifié $\mathcal{S}$. La preuve utilise la notion de base que nous présentons d'abord dans la section 6.3.1. Dans la section 6.3.2 nous énonçons quelques propriétés des bases, et finalement la preuve de correction est présentée dans la section 6.3.3.

### 6.3.1 Bases – définitions et exemples

Nous voulons pouvoir représenter un ensemble d'arbres $T$ sous la forme d'un couple (un produit) $D, \Theta$, où $\Theta$ est un support et peut être apparenté à un vecteur d'ensembles d'éléments d'arbres et $D$ est la décomposition de $T$ sur $\Theta$ est peut être apparenté à un ensemble de vecteurs. Comme nous l'avons vu, cette représentation des ensembles d'arbres sous la forme d'un produit est utilisée dans la logique à faisceaux, mais aussi dans les automates à contraintes numériques. Dans la formule de la logique à faisceaux $\exists \mathbf{z}.p(\mathbf{z}).\mathbf{E}$, $\mathbf{E}$ est le support et $\mathbf{z}$ est la décomposition. Pour l'automate à contraintes numériques $A = (\Sigma, \{q_1, \ldots, q_n\}, \Delta, F)$, le langage de $A$ est défini comme le « produit » de la condition d'acceptation (le décomposition) $F$ et le support

$\mathcal{L}_A(q_1), \ldots, \mathcal{L}_A(q_n)$. Comme ça a été noté dans [Dal Zilio et al., 2004], certains supports ont de meilleures propriétés que d'autres, par exemple les supports qui forment une partition de l'ensemble des éléments d'arbres. De tels supports sont appelés des bases. Donc, du point de vue de son utilisation, une base est une partition finie de l'ensemble des éléments d'arbres $\mathsf{EArbres}_\Sigma$.[3]

Étant donné un système d'équations simplifié $\mathcal{S}$, nous allons définir une méthode pour construire une base $\Theta_\mathcal{S}$ et une décomposition $D$ telle que l'ensemble d'arbres $Sol_\mathcal{T}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$ s'écrit comme le couple $D, \Theta_\mathcal{S}$. Nous allons montrer ensuite que la partition de l'ensemble des éléments d'arbres $\mathsf{EArbres}_\Sigma$ définie par $\Theta_\mathcal{S}$ est la même que la partition définie par l'ensemble des états de l'automate $A_\mathcal{S}$ défini dans la section précédente. De cela nous déduirons la correction de la construction de l'automate $A_\mathcal{S}$.

Nous commençons par introduire la notion de base et définir quelques propriétés des bases.

Rappelons une notation qui a été introduite au début de la thèse : si $P$ et $Q$ sont des ensembles quelconques, $P^Q$ l'ensemble des applications de $Q$ dans $P$. Si $P$ est l'ensemble des entiers naturels, ces applications sont appelées des multiensembles.

Chacune des définitions qui suivent est illustrée dans l'exemple 6.1 de la page 122.

**Définition 6.3 (Base)** Soit $Q$ un ensemble fini et non vide. Une *base sur* $Q$ est une application $\Theta$ de $Q$ dans $\wp(\mathsf{EArbres}_\Sigma)$ telle que
- pour tous états $q, q'$ dans $Q$ différents , l'ensemble $\Theta(q) \cap \Theta(q')$ est vide ;
- $\bigcup_{q \in Q} \Theta(q) = \mathsf{EArbres}_\Sigma$.

Pour des exemples de bases, voir les points (i), (ii) et (iii) de l'exemple 6.1.

**Définition 6.4 (Produit de bases)** Soient $Q_1, \ldots, Q_n$ des ensembles finis non vides et soient $\Theta_1, \ldots, \Theta_n$ des bases sur $Q_1, \ldots, Q_n$ respectivement. Le *produit* de ces $n$ bases, noté $\prod_{i \in 1..n} \Theta_i$ est l'application dans $\wp(\mathsf{EArbres}_\Sigma)^{Q_1 \times \cdots \times Q_n}$ définie par

$$\prod_{i \in 1..n} \Theta_i \ : \ Q_1 \times \cdots \times Q_n \ \to \ \wp(\mathsf{EArbres}_\Sigma)$$
$$\prod_{i \in 1..n} \Theta_i \ : \ (q_1, \ldots, q_n) \ \mapsto \ \bigcap_{i \in 1..n} \Theta_i(q_i).$$

Le produit de bases est également une base.

**Lemme 6.5** *Soient $Q_1, \ldots, Q_n$ des ensembles finis non vides, et soient $\Theta_1, \ldots, \Theta_n$ des bases sur $Q_1, \ldots, Q_n$ respectivement. Alors $\prod_{i \in 1..n} \Theta_i$ est une base sur $Q_1 \times \cdots \times Q_n$.*

*Preuve* Soit $\Theta = \prod_{i \in 1..n} \Theta_i$. Le fait que pour tous $(q_1, \ldots, q_n), (q'_1, \ldots, q'_n) \in Q_1 \times \cdots \times Q_n$, $\Theta((q_1, \ldots, q_n)) \cap \Theta((q'_1, \ldots, q'_n)) = \varnothing$ est immédiat de la définition. Pour montrer que

$$\bigcup_{(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n} \Theta((q_1, \ldots, q_n)) = \mathsf{EArbres}_\Sigma,$$

il suffit de montrer que pour tout élément d'arbre $e$, il existe $(q_1, \ldots, q_n)$ tel que $e \in \Theta((q_1, \ldots, q_n))$. En effet, il suffit de prendre $q_i$ tel que $e \in \Theta_i(q_i)$, pour tout $i$ dans $1..n$.

[3] Techniquement, la définition que nous donnons est légèrement différente.

□

Les points (iv) et (v) de l'exemple 6.1 illustrent des produits de bases.

**Définition 6.6 (Décomposition)** Soit $Q$ un ensemble fini non vide et soit $\Theta$ une base sur $Q$.

Pour tout arbre $t = \{\!\{e_1, \ldots, e_n\}\!\}$, la *décomposition de $t$ sur $\Theta$*, notée $\mathsf{dcmp}(t, \Theta)$, est le multiensemble dans $\mathbb{N}^Q$ qui à tout $q$ dans $Q$ associe le nombre d'éléments de $\{\!\{e_1, \ldots, e_n\}\!\}$ qui appartiennent à $\Theta(q)$, càd

$$\mathsf{dcmp}(t, \Theta) : q \mapsto \sum_{i \in 1..n} |\{e_i\} \cap \Theta(q)|$$

où $|F|$ désigne le cardinal de l'ensemble $F$.

Pour tout ensemble d'arbres $T$, *la décomposition de $T$ sur $\Theta$*, notée $\mathsf{dcmp}(T, \Theta)$, et l'ensemble de multiensembles dans $\mathbb{N}^Q$

$$\mathsf{dcmp}(T, \Theta) = \{\mathsf{dcmp}(t, \Theta) \mid t \in T\}.$$

Inversement, tout multiensemble dans $\mathbb{N}^Q$ peut être interprété sur une base sur $Q$, obtenant ainsi un ensemble d'arbres.

**Définition 6.7 (Interprétation)** Soit $Q = \{q_1, \ldots, q_k\}$ un ensemble fini non vide et soit $\Theta$ une base sur $Q$.

Pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$, *l'interprétation de $\mathbf{d}$ sur $\Theta$*, notée $[\![\mathbf{d}, \Theta]\!]$, est l'ensemble d'arbres

$$[\![\mathbf{d}, \Theta]\!] = \mathbf{d}(q_1) \cdot \Theta(q_1) | \cdots | \mathbf{d}(q_k) \cdot \Theta(q_k)$$

Pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^Q$, *l'interprétation de $D$ sur $\Theta$* notée $[\![D, \Theta]\!]$, est l'ensemble d'arbres

$$[\![D, \Theta]\!] = \bigcup_{\mathbf{d} \in D} [\![\mathbf{d}, \Theta]\!].$$

Avant de voir le lien qui existe entre décomposition et interprétation, nous introduisons une dernière notion, celle de base discriminante.

**Définition 6.8 (Discrimination)** Une base $\Theta$ est *discriminante pour l'ensemble d'arbres $T$* si

$$[\![\mathsf{dcmp}(T, \Theta), \Theta]\!] = T.$$

Autrement dit, la base $\Theta$ sur $Q$ est discriminante pour l'ensemble d'arbres $T$ exprime le fait que pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$, ou bien tous les arbres dont la décomposition sur $\Theta$ est $\mathbf{d}$ appartiennent à $T$, ou bien il n'y a pas de tel arbre appartenant à $T$.

Les notions de décomposition, interprétation et discrimination sont illustrées dans les points (vi), (vii) et (viii). de l'exemple 6.1.

*Unicité de la décomposition* Nous terminons par quelques remarques sur l'unicité de la décomposition d'un ensemble d'arbres sur une base. Soit $\Theta$ une base, $T$ un ensemble d'arbres pour

lequel la base $\Theta$ est discriminante et soit $D = \mathsf{dcmp}(T, \Theta)$. L'ensemble $D$ est défini de manière unique. De part les propriétés des bases discriminantes, nous savons que $[\![D, \Theta]\!] = T$. Mais l'ensemble $D$ n'est pas l'unique ensemble ayant cette propriété. En effet, il se peut qu'il existe un ensemble $D'$ différent de $D$ tel que $[\![D', \Theta]\!] = T$. Ceci vient du fait que la base $\Theta$ peut contenir des éléments qui sont des ensembles vides. Considérons par exemple l'ensemble $Q = \{q_1, q_2, q_3\}$ et l'ensemble d'étiquettes fini ou co fini non vide $\alpha$. Soit la base $\Theta$ sur $Q$ suivante :

$$\Theta = (q_1 \mapsto \alpha[\mathcal{A}_\Sigma], q_2 \mapsto \overline{\alpha}[\mathcal{A}_\Sigma], q_3 \mapsto \varnothing).$$

Pour tout multiensemble $\mathbf{d}$ sur $Q$ tel que $\mathbf{d}(q_3)$ est différent de 0, on aurait $[\![\mathbf{d}, \Theta]\!] = \varnothing$. En effet, par définition,

$$[\![\mathbf{d}, \Theta]\!] = \mathbf{d}(q_1) \cdot \Theta(q_1) \,|\, \mathbf{d}(q_2) \cdot \Theta(q_2) \,|\, \mathbf{d}(q_3) \cdot \Theta(q_3)$$

et si $\mathbf{d}(q_3)$ est différent de zéro, on a $\mathbf{d}(q_3) \cdot \varnothing = \varnothing$.

Soit donc $\mathbf{d}$ un multiensemble tel que $\mathbf{d}(q_3) \neq 0$. Si la base $\Theta$ est discriminante pour l'ensemble $T$, en utilisant le lemme 6.10 nous pouvons déduire que $[\![\mathsf{dcmp}(T, \Theta) \cup \{d\}, \Theta]\!] = T$.

Notons qu'il est toujours le cas que si $[\![D, \Theta]\!] = T$, alors la décomposition de l'ensemble d'arbres $T$ sur la base $\Theta$ est incluse dans $D$ : $\mathsf{dcmp}(T, \Theta) \subseteq D$.

Lorsque nous écrirons $[\![D, \Theta]\!] = T$, nous sous-entendons que la base $\Theta$ est discriminante pour l'ensemble d'arbres $T$.

**Exemple 6.1** Soient $a, b, c$ des étiquettes dans $\Sigma$ et soient les ensembles $Q = \{q\}$, $P = \{p_1, p_2\}$, $R = \{r_1, r_2, r_3\}$, $S = \{s_1, s_2, s_3, s_4, s_5\}$.

(i) L'unique base sur $Q$ est l'application

$$\Theta_1 = (q \mapsto \Sigma[\mathcal{A}_\Sigma]).$$

(ii) Pour tout ensemble d'étiquettes $\alpha$ et tout ensemble d'arbres $T$, les applications $\Theta_2^\alpha$ et $\Theta_3^{\alpha,T}$ ci-dessous sont des bases sur $P$ et $R$ respectivement.

$$\Theta_2^\alpha = (p_1 \mapsto \alpha[\mathcal{A}_\Sigma], \quad p_2 \mapsto \overline{\alpha}[\mathcal{A}_\Sigma])$$
$$\Theta_3^{\alpha,T} = (r_1 \mapsto \alpha[T], \quad r_2 \mapsto \alpha[\mathcal{A}_\Sigma \setminus T], \quad r_3 \mapsto \overline{\alpha}[\mathcal{A}_\Sigma])$$

Par exemple, pour l'étiquette $a$ et l'ensemble d'arbres $\{\{\!|\}\!\}$,

$$\Theta_2^{\{a\}} = (p_1 \mapsto \{a\}[\mathcal{A}_\Sigma], \quad p_2 \mapsto \overline{\{a\}}[\mathcal{A}_\Sigma])$$
$$\Theta_3^{\{a\},\{\{\!|\}\!\}} = (r_1 \mapsto \{a\}[\{\{\!|\}\!\}], \quad r_2 \mapsto \{a\}[\mathcal{A}_\Sigma \setminus \{\{\!|\}\!\}], \quad r_3 \mapsto \overline{\{a\}}[\mathcal{A}_\Sigma])$$

(iii) L'application $\Theta_4 \in \mathbb{N}^S$ ci-dessous est une base.

$$\Theta_4 = (s_1 \mapsto \{a\}[\{\{\!|\}\!\}], \quad s_2 \mapsto \{a\}[\mathcal{A}_\Sigma \setminus \{\{\!|\}\!\}],$$
$$s_3 \mapsto \{b\}[\mathcal{A}_\Sigma], \quad s_4 \mapsto \{c\}[\mathcal{A}_\Sigma], \quad s_5 \mapsto \overline{\{a, b, c\}}[\mathcal{A}_\Sigma])$$

(iv) Pour tout ensemble d'étiquettes $\alpha$, le produit des bases $\Theta_1$ (voir le point (i)) et $\Theta_2^\alpha$ (voir le point (ii)) est l'application $\Theta_5 \in \mathbb{N}^{Q \times P}$ suivante :

$$\Theta_5 = ((q, p_1) \mapsto \alpha[\mathcal{A}_\Sigma], \quad (q, p_2) \mapsto \overline{\alpha}[\mathcal{A}_\Sigma].$$

Remarquons que $\Theta_5$ définit le même partitionnement des éléments d'arbres que $\Theta_2^\alpha$. De façon générale, pour toute base $\Theta$, les produits $\Theta \times \Theta_1$ et $\Theta_1 \times \Theta$ définissent le même partitionnement de l'ensemble des éléments d'arbres que $\Theta$.

(v) Soient $\alpha$ et $\beta$ des ensembles d'étiquettes et soit $T$ un ensemble d'arbres. L'application $\Theta_6 = \Theta_2^{\alpha} \times \Theta_3^{\beta,T}$ dans $\mathbb{N}^{Q\times R}$ est

$$\Theta_6 = ((p_1,r_1) \mapsto (\alpha \cap \beta)[T],\quad (p_1,r_2) \mapsto (\alpha \cap \beta)[\mathcal{A}_\Sigma \setminus T],\quad (p_1,r_3) \mapsto (\alpha \cap \overline{\beta})[\mathcal{A}_\Sigma],$$
$$(p_2,r_1) \mapsto (\overline{\alpha} \cap \beta)[T],\quad (p_2,r_2) \mapsto (\overline{\alpha} \cap \beta)[\mathcal{A}_\Sigma \setminus T],\quad (p_2,r_3) \mapsto (\overline{\alpha} \cap \overline{\beta})[\mathcal{A}_\Sigma]$$

De façon générale, si $\Theta'$ et $\Theta''$ sont des bases sur les ensembles $Q'$ et $Q''$ respectivement, et si pour tout $q' \in Q'$ et tout $q'' \in Q''$ les ensembles $\Theta'(q')$ et $\Theta''(q'')$ sont de la forme $\alpha'[T]$, alors c'est aussi le cas de la base produit $\Theta' \times \Theta''$, càd pour tout $(q',q'')$ dans $Q' \times Q''$, l'ensemble $(\Theta' \times \Theta'')((q',q''))$ est de la forme $\alpha'[T]$.

(vi) Soit $t_1$ l'arbre $\{\!|a[\{\!|\}\!],a[b[\{\!|\}\!]],b[\{\!|\}\!]|\!\}$. Alors la décomposition de $t$ sur les bases $\Theta_2^{\{a\}}$ et $\Theta_3^{\{a\},\{\{\!|\}\!\}}$ est comme suit :

$$\mathsf{dcmp}(t,\Theta_2^{\{a\}}) = (p_1 \mapsto 2,\ p_2 \mapsto 1)$$
$$\mathsf{dcmp}(t,\Theta_3^{\{a\},\{\{\!|\}\!\}}) = (r_1 \mapsto 1,\ r_2 \mapsto 1,\ r_3 \mapsto 1)$$

(vii) Soit $T_1$ l'ensemble des arbres dont les arêtes successeurs de la racine sont étiquetées par $a$, $b$ ou $c$ et toutes les arêtes étiquetées par $a$ mènent vers un arbre non vide, càd exprimé par une formule logique, $T_1 = [\![a[\overline{\mathbf{0}}]* \,|\, b[\top]* \,|\, c[\top]*]\!]$.
Alors $\mathsf{dcmp}(T_1,\Theta_4)$ est l'ensemble de multiensembles dans $\mathbb{N}^S$ ci-dessous

$$\mathsf{dcmp}(T_1,\Theta_4) = \{(s_1 \mapsto 0, s_2 \mapsto n_2, s_3 \mapsto n_3, s_4 \mapsto n_4, s_5 \mapsto 0) \mid n_2,n_3,n_4 \in \mathbb{N}\}.$$

(viii) Soit $T_2$ l'ensemble d'arbres pour lesquels au moins la moitié des arêtes successeurs de la racine sont étiquetées par $a$ et au moins deux arêtes successeurs de la racine étiquetées par $a$ mènent vers un arbre non vide. Exprimé par une formule logique, $T_2$ est l'ensemble d'arbres $[\![\phi]\!]$ où $\phi$ est la formule

$$(\ (a[\top] \,|\, \overline{\{a\}}[\top])* \mid a[\top]*\ ) \quad \wedge \quad (a[\overline{\mathbf{0}}] \,|\, a[\overline{\mathbf{0}}] \,|\, \top).$$

Alors la décomposition de $T_2$ sur les bases $\Theta_2^{\{a\}}$ et $\Theta_3^{\{a\},\{\{\!|\}\!\}}$ est comme suit :

$$\mathsf{dcmp}(T_2,\Theta_2^{\{a\}}) = \{(p_1 \mapsto n_1,\ p_2 \mapsto n_2) \mid n_1 \geq n_2, n_1 \geq 2\}$$
$$\mathsf{dcmp}(T_2,\Theta_3^{\{a\},\{\{\!|\}\!\}}) = \{(r_1 \mapsto n_1\ r_2 \mapsto n_2\ r_3 \mapsto n_3) \mid n_1 + n_2 \geq n_3 \text{ et } n_2 \geq 2\}$$

La base $\Theta_3^{\{a\},\{\{\!|\}\!\}}$ est discriminante pour l'ensemble d'arbres $T_2$, mais pas la base $\Theta_2^{\{a\}}$. Intuitivement, la base $\Theta_2^{\{a\}}$ peut distinguer les arêtes successeurs de la racine étiquetées par $a$ des arêtes successeurs de la racine qui ne sont pas étiquetées par $a$, mais cette base ne permet pas de distinguer si une arête étiquetée par $a$ mène vers un arbre non vide, ce qui est nécessaire pour définir la deuxième condition sur $T_2$. Voici un exemple qui montre que $\Theta_2^{\{a\}}$ n'est pas discriminante pour $T_2$. Le multiensemble $\mathbf{d} = (p_1 \mapsto 2,\ p_2 \mapsto 1)$ appartient à $\mathsf{dcmp}(T_2,\Theta_2^{\{\alpha\}})$. Or, nous savons que $\mathbf{d} = \mathsf{dcmp}(t_1,\Theta_2^{\{\alpha\}})$ (voir le point (vi)), mais $t_1$ n'appartient pas à $T_2$.

### 6.3.2 Propriétés des bases discriminantes

Nous annonçons ici quelques propriétés utiles des bases discriminantes, à savoir comment le caractère discriminant d'une base est préservé par le produit de bases (le lemme 6.9) et la stabilité de la décomposition d'un ensemble d'arbres par les opérations booléennes (le lemme 6.10) et l'unicité de la décomposition d'un ensemble sur une base.

**Lemme 6.9** *Soient $Q_1, \ldots, Q_n$ des ensembles finis non vides et soient $\Theta_1, \ldots, \Theta_n$ des bases sur $Q_1, \ldots, Q_n$ respectivement. Notons $U = \prod_{i \in 1..n} Q_i$ et $\Theta = \prod_{i \in 1..n} \Theta_i$. Soit $S$ un ensemble d'arbres tel que la base $\Theta_i$ est discriminante pour $S$, pour un certain $i$ dans $1..n$. Alors la base $\Theta$ est discriminante pour $S$. De plus, si $S = [\![D, \Theta_i]\!]$, $S = [\![H, \Theta]\!]$ avec*

$$H = \bigcup_{\mathbf{d} \in D} \left\{ \mathbf{h} \;\middle|\; \forall q \in Q_i. \left( \sum_{u \in U \mid u(i)=q} \mathbf{h}(u) \right) = \mathbf{d}(q) \right\},$$

*où pour tout $u$ dans $U$ et tout $i$ dans $1..n$, $u(i)$ désigne la composante de $u$ provenant de l'ensemble $Q_i$.*

Avant de donner la preuve de ce lemme, regardons un exemple.

**Exemple 6.2** Soient les ensemble $P = \{p_1, p_2\}$ et $Q = \{q_1, q_2, q_3\}$, $S$ l'ensemble d'arbres $\{c[\{\!|\,|\!\}]\}$ et $\Theta_P, \Theta_Q$ les bases sur $P, Q$ respectivement :

$$\Theta_P = (p_1 \mapsto \{a\}[\mathcal{A}_\Sigma],\ p_2 \mapsto \overline{\{a\}}[\mathcal{A}_\Sigma])$$
$$\Theta_Q = (q_1 \mapsto \{a,b\}[S],\ q_2 \mapsto \{a,b\}[\mathcal{A}_\Sigma \setminus S],\ q_3 \mapsto \overline{\{a,b\}}[\mathcal{A}_\Sigma]).$$

Alors $\Theta_P \times \Theta_Q$ est la base

$$\begin{aligned}\Theta_P \times \Theta_Q = ((p_1,q_1) \mapsto \{a\}[S] \quad & (p_1,q_2) \mapsto \{a\}[\mathcal{A}_\Sigma \setminus S] \quad & (p_1,q_3) \mapsto \varnothing,\\ (p_2,q_1) \mapsto \{b\}[S], \quad & (p_2,q_2) \mapsto \{b\}[\mathcal{A}_\Sigma \setminus S] \quad & (p_2,q_3) \mapsto \overline{\{a,b\}}[\mathcal{A}_\Sigma]\end{aligned}$$

Soit $T$ l'ensemble d'arbres $[\![(q_1 \mapsto 2,\ q_2 \mapsto 1,\ q_3 \mapsto 0), \Theta_Q]\!]$, càd $T$ contient exactement les arbres ayant au total trois arêtes successeurs de la racine, toutes étiquetées par $a$ ou $b$, et deux exactement de ces arêtes mènent vers le sous arbre $c[\{\!|\,|\!\}]$. L'ensemble de multiensembles $D$ tel que $T = [\![D, \Theta_P \times \Theta_Q]\!]$ est alors :

$$\begin{aligned}D = \{(\,&(p_1,q_1) \mapsto x_{1,1},\ (p_1,q_2) \mapsto x_{1,2},\ (p_1,q_3) \mapsto x_{1,3},\\ &(p_2,q_1) \mapsto x_{2,1},\ (p_2,q_2) \mapsto x_{2,2},\ (p_2,q_3) \mapsto x_{2,3}\,)\\ &\mid x_{1,1} + x_{2,1} = 2,\ x_{1,2} + x_{2,2} = 1,\ x_{1,3} + x_{2,3} = 0\}\end{aligned}$$

*Preuve du lemme 6.9* Il suffit de montrer que pour tout arbre $t$, $\mathsf{dcmp}(t, \Theta_i) \in D$ si et seulement si $\mathsf{dcmp}(t, \Theta) \in H$. Soit un arbre $t$ et soient $\mathbf{d} = \mathsf{dcmp}(t, \Theta_i)$ et $\mathbf{h} = \mathsf{dcmp}(t, \Theta)$. Par définition de $H$, il suffit de montrer que pour tout $q \in Q_i$, $\sum_{u \in U \mid u(i)=q} \mathbf{h}(u) = \mathbf{d}(q)$. Soit $q \in Q_i$. Par définition de la décomposition d'un arbre sur une base, il suffit de montrer que ($\star$) pour tout $q \in Q_i$, et pour tout élément d'arbre $e$ composant de $t$, $e \in \Theta_i(q)$ si et seulement si $e \in \Theta(u)$ pour un certain $u$ tel que $u(i) = q$. Or, par définition du produit de bases, il est facile de voir que $\Theta_i(q) = \bigcup_{u \in U \mid u(i)=q} \Theta(u)$, d'où ($\star$) suit immédiatement.

□

Les propriétés énoncées dans le lemme qui suit sont faciles à vérifier.

**Lemme 6.10** *(i) Tout base $\Theta$ est discriminante pour les ensembles d'arbres $\mathcal{A}_\Sigma$, $\varnothing$, $\{\{\!|\}\!\}$ et $\mathcal{A}_\Sigma \setminus \{\{\!|\}\!\}$. De plus,*

$$\begin{aligned}\mathcal{A}_\Sigma &= [\![\mathbb{N}^Q, \Theta]\!]\\ \varnothing &= [\![\varnothing, \Theta]\!]\\ \{\{\!|\}\!\} &= [\![\{0^Q\}, \Theta]\!]\\ \mathcal{A}_\Sigma \setminus \{\{\!|\}\!\} &= [\![\mathbb{N}^Q \setminus \{0^Q\}, \Theta]\!]\end{aligned}$$

*(ii) Pour tous ensembles d'arbres $T, T'$ et toute base $\Theta$, si $\Theta$ est discriminante pour $T$ et $T'$ et $T = [\![D, \Theta]\!]$ et $T' = [\![D', \Theta]\!]$, alors :*

$$\begin{aligned}\mathcal{A}_\Sigma \setminus T &= [\![\mathbb{N}^Q \setminus D, \Theta]\!]\\ T \cup T' &= [\![D \cup D', \Theta]\!]\\ T \cap T' &= [\![D \cap D', \Theta]\!]\\ T \,|\, T' &= [\![D + D', \Theta]\!]\\ T \,\|\, T' &= [\![D +\!\!+ D', \Theta]\!]\end{aligned}$$

### 6.3.3 Lien entre bases et automates et preuve de correction de la construction de l'automate

Nous montrons ici comment, sous certaines conditions, une base $\Theta$ sur $Q$ peut être vue comme un automate à contraintes numériques déterministe dont l'ensemble des états est $Q$. Ensuite nous définissons une base $\Theta_\mathcal{S}$ qui est discriminante pour les ensembles solutions du système d'équations simplifié $\mathcal{S}$. Cette base satisfait les condition requises et définit donc un automate à contraintes numériques. Nous montrons que cet automate est exactement l'automate $A_\mathcal{S}$ défini dans les sections précédentes. Cette mise en correspondance entre l'automate $A_\mathcal{S}$ et la base $\Theta_\mathcal{S}$ nous permet de montrer la correction de la construction de l'automate $A_\mathcal{S}$.

*Lien entre bases et automates déterministes* Soit $Q$ un ensemble fini et non vide et soit $\Theta$ une base sur $Q$ satisfaisant

**(C)** pour tout $q$ dans $Q$, l'ensemble $\Theta(q)$ peut s'écrire comme une union finie d'ensembles de la forme $\alpha[T]$ où $\alpha$ est un ensemble fini ou cofini d'étiquettes et $T$ est un ensemble d'arbres pour lequel la base $\Theta$ est discriminante.

Nous pouvons alors définir une famille d'automates $\mathcal{A}_\Theta = \{(\Sigma, Q, \Delta, F) \mid F \subseteq \mathbb{N}^Q\}$ qui diffèrent uniquement par leur condition d'acceptation. La relation de transition de ces automates est donnée plus loin. Notant $A_F$ l'automate $(\Sigma, Q, \Delta, F)$ dans cette famille, le langage accepté par cet automate est $[\![F, \Theta]\!]$. Autrement dit, une base $\Theta$ satisfaisant la condition **(C)** a la propriété que pour chaque ensemble d'arbres $T$ tel que $\Theta$ est discriminante pour $T$, il existe un automate à contraintes numériques reconnaissant $T$ et la condition d'acceptation de cet automate est $\mathsf{dcmp}(T, \Theta)$.

La relation de transition $\Delta$ de la famille d'automates $\mathcal{A}_\Theta$ est définie comme suit : pour tout $q$ dans $Q$, soit l'entier naturel $n_q$, les ensemble d'étiquettes $\alpha_1^q, \ldots, \alpha_{n_q}^q$ et les ensembles d'arbres $T_1^q, \ldots, T_{n_q}^q$ tels que $\Theta(q) = \bigcup_{i \in 1..n_q} \alpha_i^q[T_i^q]$. Alors $\Delta$ est l'ensemble de transitions

$$\bigcup_{q \in Q} \{(q, \alpha_i^q, \mathsf{dcmp}(T_i^q, \Theta, )) \mid i \in 1..n_q\}.$$

Avant de montrer comment cette correspondance entre bases et automates à contraintes numériques va nous servir pour la preuve de correction de la construction de l'automate $A_\mathcal{S}$, nous définissons une base $\Theta_\mathcal{S}$ pour un système d'équations simplifié $\mathcal{S}$.

**Définition 6.11 (Base pour un système d'équations)** Soit $\mathcal{S}$ un système d'équations simplifié. Pour toute variable $\xi$ dans $\mathsf{EVars}(\mathcal{S})$, $\Theta_\xi$ est la base sur l'ensemble $\{\mathsf{Pos}(\xi), \mathsf{Neg}(\xi), \mathsf{Compl}(\xi)\}$ définie par :

$$\Theta_\xi(\mathsf{Pos}(\xi)) = \alpha[Sol_\mathcal{T}(\mathcal{S}, \xi')] \quad \Theta_\xi(\mathsf{Neg}(\xi)) = \alpha[\mathcal{A}_\Sigma \setminus Sol_\mathcal{T}(\mathcal{S}, \xi')] \quad \Theta_\xi(\mathsf{Compl}(\xi)) = \overline{\alpha}[\mathcal{A}_\Sigma],$$

où $\alpha = \mathsf{etq}(\xi)$ et $\xi' = \mathsf{var}(\xi)$ (càd la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est $\alpha[\xi']$).

$\Theta_\mathcal{S}$ est la base sur l'ensemble $Q_\mathcal{S}$ définie par :

$$\Theta_\mathcal{S} = \prod_{\xi \in \{\mathsf{EVars}(\mathcal{S})\}} \Theta_\xi.$$

(Les formules $\mathsf{Pos}(\xi), \mathsf{Neg}(\xi)$ et $\mathsf{Compl}(\xi)$ ainsi que l'ensemble $Q_\mathcal{S}$ ci-dessous ont été définis dans la section 6.2.1. L'ensemble d'étiquettes $\mathsf{etq}(\xi)$ et la variable $\mathsf{var}(\xi)$ sont définis à la page 114.)

Il est facile de voir par cette définition que les $\Theta_\xi$ sont des bases, et donc, par le lemme 6.5, $\Theta_\mathcal{S}$ est également une base.

*Schéma de la preuve de correction* La suite est consacrée à la preuve que le langage de l'automate $A_\mathcal{S}$ est égal à $Sol_\mathcal{T}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$. Nous commençons par montrer dans le lemme 6.13 que

(i) pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, la base $\Theta_\mathcal{S}$ est discriminante pour les ensembles $Sol_\mathcal{T}(\mathcal{S}, \xi)$,

Ceci implique en particulier que pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, il existe un sous ensemble $D$ de $\mathbb{N}^{Q_\mathcal{S}}$ tel que $Sol_\mathcal{T}(\mathcal{S}, \xi) = [\![D, \Theta_\mathcal{S}]\!]$. Nous montrons dans le lemme 6.14 que cet ensemble est $Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \xi)$, càd

(ii) pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, nous avons $Sol_\mathcal{T}(\mathcal{S}, \xi) = [\![Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \xi), \Theta_\mathcal{S}]\!]$.

Finalement, nous montrons dans le lemme 6.15 que

(iii a) pour tout état $q$ dans $Q_\mathcal{S}$, $\mathcal{L}_{A_\mathcal{S}}(q) = \Theta_\mathcal{S}(q)$ et

(iii b) pour tout sous ensemble $D$ de $\mathbb{N}^{Q_\mathcal{S}}$, $\mathcal{L}_{A_\mathcal{S}}(D) = [\![D, \Theta_\mathcal{S}]\!]$.

Maintenant, par définition nous savons que le langage de l'automate $A_\mathcal{S}$ est égal à $\mathcal{L}_{A_\mathcal{S}}(F)$, où $F$ est l'ensemble $Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \mathsf{dern}(\mathcal{S}))$. Donc, par (iii b), le langage de l'automate $A_\mathcal{S}$ est égal à $[\![Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \mathsf{dern}(\mathcal{S})), \Theta_\mathcal{S}]\!]$ et par (i), c'est égal à $Sol_\mathcal{T}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$. Nous en déduisons que

**Proposition 6.12** *Le langage de l'automate $A_S$ est égal à $Sol_T(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$.*

Notons également que par (iii a) et par définition d'une base, l'automate $A_S$ est déterministe et complet.

Dans le reste de la section nous énonçons et montrons les lemmes 6.13 et 6.14 et 6.15. Nous utilisons, sans le citer, la première partie du lemme 6.10 statuant que toute base est discriminante pour les ensembles d'arbres $\mathcal{A}_\Sigma$, $\varnothing$, $\{\{\!|\}\}$ et $\mathcal{A}_\Sigma \setminus \{\{\!|\}\}$.

**Lemme 6.13** *Pour tout système d'équations $\mathcal{S}$ simplifié et pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, la base $\Theta_S$ est discriminante pour l'ensemble $Sol_T(\mathcal{S}, \xi)$.*

*Preuve* Pour cette preuve nous allons utiliser le fait que le système simplifié $\mathcal{S}$ peut être transformé en un système normalisé équivalent à $\mathcal{S}$ sur l'ensemble de variables $\mathsf{Vars}(\mathcal{S})$. Soit $\mathcal{S}'$ le système d'équations normalisé construit à partir de $\mathcal{S}$, càd les parties droite d'équations dans $\mathcal{S}'$ sont ou bien des constantes $\mathbf{0}$, $\overline{\mathbf{0}}$, $\top$, $\bot$, ou bien de la forme $\alpha[\xi]$, ou bien des termes à composition gardée de la forme $\xi \lceil\!\rceil \xi'$, $\xi \rfloor\!\lfloor \xi'$, ou bien des termes booléens. De plus, $\{\mathsf{Vars}(\mathcal{S})\} \subseteq \{\mathsf{Vars}(\mathcal{S}')\}$ et les solutions de $\mathcal{S}$ et $\mathcal{S}'$ coïncident pour les variables dans $\mathsf{Vars}(\mathcal{S})$. A partir de la procédure de normalisation (voir la section 4.3.2), il est facile de voir que les équations associées aux variables élémentaires de $\mathcal{S}$ sont préservées dans $\mathcal{S}'$ et de plus, dans $\mathcal{S}'$, les seules variables élémentaires sont celles qui étaient déjà des variables élémentaires dans $\mathcal{S}$. De ce fait et grâce à la préservation des solution par la normalisation, il suffit de montrer le lemme pour $\mathcal{S}$ étant un système normalisé. C'est ce que nous faisons. Dans la suite, $\mathcal{S}$ est supposé être un système d'équations normalisé.

Soit $\Theta$ une base et $S$ un ensemble d'arbres tels que $\Theta$ n'est pas discriminante pour $S$. Nous dirons que $\Theta$ n'est pas discriminante pour $S$ avec témoin $(t, s)$ si $t$ et $s$ sont deux arbres ayant la même décomposition sur $\Theta$ et tels que $t \in S$ et $s \notin S$. Montrons d'abord quelques résultats intermédiaires.

**Fait 6.1** Si $f$ est un terme booléen fonctionnel dans $fonct(\tau, \{\xi_1, \ldots, \xi_n\})$, $S_1, \ldots, S_n$ sont des ensembles d'arbres et $\Theta$ est une base qui n'est pas discriminante pour $f(S_1, \ldots, S_n)$ avec témoin $(t, s)$, alors il existe $i$ dans $1..n$ tel que la variable $\xi_i$ apparaît dans $f$ et $\Theta$ n'est pas discriminante pour $S_i$ avec témoin $(t, s)$.

La preuve se fait par récurrence sur la structure de $f$. Pour le cas de base, $f$ est une variable ; soit $f$ la variable $\xi_j$. Alors $f(S_1, \ldots, S_n) = S_j$ et donc il suffit de prendre $i = j$. $f$ ne peut pas être $\top$ ou $\bot$, puisque dans ce cas $f(S_1, \ldots, S_n) = \mathcal{A}_\Sigma$, respectivement à $\varnothing$, et nous savons que toute base est discriminante pour ces deux ensembles. Pour l'induction, si $f = f' \vee f''$ (respectivement $f = f' \wedge f''$), alors $f(S_1, \ldots, S_n) = f'(S_1, \ldots, S_n) \vee f''(S_1, \ldots, S_n)$ (respectivement $f(S_1, \ldots, S_n) = f'(S_1, \ldots, S_n) \wedge f''(S_1, \ldots, S_n)$). Dans ce cas, par le lemme 6.10,

(a) $\Theta$ n'est pas discriminante pour $f'(S_1, \ldots, S_n)$ avec témoin $(t, s)$ ou

(b) $\Theta$ n'est pas discriminante pour $f''(S_1, \ldots, S_n)$ avec témoin $(t, s)$.

Soit (a), donc, par hypothèse de récurrence, il existe un $j$ dans $1..n$ tel que la variable $\xi_j$ apparaît dans $f'$ et $\Theta$ n'est pas discriminante pour $S_j$. Donc la variable $\xi_j$ apparaît également dans $f$, et il suffit de prendre $i = j$. Le même raisonnement s'applique si (b) est vérifié. Ceci termine la preuve du fait 6.1.

**Fait 6.2** Soit $\mathsf{Vars}(\mathcal{S}) = \xi_1, \ldots, \xi_n$, et soit $Sol_{\mathcal{T}}(\mathcal{S}) = \langle S_1, \ldots, S_n \rangle$. Si la variable $\xi_i$ est telle que $\Theta_{\mathcal{S}}$ n'est pas discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi_i)$ avec témoin $(t, s)$, alors il existe une variable $\xi_I$ dans $\mathsf{Vars}(\mathcal{S})$ telle que $\Theta_{\mathcal{S}}$ n'est pas discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi_I)$ avec témoin $(t, s)$ et la partie droite de l'équation pour $\xi_I$ dans $\mathcal{S}$ est une formule à composition.

Pour tout $j$ dans $1..n$, soient $\kappa_j$ et $f_j$ tels que l'équation pour la variable $\xi_j$ dans $\mathcal{S}$ est $\xi_j \stackrel{\kappa_j}{=} f_j$. Si $f_i$ est de la forme $\xi' \lceil \xi''$ ou $\xi' \rfloor\!\lceil \xi''$, alors il n'y a rien à prouver. Supposons que ce n'est pas le cas. Remarquons que

($\star$) $f_i$ ne peut pas être $\bot, \top, \mathbf{0}$ ou $\overline{\mathbf{0}}$ ni de la forme $\alpha[\xi']$,

puisque dans ce cas $Sol_{\mathcal{T}}(\mathcal{S}, \xi_i)$ est l'ensemble d'arbres $\varnothing, \mathcal{A}_\Sigma, \{\{\!|\}\!\}$ ou $\mathcal{A}_\Sigma \setminus \{\{\!|\}\!\}$ respectivement, et toute base est discriminante pour ces ensembles d'arbres. De même, $\xi_i$ ne peut pas être une variable élémentaire, puisque par définition $\Theta_{\mathcal{S}}$ est un produit de bases dont la base $\Theta_{\xi_i}$, et par construction $\Theta_{\xi_i}$ est discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi_i)$, et donc par le lemme 6.9, $\Theta_{\mathcal{S}}$ est également discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi_i)$. Donc nécessairement $f_i$ est un terme booléen différent de $\top, \bot$. Soit la suite $u_1, \ldots, u_k, \ldots$ d'éléments de $1..n$ telle que :

- $u_1 = i$ ;
- pour tout $j \geq 0$, $u_{j+1}$ est tel que $f_{u_j}$ est un terme booléen, $\xi_{u_{j+1}}$ apparaît libre dans $f_{u_j}$ et $\Theta_{\mathcal{S}}$ n'est pas discriminante pour $S_{u_{j+1}}$ avec témoin $(t, s)$.

Par le fait 6.1 nous savons que pour tout $j \geq 0$, si $f_{u_j}$ est un terme booléen, alors $u_{j+1}$ existe. Comme $\mathcal{S}$ est un système normalisé, et donc sans cycles de dépendance booléenne, la suite $u_1, \ldots, u_p, \ldots$ est finie et contient au plus $n - 1$ éléments. De plus, par définition, si $m$ est le dernier élément de la suite, alors $\Theta_{\mathcal{S}}$ n'est pas discriminante pour $S_m$ avec témoin $(t, s)$ et $f_m$ n'est pas un terme booléen. Par ($\star$), $f_m$ n'est pas non plus $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ ni un terme de la forme $\alpha[\xi']$. Donc $f_m$ est de la forme $\xi' \lceil \xi''$ ou $\xi' \rfloor\!\lceil \xi''$. Ceci termine la preuve du fait 6.2

**Fait 6.3** Soit l'arbre $s$, l'ensemble fini $Q$ et la base $\Theta$ sur $Q$, et soit $\mathsf{dcmp}(s, \Theta) = \mathbf{d}$. Soient $\mathbf{d}'$ et $\mathbf{d}''$ dans $\mathbb{N}^Q$ tels que $\mathbf{d} = \mathbf{d}' + \mathbf{d}''$. Alors il existe des arbres $s', s''$ tels que $\mathsf{dcmp}(s', \Theta) = \mathbf{d}'$, $\mathsf{dcmp}(s'', \Theta) = \mathbf{d}''$ et $s = s' \uplus s''$.

La preuve du fait 6.3 est une récurrence facile sur le nombre $n = \sum_{q \in Q} \mathbf{d}'(q)$.

Revenons à la preuve du lemme. Dans la suite, nous notons $<_{\mathbf{d}}$ l'inclusion (stricte) de multiensembles dans $\mathbb{N}^{Q_{\mathcal{S}}}$ ; naturellement $<_{\mathbf{d}}$ est un ordre partiel bien fondé sur $\wp(\mathbb{N}^{Q_{\mathcal{S}}})$. Soit $\mathbf{d}$ un plus petit multiensemble pour $<_{\mathbf{d}}$ tel que

- il existe une variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$ telle que $\Theta_{\mathcal{S}}$ n'est pas discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ avec témoin $(t, s)$ et $\mathsf{dcmp}(t, \Theta_{\mathcal{S}}) = \mathsf{dcmp}(s, \Theta_{\mathcal{S}}) = \mathbf{d}$ et
- la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est une formule à composition.

Nous savons par le fait 6.2 que cette deuxième condition peut toujours être vérifiée. Nous allons montrer que dans ce cas il existe une variable $\xi'$ et un multiensemble $\mathbf{d}' \in \mathbb{N}^{Q_{\mathcal{S}}}$ tels que $\Theta_{\mathcal{S}}$ n'est pas discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi')$ avec témoin $(t', s')$, $\mathsf{dcmp}(t', \Theta_{\mathcal{S}}) = \mathsf{dcmp}(s', \Theta_{\mathcal{S}}) = \mathbf{d}'$ et $\mathbf{d}' <_{\mathbf{d}} \mathbf{d}$. Ceci contredirait l'hypothèse que $\mathbf{d}$ est minimal, et montrerait donc par contradiction que $\Theta_{\mathcal{S}}$ est discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$.

Soit $\xi_1 \lceil \xi_2$ la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$. Alors $Sol_{\mathcal{T}}(\mathcal{S}, \xi) = Sol_{\mathcal{T}}(\mathcal{S}, \xi_1) \lceil^{\mathcal{T}} Sol_{\mathcal{T}}(\mathcal{S}, \xi_2)$. Par hypothèse, $t \in Sol_{\mathcal{T}}(\mathcal{S}, \xi_1) \lceil^{\mathcal{T}} Sol_{\mathcal{T}}(\mathcal{S}, \xi_2)$, donc il existe deux arbres $t_1, t_2$ tels que $t = t_1 \uplus t_2$ et $t_1 \in Sol_{\mathcal{T}}(\mathcal{S}, \xi_1) \setminus \{\{\!|\}\!\}$ et $t_2 \in Sol_{\mathcal{T}}(\mathcal{S}, \xi_1) \setminus \{\{\!|\}\!\}$. Soient $\mathbf{d}_1 = \mathsf{dcmp}(t_1, \Theta_{\mathcal{S}})$ et $\mathbf{d}_2 = \mathsf{dcmp}(t_2, \Theta_{\mathcal{S}})$ ; nécessairement $\mathbf{d}_1$ et $\mathbf{d}_2$ sont tous les deux différents de $0^{Q_{\mathcal{S}}}$ par conséquent $\mathbf{d}_1 <_{\mathbf{d}} \mathbf{d}$ et $\mathbf{d}_2 <_{\mathbf{d}} \mathbf{d}$. D'autre part, comme $\mathsf{dcmp}(s, \Theta_{\mathcal{S}}) = \mathbf{d}$, par le

fait 6.3 il existe deux arbres $s_1, s_2$ tels que $s = s_1 \uplus s_2$, $\mathsf{dcmp}(s_1, \Theta_\mathcal{S}) = \mathbf{d}_1$ et $\mathsf{dcmp}(s_2, \Theta_\mathcal{S}) = \mathbf{d}_2$. Comme $s \notin Sol_\mathcal{T}(\mathcal{S}, \xi_1) \,]\!]^\mathcal{T}\, Sol_\mathcal{T}(\mathcal{S}, \xi_2)$, nécessairement $s_1 \notin Sol_\mathcal{T}(\mathcal{S}, \xi_1)$ ou $s_2 \notin Sol_\mathcal{T}(\mathcal{S}, \xi_2)$. Supposons que $s_i \notin Sol_\mathcal{T}(\mathcal{S}, \xi_i)$, pour $i$ étant 1 ou 2. Alors il suffit de prendre $\xi' = \xi_i$, $(t', s') = (t_i, s_i)$ et $\mathbf{d}' = \mathbf{d}_i$.

Soit $\xi_1 \,[\![\, \xi_2$ la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$. Alors $Sol_\mathcal{T}(\mathcal{S}, \xi) = Sol_\mathcal{T}(\mathcal{S}, \xi_1) \,]\!]^\mathcal{T}\, Sol_\mathcal{T}(\mathcal{S}, \xi_2)$. Par hypothèse, $s \notin Sol_\mathcal{T}(\mathcal{S}, \xi_1) \,]\!]^\mathcal{T}\, Sol_\mathcal{T}(\mathcal{S}, \xi_2)$, donc il existe deux arbres $s_1, s_2$ tels que $s = s_1 \uplus s_2$ et $s_1 \notin Sol_\mathcal{T}(\mathcal{S}, \xi_1) \cup \{\{\!|\}\}$ et $s_2 \notin Sol_\mathcal{T}(\mathcal{S}, \xi_2) \cup \{\{\!|\}\}$. Soient $\mathbf{d}_1 = \mathsf{dcmp}(s_1, \Theta_\mathcal{S})$ et $\mathbf{d}_2 = \mathsf{dcmp}(s_2, \Theta_\mathcal{S})$ ; nécessairement $\mathbf{d}_1$ et $\mathbf{d}_2$ sont tous les deux différents de $0^{Q_\mathcal{S}}$ et par conséquent $\mathbf{d}_1 <_\mathsf{d} \mathbf{d}$ et $\mathbf{d}_2 <_\mathsf{d} \mathbf{d}$. D'autre part, comme $\mathsf{dcmp}(t, \Theta_\mathcal{S}) = \mathbf{d}$, par le fait 6.3 il existe deux arbres $t_1, t_2$ tels que $t = t_1 \uplus t_2$, $\mathsf{dcmp}(t_1, \Theta_\mathcal{S}) = \mathbf{d}_1$ et $\mathsf{dcmp}(t_2, \Theta_\mathcal{S}) = \mathbf{d}_2$. Comme $t \in Sol_\mathcal{T}(\mathcal{S}, \xi_1) \,]\!]^\mathcal{T}\, Sol_\mathcal{T}(\mathcal{S}, \xi_2)$ et $t = t_1 \uplus t_2$, par définition nécessairement $t_1 \in Sol_\mathcal{T}(\mathcal{S}, \xi_1)$ ou $t_2 \in Sol_\mathcal{T}(\mathcal{S}, \xi_2)$. Supposons que $t_i \in Sol_\mathcal{T}(\mathcal{S}, \xi_i)$, pour $i$ étant 1 ou 2. Alors il suffit de prendre $\xi' = \xi_i$, $(t', s') = (t_i, s_i)$ et $\mathbf{d}' = \mathbf{d}_i$.

□

**Lemme 6.14** *Pour tout système d'équations simplifié $\mathcal{S}$ et pour toute variable $\xi$ dans* $\mathsf{Vars}(\mathcal{S})$,

$$Sol_\mathcal{T}(\mathcal{S}, \xi) = [\![ Sol_\mathcal{R}(\mathcal{S}^{num}, \xi), \Theta_\mathcal{S} ]\!].$$

*Preuve* Comme pour le lemme précédent, nous faisons la preuve pour les systèmes d'équations normalisés, en utilisant le fait que tout système d'équations simplifié sur $fonct(\tau, \mathcal{X}_\mathsf{r})$ (respectivement sur $fonct(\rho, \mathcal{X}_\mathsf{r})$) peut être transformé en un système normalisé équivalent sur $fonct(\tau', \mathcal{X}_\mathsf{r})$ (respectivement sur $fonct(\rho', \mathcal{X}_\mathsf{r})$). La preuve du présent lemme utilise plusieurs arguments qui ont servi dans la preuve du lemme 6.13 ; par conséquent nous donnerons pas ici une preuve complètement détaillée et nous contenterons parfois d'orienter le lecteur vers des parties de la preuve du lemme 6.13.

Pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, soit $D_\xi = \mathsf{dcmp}(Sol_\mathcal{T}(\mathcal{S}, \xi), \Theta_\mathcal{S})$. Par le lemme 6.13, nous savons que la base $\Theta_\mathcal{S}$ est discriminante pour $Sol_\mathcal{T}(\mathcal{S}, \xi)$, et donc $[\![ D_\xi, \Theta_\mathcal{S} ]\!] = Sol_\mathcal{T}(\mathcal{S}, \xi)$.

Nous commençons par énoncer quelques résultats intermédiaires.

**Fait 6.4** Si la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est l'un parmi $\top, \bot, \mathbf{0}, \overline{\mathbf{0}}$ et $\alpha[\xi']$, alors $[\![ Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \xi), \Theta_\mathcal{S} ]\!] = Sol_\mathcal{R}(\mathcal{S}, \xi)$.

Si la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est parmi $\top, \bot, \mathbf{0}, \overline{\mathbf{0}}$, la propriété est conséquence immédiate de la définition du système d'équations $\mathcal{S}^{\mathrm{num}}$, de la définition de l'interprétation $\mathcal{R}$ présentée sur la figure 1 et du lemme 6.10. Si $\xi$ est une variable élémentaire, alors l'équation pour $\xi$ dans $\mathcal{S}^{\mathrm{num}}$ est $\xi \stackrel{\kappa}{=} C_{\mathsf{prj}(\xi)}$ où, par définition, $\mathsf{prj}(\xi) = \bigcup_{q \in Q_\mathcal{S} | q(\xi) = \mathsf{Pos}(\xi)} \{1_q\}$. Par définition même de la base $\Theta_\xi$ (voir définition 6.11), il est facile de voir que $\Theta_\xi$ est discriminante pour l'ensemble $Sol_\mathcal{T}(\mathcal{S}, \xi)$ et que la décomposition de cet ensemble sur la base $\Theta_\xi$ est l'ensemble à un élément $\{(\mathsf{Pos}(\xi) \mapsto 1, \mathsf{Neg}(\xi) \mapsto 0, \mathsf{Compl}(\xi) \mapsto 0)\}$. Par le lemme 6.9 et par définition de $\Theta_\mathcal{S}$, nous savons que la base $\Theta_\mathcal{S}$ est discriminante pour $Sol_\mathcal{T}(\mathcal{S}, \xi)$ et, toujours par le même lemme, la décomposition de cet ensemble sur la base $\Theta_\mathcal{S}$, càd $D_\xi$, est exactement $\mathsf{prj}(\xi)$. Ceci termine la preuve du fait 6.4

**Fait 6.5** Soit $\mathbf{d}$ un multiensemble dans $\mathbb{N}^{Q_\mathcal{S}}$ et $\xi$ une variable dans $\mathsf{Vars}(\mathcal{S})$. Si $\mathbf{d} \in D_\xi$ et $\mathbf{d} \notin Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \xi)$ (respectivement si $\mathbf{d} \notin D_\xi$ et $\mathbf{d} \in Sol_\mathcal{R}(\mathcal{S}^{\mathrm{num}}, \xi)$), alors il existe une

variable $\xi'$ dans $\mathsf{Vars}(\mathcal{S})$ telle que $\mathbf{d} \in D_{\xi'}$, $\mathbf{d} \notin Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi')$ (respectivement $\mathbf{d} \notin D_{\xi}$, $\mathbf{d} \in Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi')$) et la partie droite de l'équation pour $\xi'$ dans $\mathcal{S}$ est une formule à composition.

Les preuves des deux propriétés énoncées étant très similaires, nous ne donnons ici que la preuve de l'une d'entre elles. Supposons donc que $\xi$ est une variable dans $\mathsf{Vars}(\mathcal{S})$ telle que $\mathbf{d} \in D_{\xi}$ et $\mathbf{d} \notin Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$ et soit $f$ la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$. Si $f$ est un terme à composition, nous n'avons rien à montrer. Par le fait 6.4, $f$ ne peut être de la forme $\top, \bot, \mathbf{0}, \overline{\mathbf{0}}$ et $\xi$ n'est pas une variable élémentaire. Si $f$ est un terme booléen, alors il existe une variable $\xi''$ apparaissant dans $f$ (et donc aussi dans la partie droite de l'équation pour $\xi$ dans $\mathcal{S}^{\mathrm{num}}$) telle que $\mathbf{d} \in D_{\xi''}$ et

($\star$) $\mathbf{d} \notin Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi'')$.

Ceci est facile à montrer par récurrence sur la structure de $f$ (avec des arguments similaires à ceux utilisés pour la preuve du fait 6.1 énoncé dans la preuve du lemme 6.13) et en utilisant le lemme 6.10. On peut alors compléter la preuve en utilisant que $\mathcal{S}$ est sans cycles de dépendance booléenne (et donc $\mathcal{S}^{\mathrm{num}}$ et sans cycles de dépendance booléenne) et en itérant ($\star$), le mécanisme étant similaire à celui utilisé dans la preuve du fait 6.2 énoncé dans la preuve du lemme 6.13. Ceci termine la preuve du fait 6.5.

Revenons à la preuve du lemme, et supposons le contraire : soit $\mathbf{d}$ un plus petit multiensemble (pour l'inclusion de multiensembles, notée $<_{\mathbf{d}}$ ici) tel qu'il existe une variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$ pour laquelle $D_{\xi}$ et $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$ diffèrent par $\mathbf{d}$, càd $\mathbf{d} \in D_{\xi} \cup Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$ mais $\mathbf{d} \notin D_{\xi} \cap Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$. Par le fait 6.5 nous pouvons supposer que la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$ est une formule à composition gardée ; soit cette formule $\xi_1 \int \xi_2$, le cas $\xi_1 \int\!\!\!\int \xi_2$ étant très similaire par symétrie.

Supposons d'abord que $\mathbf{d} \in D_{\xi}$ et $\mathbf{d} \notin Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$. Soit $t$ un arbre appartenant à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ et tel que $\mathsf{dcmp}(t, \Theta_{\mathcal{S}}) = \mathbf{d}$. Alors il existe deux arbres $t_1, t_2$ tels que $t_1 \in Sol_{\mathcal{T}}(\mathcal{S}, \xi_1) \setminus \{\{\!|\}\!\}$ et $t_2 \in Sol_{\mathcal{T}}(\mathcal{S}, \xi_2) \setminus \{\{\!|\}\!\}$ et $t = t_1 \uplus t_2$. Soient $\mathbf{d}_1 = \mathsf{dcmp}(t_1, \Theta_{\mathcal{S}})$ et $\mathbf{d}_2 = \mathsf{dcmp}(t_2, \Theta_{\mathcal{S}})$, alors $\mathbf{d}_1$ et $\mathbf{d}_2$ sont tous deux différentes de $0^{Q_{\mathcal{S}}}$. D'autre part, comme la base $\Theta_{\mathcal{S}}$ est discriminante pour $Sol_{\mathcal{T}}(\mathcal{S}, \xi_1)$ et $Sol_{\mathcal{T}}(\mathcal{S}, \xi_2)$, nous en déduisons que $\mathbf{d}_1 \in D_{\xi_1}$ et $\mathbf{d}_2 \in D_{\xi_2}$. Considérons maintenant $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_1)$ et $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_2)$. Par définition, nous savons que $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi) = Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_1) + Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_2)$. Comme, par hypothèse, $\mathbf{d}$ n'appartient pas à $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$, et comme $\mathbf{d} = \mathbf{d}_1 + \mathbf{d}_2$, nous en déduisons que $\mathbf{d}_1$ n'appartient pas à $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_1)$ ou $\mathbf{d}_2$ n'appartient pas à $Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_2)$. Ceci constitue une contradiction avec la minimalité de $\mathbf{d}$.

Supposons maintenant que $\mathbf{d} \in Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi)$ et $\mathbf{d} \notin D_{\xi}$. Rappelons que l'équation dans $\mathcal{S}$ pour la variable $\xi$ est $\xi_1 \int \xi_2$, donc l'équation pour cette variable pour $\mathcal{S}^{\mathrm{num}}$ est $\xi_1 \int \xi_2$ et alors

$$Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi) = Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_1) \setminus \{0^{Q_{\mathcal{S}}}\} + Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_2) \setminus \{0^{Q_{\mathcal{S}}}\}.$$

Soient $\mathbf{d}_1 \in Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_1) \setminus \{0^Q_{\mathcal{S}}\}$ et $\mathbf{d}_2 \in Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \xi_2) \setminus \{0^Q_{\mathcal{S}}\}$ tels que $\mathbf{d} = \mathbf{d}_1 + \mathbf{d}_2$. De l'hypothèse que $\mathbf{d} \notin D_{\xi}$, nous déduisons que $\mathbf{d}_1 \notin D_{\xi_1}$ ou $\mathbf{d}_2 \notin D_{\xi_2}$, et comme $\mathbf{d}_1$ et $\mathbf{d}_2$ sont tous deux différents de $0^{Q_{\mathcal{S}}}$, ceci est une contradiction avec la minimalité de $\mathbf{d}$.

□

**Lemme 6.15** *Soit $\mathcal{S}$ un système d'équations simplifié. Pour tout état $q$ dans $Q_{\mathcal{S}}$, $\mathcal{L}_{A_{\mathcal{S}}}(q) = \Theta_{\mathcal{S}}(q)$ et pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^{Q_{\mathcal{S}}}$, $\mathcal{L}_{A_{\mathcal{S}}}(D) = [\![D, \Theta_{\mathcal{S}}]\!]$.*

*Preuve* Nous définissons la hauteur d'un élément d'arbre par : $hauteur(e) = hauteur(\{\!|e|\!\})$. Pour tout entier naturel $n$, soit $\mathcal{A}_n$ l'ensemble des arbres de hauteur au plus $n$ et soit $\mathcal{EA}_n$ l'ensemble des éléments d'arbres de hauteur au plus $n$. Nous montrons que pour tout entier naturel $n$, pour tout état $q$ dans $Q_{\mathcal{S}}$ et pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^{Q_{\mathcal{S}}}$

(i) $\mathcal{L}_{A_{\mathcal{S}}}(q) \cap \mathcal{EA}_n = \Theta_{\mathcal{S}}(q) \cap \mathcal{EA}_n$ et

(ii) $\mathcal{L}_{A_{\mathcal{S}}}(D) \cap \mathcal{A}_n = [\![D, \Theta_{\mathcal{S}}]\!] \cap \mathcal{A}_n$.

La preuve va par récurrence sur $n$. Dans cette preuve, nous utilisons les définitions de $Q_{\mathcal{S}}$, l'ensemble des états de l'automate $A_{\mathcal{S}}$ défini dans la section 6.2.1, les définitions des ensembles d'étiquettes $\alpha_q$ et des ensembles de multiensembles $D_q$ (pour $q$ état de $A_{\mathcal{S}}$) définis dans la section 6.2.3, ainsi que des bases $\Theta_\xi$, pour $\xi$ variables élémentaire dans $\mathcal{S}$, définies au début de cette section.

Cas de base: $n = 0$, pour (i) nous n'avons rien à montrer puisque $\mathcal{EA}_0 = \varnothing$. Pour (ii), $\mathcal{A}_0 = \{\{\!|\}\!\}$. Par définition de $\mathcal{L}_{A_{\mathcal{S}}}(D)$, il est facile de voir $\{\!|\}\!\} \in \mathcal{L}_{A_{\mathcal{S}}}$ si et seulement si $0^{Q_{\mathcal{S}}} \in D$. D'autre part, par le lemme 6.10, $0^{Q_{\mathcal{S}}} \in D$ si et seulement si $\{\!|\}\!\} \in [\![D, \Theta_{\mathcal{S}}]\!]$.

Cas d'induction: $n > 0$. Pour (i), par définition,

$$\mathcal{L}_{A_{\mathcal{S}}}(q) = \bigcup_{(q,\alpha,D)\in\Delta_{\mathcal{S}}} \alpha[\mathcal{L}_D],$$

où $\Delta_{\mathcal{S}}$ désigne la relation de transitions de l'automate $A_{\mathcal{S}}$. Par définition de $\Delta_{\mathcal{S}}$ (voir page 114), nous savons que pour tout état $q$, la seule transition dans $\Delta_{\mathcal{S}}$ faisant intervenir $q$ est la transition $(q, \alpha_q, D_q)$. Nous en déduisons que $\mathcal{L}_{A_{\mathcal{S}}}(q) = \alpha_q[\mathcal{L}_{A_{\mathcal{S}}}(D_q)]$. Donc, $\mathcal{L}_{A_{\mathcal{S}}}(q) \cap \mathcal{EA}_n = \alpha_q[\mathcal{L}_{A_{\mathcal{S}}}(D_q)]$ et il est facile de voir que ceci est égal à $\alpha_q[\mathcal{L}_{A_{\mathcal{S}}}(D_q) \cap \mathcal{A}_{n-1}]$. Par hypothèse de récurrence, nous déduisons que

$$\mathcal{L}_{A_{\mathcal{S}}}(q) \cap \mathcal{EA}_n = \alpha_q[[\![D_q, \Theta_{\mathcal{S}}]\!] \cap \mathcal{A}_{n-1}] \tag{6.3}$$

Considérons maintenant l'ensemble $\Theta_{\mathcal{S}}(q)$. Par définition, celui-ci est égal à :

$$\Theta_{\mathcal{S}}(q) = \bigcap_{\xi\in\{\mathsf{EVars}(\mathcal{S})\}} \Theta_\xi(q(\xi)).$$

En utilisant les définitions des ensembles $\Theta_\xi(\mathsf{Pos}(\xi))$, $\Theta_\xi(\mathsf{Neg}(\xi))$ et $\Theta_\xi(\mathsf{Compl}(\xi))$, nous déduisons que

$$\begin{aligned}\Theta_{\mathcal{S}}(q) = &\bigcap_{\xi \mid q(\xi)=\mathsf{Pos}(\xi)} \mathsf{etq}(\xi)[Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{var}(\xi))] \\ \cap &\bigcap_{\xi \mid q(\xi)=\mathsf{Neg}(\xi)} \mathsf{etq}(\xi)[\mathcal{A}_\Sigma \setminus Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{var}(\xi))] \\ \cap &\bigcap_{\xi \mid q(\xi)=\mathsf{Compl}(\xi)} (\Sigma \setminus \mathsf{etq}(\xi))[\mathcal{A}_\Sigma].\end{aligned}$$

Par le lemme 6.14, $Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{var}(\xi)) = [\![Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \mathsf{var}(\xi)), \Theta_{\mathcal{S}}]\!]$, et donc, par le lemme 6.10, $\mathcal{A}_\Sigma \setminus Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{var}(\xi)) = [\![\mathbb{N}^{Q_{\mathcal{S}}} \setminus Sol_{\mathcal{R}}(\mathcal{S}^{\mathrm{num}}, \mathsf{var}(\xi)), \Theta_{\mathcal{S}}]\!]$ et $\mathcal{A}_\Sigma = [\![\mathbb{N}^{Q_{\mathcal{S}}}, \Theta_{\mathcal{S}}]\!]$. Maintenant, en utilisant que

- pour tous ensembles de multiensembles $D, D' \subseteq \mathbb{N}^{Q_S}$ on a $[\![D, \Theta_S]\!] \cap [\![D', \Theta_S]\!] = [\![D \cap D', \Theta_S]\!]$ (lemme 6.10) et
- pour tous ensembles d'étiquettes $\beta_1, \ldots, \beta_k$ et tous ensembles d'arbres $T_1, \ldots, T_k$ on a $\beta_1[T_1] \cap \cdots \cap \beta_k[T_k] = (\beta_1 \cap \cdots \cap \beta_k)[T_1 \cap \cdots \cap T_k]$

et en rappelant les définitions des ensembles $\alpha_q$ et $D_q$ (voir page 114), il n'est pas très difficile de déduire que $\Theta_S(q) = \alpha_q[[\![D_q, \Theta_S]\!]]$. Alors $\Theta_S(q) \cap \mathcal{EA}_n = \alpha_q[[\![D_q, \Theta_S]\!] \cap \mathcal{A}_{n-1}]$ et la conclusion pour (i) suit immédiatement de (6.3).

Pour (ii), admettant que l'ensemble $Q$ des états de l'automate $A_S$ est $\{q_1, \ldots, q_k\}$, par définition on a $\mathcal{L}_{A_S}(D) = \bigcup_{\mathbf{d} \in D} \mathbf{d}(q_1) \cdot \mathcal{L}_{A_S}(q_1) | \cdots | \mathbf{d}(q_k) \cdot \mathcal{L}_{A_S}(q_k)$ et il en suit immédiatement que

$$\mathcal{L}_{A_S}(D) \cap \mathcal{A}_n = \bigcup_{\mathbf{d} \in D} \mathbf{d}(q_1) \cdot (\mathcal{L}_{A_S}(q_1) \cap \mathcal{EA}_n) | \cdots | \mathbf{d}(q_k) \cdot (\mathcal{L}_{A_S}(q_k) \cap \mathcal{EA}_n).$$

Par (i) on en déduit que

$$\mathcal{L}_{A_S}(D) \cap \mathcal{A}_n = \bigcup_{\mathbf{d} \in D} \mathbf{d}(q_1) \cdot (\Theta_S(q_1) \cap \mathcal{A}_n) | \cdots | \mathbf{d}(q_k) \cdot (\Theta_S(q_k) \cap \mathcal{A}_n)$$

De la définition de l'interprétation des ensembles de multiensembles sur un base (voir définition 6.7), il est facile de voir que la partie droite de cette égalité est $[\![D, \Theta_S]\!] \cap \mathcal{A}_n$.

□

Ceci termine les preuves des lemmes nécessaires pour établir la proposition 6.12, statuant que la construction de l'automate $A_S$ est correcte.

Nous avons donc défini, pour tout système d'équations simplifié $\mathcal{S}$, l'automate à contraintes numériques $A_S$ tel que le langage de $A_S$ est la solution du système de $\mathcal{S}$ pour sa dernière variable. Soit maintenant $\phi$ une formule close de $\mathrm{LS}_{|\exists}$ et soit $\mathcal{S}$ le système d'équations simplifié construit à partir de $\phi$ tel que $[\![\phi]\!]$ est la solution de $\mathcal{S}$ pour sa dernière variable. Nous avons décrit dans le chapitre 4 comment le système $\mathcal{S}$ peut être défini. Alors l'automate $A_S$ est tel que le langage de $A_S$ est $[\![\phi]\!]$. Nous en déduisons que

**Théorème 6.16** *Pour toute formule close $\phi$ de la logique $LS_{|\exists}$, on peut définir un automate à contraintes numériques $A$ tel que $[\![\phi]\!] = \mathcal{L}(A)$.*

Dans la section suivante, nous nous intéressons au test d'appartenance pour un tel automate $A$.

## 6.4 Test d'appartenance pour l'automate équivalent à une formule

Rappelons que le test d'appartenance est, étant donné un automate $A$ et un arbre $t$, décider si $t$ appartient au langage de $A$. Nous montrons dans cette section que le test d'appartenance est décidable lorsque $A$ est l'automate correspondant à une formule de la logique spatiale, tel qu'il a été défini dans ce chapitre. Nous définissons un algorithme pour le test d'appartenance, et nous montrons que cet algorithme est polynomial dans la taille de l'arbre et exponentiel dans le nombre d'états de l'automate.

Dans la section 2.3.5, nous avons établi des conditions suffisantes sur un automate à contraintes numériques $A = (\Sigma, Q, \Delta, F)$ pour que le test d'appartenance soit décidable pour cet automate. Ces conditions sont :

- on peut énumérer, en un temps fini, les triplets $(q, \alpha, D)$ appartenant à $\Delta$,
- pour toute étiquette $a$ et tout ensemble d'étiquettes $\alpha$ tel que $(q, \alpha, D)$ est une transition de l'automate, on peut décider en un temps fini si $a$ appartient à $\alpha$ et
- pour tout multiensemble $\mathbf{d}$ dans $\mathbb{N}^Q$ et pour tout ensemble de multiensembles $D \subseteq \mathbb{N}^Q$ dans
$$\{F\} \cup \{D \mid (q, \alpha, D) \text{ dans } \Delta\}$$
on peut décider en un temps fini si $\mathbf{d}$ appartient à $D$.

Lorsque ces conditions sont satisfaites, le test d'appartenance pour l'arbre $t$ et l'automate $A$ peut s'effectuer en calculant une exécution acceptante (si elle existe) de $A$ sur $t$, exécution calculée de manière ascendante. Nous avons également établi que le coût en temps de calcul d'une telle exécution est en $\mathcal{O}(|t| \times |\Delta| \times f(A,t))$, où $f(A,t)$ est la complexité au pire des cas pour tester si un multiensemble $\mathbf{d}$ appartient à un des ensembles $D$ apparaissant dans une transition de l'automate $A$. Une fois qu'une exécution $r$ est calculée, il nous faut vérifier si elle est acceptante, en vérifiant si le multiensemble d'états $\{\!|r(e_1), \ldots, r(e_k)|\!\}$ appartient à la condition d'acceptation $F$, où $e_1, \ldots, e_k$ sont les arêtes successeurs de la racine de l'arbre $t$. Donc, au total, la complexité du test d'appartenance pour l'arbre $t$ et l'automate $A$ est en $\mathcal{O}(|t| \times |\Delta| \times f(A,t)) + f(A,t)$, ce qui est égal à $\mathcal{O}(|t| \times |\Delta| \times f(A,t))$.

Intéressons nous à présent au cas où l'automate $A$ est l'automate correspondant à une formule logique. Pour la reste de la section, nous posons $\phi$ une formule fixée, $t$ un arbre fixé, $\mathcal{S}$ le système d'équations simplifié construit à partir de $\phi$, $\mathcal{S}^{\text{num}}$ le système d'équations numérique correspondant à $\mathcal{S}$ et $\mathcal{S}^{\text{num}}_{\text{n}}$ le système d'équations numérique normalisé correspondant à $\mathcal{S}^{\text{num}}$. $A = (\Sigma, Q, \Delta, F)$ est l'automate $A_{\mathcal{S}}$. Rappelons que $A$ est un automate déterministe et complet.

Nous adoptons également les notations suivantes : pour une fonction fonc (dans le sens informatique du terme) à paramètres formels $x_1, \ldots, x_k$, nous notons $\mathcal{C}(\text{fonc}, x_1, \ldots, x_k)$ la complexité en temps de la fonction fonc ; les $x_1, \ldots, x_k$ seront utilisés dans ce cas principalement pour leur taille. Pour un système d'équations $\mathcal{S}'$, nous notons $|\mathcal{S}'|$ le nombre de ses variables.

Tout d'abord, en ce qui concerne la décidabilité du test d'appartenance, il est facile de voir par définition que les deux premières conditions données ci-dessus sont satisfaites par l'automate $A$. Il suffit donc de montrer que la troisième condition est également satisfaite. Remarquons d'abord que la troisième condition peut être affaiblie. Dans le calcul d'une exécution ascendante de l'automate $A$ sur l'arbre $t$, les multiensembles $\mathbf{d}$ que nous aurons à considérer sont obtenus à partir du nombre de chacun des états de l'automate $A$ à l'ensemble des arêtes successeurs d'un nœud particulier de l'arbre $t$. Donc, les ensemble $\mathbf{d}$ que nous aurons à considérer satisfont $|\mathbf{d}| \leq |t|$, où $|\mathbf{d}|$ est la somme des multiplicités $\mathbf{d}(q)$ pour tout $q$ état de l'automate $A$, et $|t|$ est la taille de l'arbre $A$, càd son nombre de nœuds. Dans ce qui suit, nous allons donner une fonction permettant d'effectuer le test d'appartenance $\mathbf{d} \in D$ (pour $\mathbf{d}$ tel que $|\mathbf{d}| \leq |t|$ et $D$ apparaissant dans une transition de l'automate ou $D$ étant la condition d'acceptation de $A$) et nous allons en déduire une valeur pour $f(A,t)$.

Notons que la définition de l'automate $A$ contient les système d'équations $\mathcal{S}$ (dans la définition de $Q$) et $\mathcal{S}^{\text{num}}$ (dans la définition de $\Delta$). Qui plus est, la fonction de test d'appartenance que nous allons définir va utiliser les systèmes $\mathcal{S}$ et $\mathcal{S}^{\text{num}}$ tels quels. Pour ces raisons, nous consi-

dérons dans la suite que l'automate $A$ est non pas le quadruplet $(\Sigma, Q, \Delta, F)$, mais le sixtuplet $(\Sigma, Q, \Delta, F, \mathcal{S}, \mathcal{S}^{\text{num}})$.

D'après la définition de l'automate $A$, l'ensemble des transitions de $A$ est $\{(q, \alpha_q, D_q) \mid$ pour $q$ état de $A\}$ et pour tout $q$ état de l'automate, l'ensemble $D_q$ est

$$\bigcap_{\xi \mid q(\xi)=\mathsf{Pos}(\xi)} Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}, \mathsf{var}(\xi)) \cap \bigcap_{\xi \mid q(\xi)=\mathsf{Neg}(\xi)} \mathbb{N}^{Qs} \setminus Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}, \mathsf{var}(\xi))$$

pour un certain $q$ état de l'automate $A$ où les $\mathsf{var}(\xi)$ sont des variables dans $\mathsf{Vars}(\mathcal{S}^{\text{num}})$. Supposons que nous disposons d'une fonction appartient_solution$(\mathbf{d}, \mathcal{S}^{\text{num}})$ qui retourne une table $tab$ indexée par $\mathsf{Vars}(\mathcal{S}_{\text{n}}^{\text{num}})$ telle que pour tout $\xi$ dans $\mathsf{Vars}(\mathcal{S}_{\text{n}}^{\text{num}})$, $tab(\xi) = \mathsf{vrai}$ si $\mathbf{d}$ appartient à $Sol_{\mathcal{R}}(\mathcal{S}_{\text{n}}^{\text{num}}, \xi)$ et $tab(\xi) = \mathsf{faux}$ sinon. En utilisant le fait que les systèmes d'équations $\mathcal{S}^{\text{num}}$ et $\mathcal{S}_{\text{n}}^{\text{num}}$ sont équivalents sur l'ensemble des variables $\mathsf{Vars}(\mathcal{S}^{\text{num}})$, cette fonction peut être utilisée pour savoir si $\mathbf{d}$ appartient ou non à $Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}, \xi)$ pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{\text{num}})$. Les raisons d'utiliser le système d'équations $\mathcal{S}_{\text{n}}^{\text{num}}$ et non pas $\mathcal{S}^{\text{num}}$ deviendront claires par la suite. Maintenant, si dans l'expression définissant $D$ nous remplaçons $Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}, \xi)$ par $tab(\xi)$ et les opérations sur les ensembles par les opérations booléennes correspondantes, nous obtenons une expression booléenne close dont la valeur est vrai si et seulement si $\mathbf{d}$ appartient à $D$. C'est à dire, le multiensemble $\mathbf{d}$ appartient à l'ensemble $D_q$ si et seulement si

$$\bigwedge_{\xi \mid q(\xi)=\mathsf{Pos}(\xi)} tab(\mathsf{var}(\xi)) \wedge \bigwedge_{\xi \mid q(\xi)=\mathsf{Neg}(\xi)} \neg tab(\mathsf{var}(\xi))$$

La construction et évaluation de cette expression booléenne sont linéaires dans la définition de l'ensemble $D$, qui elle-même est linéaire dans $|\mathcal{S}|$. Nous en déduisons que $f(A,t) = \mathcal{O}(|\mathcal{S}|) + C(\text{appartient\_solution}, \mathbf{d}, \mathcal{S}^{\text{num}})$, où $\mathbf{d}$ est n'importe quel multiensemble tel que $|\mathbf{d}| \leq |t|$.

Regardons maintenant comment la fonction appartient_solution peut être définie.

```
1.   appartient_solution(d, S^num) =
2.     soit S_n^num le système d'équations normalisé obtenu à partir de S^num ;
3.     soit P ⊆ N^Q l'ensemble {d' | d' ⊆ d} ;
4.     TAB : table de booléens à deux dimensions indicées par P et par Vars(S_n^num) ;
5.     remplir(TAB, S_n^num) ;
6.     retourner TAB(d) ;
```

Ici, $\subseteq$ indique l'inclusion large de multiensembles. La table $TAB$ est une table à deux dimensions dont les lignes sont indicées par les multiensembles $\mathbf{d}'$ tels que $\mathbf{d}' \subseteq \mathbf{d}$ et les colonnes sont indicées par les variables du système d'équations $\mathcal{S}_{\text{n}}^{\text{num}}$. Ainsi, $TAB(\mathbf{d})$ est une table à une dimension indicée par l'ensemble de variables $\mathsf{Vars}(\mathcal{S}_{\text{n}}^{\text{num}})$. La fonction remplir ci-dessus remplit la table $TAB$ de façon à ce que pour tout multiensemble $\mathbf{d}'$ dans $P$ et pour toute variable $\xi$ dans le système $\mathcal{S}_{\text{n}}^{\text{num}}$, $TAB(\mathbf{d}', \xi) = \mathsf{vrai}$ si $\mathbf{d}'$ appartient à $Sol_{\mathcal{R}}(\mathcal{S}_{\text{n}}^{\text{num}}, \xi)$ et $TAB(\mathbf{d}', \xi) = \mathsf{faux}$ sinon. Cette fonction sera définie par un algorithme de programmation dynamique présenté dans la suite. Mais avant cela, faisons quelques remarques relatives à la complexité. La taille du système d'équations $\mathcal{S}_{\text{n}}^{\text{num}}$ est linéaire dans $|\mathcal{S}^{\text{num}}|$. En ce qui concerne sa construction, elle est en $\mathcal{O}(2^{|\mathcal{S}^{\text{num}}|})$. Sans rentrer dans les détails, donnons quelques arguments pour justifier cette affirmation. Pour la construction de $\mathcal{S}_{\text{n}}^{\text{num}}$, nous devons résoudre un système d'équations $\mathcal{S}'$ dans l'algèbre de Boole. La taille de $\mathcal{S}'$ est équivalente à la taille de $\mathcal{S}^{\text{num}}$. Pour résoudre $\mathcal{S}'$, il suffit

de construire tous les $n$-uplets de Booléens ($2^{|\mathcal{S}^{\text{num}}|}$ en tout) et tester lequel est solution du système. Ce test se fait en temps linéaire dans $|\mathcal{S}^{\text{num}}|$. [4] Quant à la construction de l'ensemble $P$, (défini en ligne 3. de l'algorithme), son cardinal est le produit des composantes non nulles de $\mathbf{d}$. Donc, $\text{Card}(P) \leq |\mathbf{d}|^{|Q|}$, ce qui est inférieur à $|t|^{|Q|}$. Alors,

$$\mathcal{C}(\text{appartient\_solution}, \mathbf{d}, \mathcal{S}^{\text{num}}) = \mathcal{O}(|\mathcal{S}^{\text{num}}| + 2^{|\mathcal{S}^{\text{num}}|} + |t|^{|Q|}) + \mathcal{C}(\text{remplir}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}).$$

Appelons *variables de base* les variable $\xi$ dans $\mathcal{S}_{\text{n}}^{\text{num}}$ pour lesquelles la partie droite d'équation est une parmi $\top, \bot, \mathbf{0}, \overline{\mathbf{0}}$. Les deux lemmes ci-dessous nous donnent un moyen de remplir les lignes du tableau $TAB$ correspondant à des variables de base ou des variables élémentaires.

**Lemme 6.17** – *Pour tout $\xi$ tel que la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_n^{num}$ est $\top$ et pour tout $\mathbf{d}'$ dans $P$, $TAB(\xi, \mathbf{d}') = \text{vrai}$.*
– *Pour tout $\xi$ tel que la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_n^{num}$ est $\bot$ et pour tout $\mathbf{d}'$ dans $P$, $TAB(\xi, \mathbf{d}') = \text{faux}$.*
– *Pour tout $\xi$ tel que la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_n^{num}$ est $\mathbf{0}$, $TAB(\xi, 0^Q) = \text{vrai}$ et $TAB(\xi, \mathbf{d}') = \text{faux}$ pour tout $\mathbf{d}'$ dans $P$ différent de $0^Q$.*
– *Pour tout $\xi$ tel que la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_n^{num}$ est $\overline{\mathbf{0}}$, $TAB(\xi, 0^Q) = \text{faux}$ et $TAB(\xi, \mathbf{d}') = \text{vrai}$ pour tout $\mathbf{d}'$ dans $P$ différent de $0^Q$.*

*Preuve* Suit de la définition de l'interprétation $\mathcal{R}$ et des propriétés de la solution d'un système d'équations.

□

**Lemme 6.18** *Pour toute variable élémentaire $\xi$ du système $\mathcal{S}_n^{num}$ et pour tout multiensemble $\mathbf{d}'$ dans $P$,*

$$TAB(\xi, \mathbf{d}') = \begin{cases} \text{vrai} & \text{si } \mathbf{d}' = 1_q \text{ et } q(\xi) = \text{Pos}(\xi); \\ \text{faux} & \text{sinon.} \end{cases}$$

*Preuve* Suit de la définition du système d'équations $\mathcal{S}^{\text{num}}$ et du fait que les équations élémentaires de $\mathcal{S}^{\text{num}}$ et $\mathcal{S}_{\text{n}}^{\text{num}}$ sont les mêmes.

□

Notons que pour les variables de base, chacune des cases de la ligne est remplie en temps constant. Pour les variables élémentaires, chacune des cases est remplie au pire des cas en temps linéaire en $|Q|$.

Présentons maintenant la fonction remplir. Le remplissage de la table $TAB$ se fera ligne par ligne, en suivant l'ordre d'inclusion des multiensembles de $P$ indexant les lignes de $TAB$. C'est à dire, c'est la ligne correspondant à $0^Q$ qui sera remplie d'abord, et ensuite, pour tout $\mathbf{d}'$ dans $P$, la ligne de $TAB$ correspondant à $\mathbf{d}'$ pourra être remplie si pour tous $\mathbf{d}'' \subset \mathbf{d}'$, la ligne de $TAB$ correspondant à $\mathbf{d}''$ est déjà remplie. Donc, la fonction remplir est définie par :

[4] Il existe des des algorithmes plus élaborés pour la résolution d'un système d'équations à résoudre dans l'algèbre de Boole ; voir par exemple [Mader, 1997, chapitre 6] pour un tour des algorithmes existants. La grande partie de ces algorithmes sont exponentiels dans le degré d'alternance du système d'équations (càd le nombre d'alternances entre plus petit et plus grand point fixe). Dans notre cas, nous n'avons pas d'autre borne pour le degré d'alternance que la taille de la formule $\phi$ ayant servi pour la construction du système, ou la taille du système lui même. Nous allons nous contenter de cette approximation grossière, puisque, comme nous le verrons plus tard, l'algorithme de programmation dynamique sera de toute façon exponentiel dans la taille du système d'équations.

1. remplir($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$) =
2. soit $<$ un ordre linéaire des multiensembles compatible avec l'ordre d'inclusion des multiensembles ;
3. soit $p$ un tableau contenant exactement les éléments de $P$ et trié pour $<$ ;
4. pour tout $\mathbf{d}'$ dans $p$ pris dans l'ordre faire
5. remplir_variables_de_base($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}, \mathbf{d}'$) ;
6. remplir_variables_elementaires($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}, \mathbf{d}'$) ;
7. remplir_variables_composition($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}, \mathbf{d}'$) ;
8. remplir_variables_booléen($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}, \mathbf{d}'$) ;
9. fin pour ;

Chacune des fonctions auxiliaires remplir_variables_de_base, remplir_variables_elementaires, remplir_variables_composition et remplir_variables_booléen calcule les valeurs pour les cases $TAB(\mathbf{d}', \xi)$ pour $\xi$ étant une variable d'une certaine catégorie. Pour les deux premières fonctions, $\xi$ est une variable de base et une variable élémentaire respectivement. La définition de ces fonctions est triviale à partir des lemmes 6.18 et 6.18. La fonction remplir_variables_composition calcule les valeurs $TAB(\mathbf{d}', \xi)$ pour $\xi$ étant une variable dont la partie droite d'équation dans $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ est un terme à composition et remplir_variables_booléen les valeurs $TAB(\mathbf{d}', \xi)$ pour lorsque la partie droite de $\xi$ dans $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ est un terme booléen.

Nous commençons par la définition de la fonction remplir_variables_booléen :

1. remplir_variables_booléen($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}, \mathbf{d}'$) =
2. soit $B \subseteq P$ l'ensemble des variables $\xi$ dans $\mathrm{Vars}(\mathcal{S}_{\mathrm{n}}^{\mathrm{num}})$ telles que la partie droite d'équation correspondante est un terme booléen ;
3. soit $<_{\mathrm{bool}}$ un ordre linéaire des variables compatible avec l'ordre de dépendance booléenne dans $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ ;
4. soit $b$ un tableau contenant exactement les éléments de $B$ et trié pour $<_{\mathrm{bool}}$ ;
5. pour tout $\xi$ dans $b$ pris dans l'ordre faire
6. soit $f$ la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ ;
7. soit $g$ l'expression booléenne obtenue en remplaçant dans $f$ chaque variable $\xi'$ par $TAB(\mathbf{d}', \xi')$ ;
8. $TAB(\mathbf{d}', \xi)$ := valeur de $g$ ;
9. fin pour ;

Notons que l'ordre $<_{\mathrm{bool}}$ défini à la ligne 3. existe puisque la dépendance booléenne entre variables est une relation acyclique et définit un ordre partiel. L'expression booléenne $g$ définie à la ligne 7. est une expression close puisque, d'une part, la fonction remplir_variables_booléen est appelée après les fonctions remplir_variables_de_base, remplir_variables_elementaires et remplir_variables_composition. D'autre part, grâce à la définition de la dépendance booléenne qui garantit que lorsque $\xi'$ est une variable libre dans $f$, $\xi'$ est avant $\xi$ dans le tableau $b$ et donc la valeur de $TAB(\mathbf{d}', \xi')$ est déjà connue. Donc, la valeur de $g$ qu'on utilise à la ligne 8. est une valeur booléenne parmi vrai ou faux.

Finalement, donnons la définition de la fonction remplir_variables_composition qui a un comportement différent suivant que $\mathbf{d}'$ soit égal ou non au multiensemble $0^Q$. C'est la seule des quatre fonctions spécifique qui va utiliser des valeurs $TAB(\mathbf{d}'', \xi)$ pour $\mathbf{d}''$ étant un multiensemble différent de $\mathbf{d}'$.

1. remplir_variables_composition($TAB, \mathcal{S}_{\mathrm{n}}^{\mathrm{num}}, \mathbf{d}'$) =
2. soit $U \subseteq \{\mathsf{Vars}(\mathcal{S}_{\mathrm{n}}^{\mathrm{num}})\}$ l'ensemble des variables $\xi$ dans $\mathsf{Vars}(\mathcal{S}_{\mathrm{n}}^{\mathrm{num}})$ telles que la partie droite d'équation correspondante est un terme à composition ;
3. soit $u$ la table qui à chaque variable $\xi$ dans $U$ associe le terme $f$ partie droite de l'équation pour $\xi$ dans $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ ;
4. si $\mathbf{d}' = 0^Q$ alors
5. pour tout $\xi$ indice dans $u$ faire
6. si $u(\xi)$ est de la forme $\xi_1 \lceil \xi_2$, alors $TAB(0^Q, \xi) :=$ faux ;
7. sinon $TAB(0^Q, \xi) :=$ vrai ;
8. fin si ;
9. fin pour ;
10. sinon
11. pour tout $\xi$ indice dans $u$ faire
12. si $u(\xi)$ est de la forme $\xi_1 \lceil \xi_2$, alors $TAB(0^Q, \xi) :=$ calcul_comp($TAB, \mathbf{d}', \xi, \xi_1, \xi_2$) ;
13. sinon $TAB(0^Q, \xi) :=$ calcul_comp_dual($TAB, \mathbf{d}', \xi, \xi_1, \xi_2$) ;
14. fin si ;
15. fin pour ;
16. fin si ;

Le système $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ étant normalisé, il n'est pas difficile de voir que cette fonction est correcte, à condition que les fonctions calcul_comp($TAB, \mathbf{d}', \xi, \xi_1, \xi_2$) et calcul_comp_dual($TAB, \mathbf{d}', \xi, \xi_1, \xi_2$) calculent les valeurs attendues. Montrons comment ces deux fonctions peuvent être définies.

La fonction calcul_comp va retourner vrai s'il existe une décomposition du multiensemble $\mathbf{d}$ sous la forme d'une somme de deux multiensembles $\mathbf{d}_1 + \mathbf{d}_2$ pour laquelle $TAB(\mathbf{d}_1, \xi_1) =$ vrai et $TAB(\mathbf{d}_2, \xi_2) =$ vrai ; autrement dit, si $\mathbf{d}_1$ appartient à la solution de $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ pour la variable $\xi_1$ et $\mathbf{d}_2$ appartient à la solution de $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ pour la variable $\xi_2$.

1. calcul_comp($TAB, \mathbf{d}', \xi, \xi_1, \xi_2$) =
2. soit $B \subset P$ l'ensemble $\{\mathbf{d}'' \mid \mathbf{d}'' \subset \mathbf{d}'\}$ ;
3. pour tout $\mathbf{d}_1$ dans $B$ faire
4. soit $\mathbf{d}_2$ tel que $\mathbf{d}_1 + \mathbf{d}_2 = \mathbf{d}'$ ;
5. si $TAB(\mathbf{d}_1, \xi_1) =$ vrai et $TAB(\mathbf{d}_2, \xi_2) =$ vrai alors retourner vrai ; fin si ;
6. fin pour ;
7. retourner faux ;

Cet algorithme est correct. Remarquons qu'il peut être défini grâce au fait que $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ est un système normalisé, ce qui nous permet de considérer uniquement les multiensembles $\mathbf{d}''$ strictement inclus dans $\mathbf{d}'$, et pour lesquels les lignes de la table $TAB$ sont déjà remplies.

Finalement, la fonction calcul_comp_dual va retourner vrai si pour toute décomposition du multiensemble $\mathbf{d}$ sous la forme d'une somme de deux multiensembles $\mathbf{d}_1 + \mathbf{d}_2$, ou bien $TAB(\mathbf{d}_1, \xi_1) =$ vrai ou bien $TAB(\mathbf{d}_2, \xi_2) =$ vrai ; autrement dit, si $\mathbf{d}_1$ appartient à la solution de $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ pour la variable $\xi_1$ ou $\mathbf{d}_2$ appartient à la solution de $\mathcal{S}_{\mathrm{n}}^{\mathrm{num}}$ pour la variable $\xi_2$.

1. calcul_comp($TAB, \mathbf{d}', \xi, \xi_1, \xi_2$) =

2. soit $B \subset P$ l'ensemble $\{\mathbf{d}'' \mid \mathbf{d}'' \subset \mathbf{d}'\}$ ;
3. $b := vrai$ ;
4. pour tout $\mathbf{d}_1$ dans $B$ faire
5. soit $\mathbf{d}_2$ tel que $\mathbf{d}_1 + \mathbf{d}_2 = \mathbf{d}'$ ;
6. $b := b \wedge (TAB(\mathbf{d}_1, \xi_1) \vee TAB(\mathbf{d}_2, \xi_2))$ ;
7. fin pour ;
8. retourner $b$ ;

Cette fonction est correcte.

Il nous reste maintenant à évaluer la complexité des quatre fonctions auxiliaires et en déduire une borne pour la fonction $f(A, t)$. Tout d'abord, pour la fonction remplir :

$$\begin{aligned}\mathcal{C}(\text{remplir}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}) = {} & \mathcal{O}(|TAB|)+ \\ & \mathcal{O}(\text{Card}(P)) \times ( \ \mathcal{C}(\text{remplir\_variables\_de\_base}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}, \mathbf{d}')+ \\ & \quad \mathcal{C}(\text{remplir\_variables\_elementaires}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}, \mathbf{d}')+ \\ & \quad \mathcal{C}(\text{remplir\_variables\_composition}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}, \mathbf{d}')+ \\ & \quad \mathcal{C}(\text{remplir\_variables\_booléen}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}, \mathbf{d}') \\ & \quad )\end{aligned}$$

La complexité de la fonction remplir_variables_de_base est en $\mathcal{O}(|\mathcal{S}_{\text{n}}^{\text{num}}|)$, et la complexité de la fonction remplir_variables_elementaires est en $\mathcal{O}(|\mathcal{S}_{\text{n}}^{\text{num}}| \times |Q|)$. Pour remplir_variables_booléen, sa complexité est en $\mathcal{O}(|\mathcal{S}_{\text{n}}^{\text{num}}|^2)$. Finalement, pour remplir_variables_composition, la complexité est en $\mathcal{O}(|\mathcal{S}_{\text{n}}^{\text{num}}| \times \text{Card}(P))$.

En utilisant que $|\mathcal{S}_{\text{n}}^{\text{num}}| = \mathcal{O}(|\mathcal{S}^{\text{num}}|)$, $\text{Card}(P) = \mathcal{O}(|t|^{|Q|})$ et $|TAB| = \text{Card}(P) \times |\mathcal{S}^{\text{num}}|$,

$$\mathcal{C}(\text{remplir}, TAB, \mathcal{S}_{\text{n}}^{\text{num}}) = \mathcal{O}(|t|^{|Q|}|\mathcal{S}^{\text{num}}|+|t|^{|Q|}\times(|\mathcal{S}^{\text{num}}|+|\mathcal{S}^{\text{num}}||Q|+|\mathcal{S}^{\text{num}}|^2+|\mathcal{S}^{\text{num}}||t|^{|Q|}))$$

ce qui est égal à

$$\mathcal{O}(|t|^{|Q|} \times |\mathcal{S}^{\text{num}}| \times (|Q| + |\mathcal{S}^{\text{num}}| + |t|^{|Q|})).$$

Finalement, en intégrant cela à ce qui a été déjà dit, nous obtenons que c'est également une borne pour la valeur $f(A, t)$. Nous en déduisons que la complexité du test d'appartenance d'un arbre $t$ au langage de l'automate $A$ est en

$$\mathcal{O}(|t||\Delta| \times (2^{|\mathcal{S}^{\text{num}}|} + |t|^{|Q|}|\mathcal{S}^{\text{num}}| \times (|Q| + |\mathcal{S}^{\text{num}}| + |t|^{|Q|}))).$$

**Lemme 6.19** *Pour un arbre t et un automate A donnés, la complexité du test d'appartenance de t au langage de A est au plus exponentielle dans le nombre d'états de l'automate et polynomiale dans la taille de l'arbre, avec degré du polynôme égal au nombre d'états de l'automate.*

Maintenant, d'après la construction de l'automate $A$ et de son ensemble d'états, on sait que le nombre d'états de $A$ est égal à $3^n$ où $n$ est le nombre de variables élémentaires dans le système d'équations $\mathcal{S}$ et est majoré par la taille de la formule $\phi$. Encore par définition de l'automate $A$, le nombre de ses transitions est égal au nombre de ses états, càd $|\Delta| = |Q|$.

De ce résultat nous pouvons déduire une borne supérieure pour la complexité du model checking de la logique $\text{LS}_{|\exists}$. En effet, pour répondre au problème de model checking $t \models \phi$ (pour une formule $\phi$ close), il suffit de construire l'automate équivalent à la formule $\phi$ et de

faire le test d'appartenance. Pour la construction de l'automate, nous construisons d'abord le système d'équations $\mathrm{Eq}(\phi)$, ce qui se fait en un temps linéaire dans la taille de la formule $\phi$. La construction du système simplifié est également linéaire en $|\phi|$. Finalement, la construction et la taille de l'automate $A$ sont tous deux en $\mathcal{O}(3^{|\phi|})$. Toutes ces quantités sont majorées par la complexité du test d'appartenance.

**Corollaire 6.20** *Pour un arbre $t$ et une formule $\phi$ donnés, tester si l'arbre $t$ satisfait la formule $LS_{|\exists}$ close $\phi$ est au plus doublement exponentiel dans la taille de la formule et polynomial dans la taille de l'arbre, avec degré du polynôme en $\mathcal{O}(3^{|\phi|})$.*

Remarquons finalement que ce résultat ne nous permet pas de conclure que la complexité de données du problème de model checking de la logique $\mathrm{LS}_{|\exists}$ est polynomiale, puisque le degré du polynôme dépend de la taille de la formule. Par contre, nous pouvons dire que cette complexité est au pire $\bigcup_{n\in\mathbb{N}} P_n(|t|)$, ou $P_n(t)$ désigne l'ensemble les polynômes en $|t|$ de degré $n$.

# Chapitre 7

# Fragments décidables de LS

Dans ce chapitre nous identifions deux fragments syntaxiques de la logique spatiale : LSrd et LSrpd, pour lesquels la satisfiabilité est décidable. Ces fragments sont obtenus par des restrictions de la récursion. Nous montrons ensuite que la logique LSrd est équivalente à MSO et LSrpd est équivalente à PMSO. Les résultats de décidabilité aussi bien que les résultats d'équivalence sont établis en passant par les automates à contraintes numériques.

Nous commençons par définir les logiques LSrd et LSrpd et dans la section 7.1. Ensuite nous montrons dans la section 7.2 que l'ACN correspondant à une formule LSrd est un ACN sans étoile et que l'ACN correspondant à une formule LSrpd est un ACN semilinéaires, et que ces automate peuvent être effectivement construits. Dans la section 7.3 nous montrons les résultats inverses, càd le langage d'un ACN sans étoile peut être représenté par une formule LSrd et le langage d'un ACN semilinéaires peut être représenté par une formule LSrpd. Encore une fois, le passage de l'automate à la formule est effectif (à condition que l'automate soit effectif). Ceci nous permettra d'établir l'équivalence les logiques LSrd et LSrpd et les classes d'automates ACN sans étoile et ACN semilinéaires respectivement. Finalement, dans la section 7.4 nous établissons l'équivalence des logiques LSrd et MSO et l'équivalence des logiques LSrpd et PMSO en utilisant des résultats d'équivalences entre PMSO et les ACN semilinéaires et MSO et les ACN sans étoile.

## 7.1 Les fragments LSrd et LSrpd

Rappelons qu'un chemin est une séquence d'opérateurs de la signature $\tau$ enrichis avec un rang pour les opérateurs binaires (voir la définition 4.14, page 76).

Nous commençons par définir les logiques LSrd et LSrpd comme des fragments syntaxiques de la logique spatiale, càd dans les deux définitions ci-dessous la formule $\phi$ est supposée écrite sur la syntaxe d'origine.

**Définition 7.1 (Fragment à récursion descendante)** Une *formule à récursion descendante* est une formule $\phi$ telle que pour tout chemin $c_1.\mu\xi.c_2.\xi$ dans $\phi$, le chemin $c_2$ contient au moins un opérateur d'emboîtement $\alpha$ (pour $\alpha \subseteq \Sigma$).

Le *fragment à récursion descendante* de la logique spatiale, noté LSrd, est constitué des formules closes à récursion descendante.

**Définition 7.2 (Fragment à récursion positive/descendante)** Une *formule à récursion positive/descendante* est une formule $\phi$ telle que pour tout chemin $c_1.\mu\xi.c_2.\xi$ dans $\phi$, le chemin $c_2$ satisfait une de ces deux conditions :

- $c_2$ contient un opérateur d'emboîtement $\alpha$ ou
- $c_2$ ne contient pas d'opérateur de négation.

Le *fragment à récursion positive/descendante* de la logique spatiale, noté LSrpd, est constitué des formules closes à récursion positive/descendante.

Le nom du fragment LSrpd doit être lu et compris « fragment à récursion positive *ou* descendante » et non « fragment à récursion positive *et* descendante ». Il est facile de voir de cette définition que le fragment à récursion descendante est inclus dans le fragment à récursion positive/descendante.

L'exemple 7.1 montre des formules de ces deux fragments.

**Exemple 7.1** Soient les formules

$$\begin{aligned}
\phi_1 &= (\mu\xi.\alpha[\xi] \vee \mathbf{0}) \vee \neg(\mu\xi'.\beta[\xi'] \mid \xi' \vee \mathbf{0}) \\
\phi_2 &= \mu\xi.\neg\alpha[\neg\xi] \mid (\xi \vee \beta[\xi]) \vee \mathbf{0} \\
\phi_3 &= \mu\xi.\alpha[\mu\xi'.\xi \mid \beta[\xi'] \vee \alpha[\top]] \wedge \overline{\mathbf{0}} \\
\phi_4 &= \mu\xi.\neg(\neg\xi \mid \neg\alpha[\mathbf{0}]) \wedge \beta[\top]
\end{aligned}$$

Les formules $\phi_1$ et $\phi_2$ sont des formules à récursion positive/descendante, mais ne sont pas des formules à récursion descendante. La formule $\phi_3$ est à récursion descendante. Finalement, la formule $\phi_4$ n'est ni une formule à récursion descendante, ni une formule à récursion positive/descendante.

Les deux lemmes qui suivent donnent des conditions que les formules positives satisfont lorsque celles-ci sont équivalentes à des formules des fragments LSrd et LSrpd.

**Lemme 7.3** *Soit $\Phi$ une formule positive.*

*(i) Si, pour tout chemin $c.\kappa\xi.d.\xi$ dans $\Phi$, le chemin $d$ contient une occurrence d'un opérateur $\alpha$ (pour $\alpha \subseteq \Sigma$), alors $\Phi$ est équivalente à une formule du fragment LSrd ;*

*(ii) Si, pour tout chemin $c.\kappa\xi.d.\xi$ dans $\Phi$, le chemin $d$ satisfait une de ces trois conditions :*
- *$d$ contient au moins une occurrence d'un opérateur $\alpha$ (pour $\alpha \subseteq \Sigma$ fini ou co-fini) ou*
- *$d$ est constitué uniquement d'opérateurs $\vee, \mid$ et d'opérateurs de plus petit point fixe de la forme $\mu\xi'$ ou*
- *$d$ est constitué uniquement d'opérateurs $\wedge, \|$ et d'opérateurs de plus grand point fixe de la forme $\nu\xi'$,*

*alors $\Phi$ est équivalente à une formule du fragment LSrpd.*

*Preuve* Nous donnons uniquement la preuve du cas (ii), la preuve de l'autre cas étant très similaire. Un chemin $d.\xi$ est dit *chemin de récurrence LSrpd-correct* si $d$ est un chemin sur la syntaxe d'origine et de plus $d$ contient une occurrence d'opérateur $\alpha$ ou $d$ ne contient pas de

négation. Soit $\nu\xi.\phi$ une sous formule de $\Phi$. Montrons d'abord que la formule $\psi = \neg\phi\langle\xi \rightarrow \neg\xi\rangle$ est équivalente à une formule $\psi'$ telle que tout chemin $d.\xi$ dans $\psi'$ est un chemin de récurrence LSrpd-correct. La preuve se fait par induction sur la structure de la formule $\phi$.

Notons d'abord que si $d.\xi$ est un chemin dans $\phi$, et que $\nu\xi.\phi$ est sous formule de $\Phi$, alors il existe un chemin $c$ tel que $c.\nu\xi.d.\xi$ est un chemin dans $\Phi$. Comme ce chemin contient l'opérateur $\nu\xi$, par hypothèse sur $\Phi$ nous déduisons que le chemin $d$ est constitué uniquement des opérateurs $\wedge$, $\|$ et des opérateurs de plus grand point fixe. De ce fait, si $\phi$ est de la forme $\mathbf{0}$, $\overline{\mathbf{0}}$, $\top$, $\bot$, $\phi_1 \vee \phi_2$, $\phi_1 \mid \phi_2$ ou $\mu\xi'.\phi_1$, alors $\phi$ ne peut contenir de chemin de la forme $d.\xi$ et nous n'avons rien à montrer.

Si $\phi = \xi$, alors $d$ est le chemin vide et la formule $\psi$ est égale à $\neg\neg\xi$ et est équivalente à $\xi$. Donc la propriété à montrer est vérifiée.

Si $\phi = \phi_1 \wedge \phi_2$, alors par définition de l'opérateur $\wedge$, la formule $\psi$ est équivalente à $\neg\neg(\neg\phi_1\langle\xi \rightarrow \neg\xi\rangle \vee \neg\phi_2\langle\xi \rightarrow \neg\xi\rangle)$, et donc à $\neg\phi_1\langle\xi \rightarrow \neg\xi\rangle \vee \neg\phi_2\langle\xi \rightarrow \neg\xi\rangle$. Maintenant, par hypothèse de récurrence, pour $i$ étant $1, 2$, la formule $\neg\phi_i\langle\xi \rightarrow \neg\xi\rangle$ est équivalente à une formule, soit $\psi_i$, telle que pour tout chemin $d_i.\xi$ dans $\psi_i$, $d_i$ est un chemin de récurrence LSrpd-correct. Donc $\psi_1 \vee \psi_2$ est équivalente à $\psi$ et il est facile de voir qu'elle correspond aux critères pour $\psi'$.

Finalement, si $\phi = \phi_1 \,\|\, \phi_2$ ou $\phi = \nu\xi'.\phi_1$, la preuve est semblable au cas précédent.

Maintenant, par hypothèse sur les chemins de $\Phi$, nous savons que pour toute sous formule $\mu\xi.\phi$ de $\Phi$, si $d$ est un chemin tel que $\mu\xi.\phi(d) = \xi$, alors $d$ est un chemin de récurrence LSrpd-correct.

Considérons maintenant la formule $\Psi$ obtenue à partir de $\Phi$ en remplaçant les opérateurs dérivés par leur définition. Par ce qui a été montré, il est facile de voir que $\Psi$ est équivalente à une formule à récursion positive/descendante $\Psi'$.

□

**Lemme 7.4** *Soit $\Phi$ une formule close.*

*(i) Si $\Phi$ est dans le fragment LSrd, alors pour tout chemin $c.\kappa\xi.d.\xi$ dans* $\mathsf{pos}(\Phi)$*, le chemin $d$ contient au moins un opérateur $\alpha$ (pour $\alpha \subseteq \Sigma$).*

*(ii) Si $\Phi$ est dans le fragment LSrpd, alors pour tout chemin $c.\kappa\xi.d.\xi$ dans* $\mathsf{pos}(\Phi)$*, le chemin $c$ satisfait une de ces trois conditions :*

- *$d$ contient au moins un opérateur $\alpha$ (pour $\alpha \subseteq \Sigma$ fini ou co-fini) ou*
- *$d$ est constitué uniquement des opérateurs $\vee$, $\mid$ ou d'opérateurs de plus petit point fixe de la forme $\mu\xi'$ ou*
- *$d$ est constitué uniquement des opérateurs $\wedge$, $\|$ ou d'opérateurs de plus grand point fixe de la forme $\mu\xi'$.*

*Preuve* Nous ne montrons que le point (ii), la preuve pour l'autre cas étant très similaire.

Pour un chemin $c$, son chemin *dual* est le chemin de même longueur que $c$ et obtenu à partir de $c$ en remplaçant chaque occurrence des opérateurs $\vee, \wedge, \mid, \|$ et les opérateurs de point fixe par leur opérateur dual (càd en remplaçant toute occurrence de $\vee$ par $\wedge$ et inversement, toute occurrence de $\mid$ par $\|$ et inversement, toute occurrence de $\mu\xi'$ par $\nu\xi'$ et inversement). Commençons par mettre en évidence la propriétés des chemins suivante, qui est facilement déduite des définitions :

($\star$) si $c$ est un chemin sans opérateurs constants et qui ne contient pas d'opérateur de négation ni d'opérateur d'emboîtement, alors $\mathsf{pos}(c) = c$ et $\mathsf{neg}(c)$ est le chemin dual à $c$.

Soit maintenant $c.\kappa\xi.d.\xi$ un chemin dans $\mathsf{pos}(\Phi)$. Alors, par le point (ii) du lemme 4.15, il existe un chemin $c'$ dans $\Phi$ tel que $c.\kappa\xi.d.\xi = \mathsf{pos}(c')$. Par définition de la construction de $\mathsf{pos}(c')$, on peut voir que $c'$ est nécessairement de la forme $c'.\mu\xi.d'.\xi$. Maintenant, du point (iii) du lemme 4.15 nous pouvons déduire que de deux choses l'une :

- $\mathsf{pos}(c'.\mu\xi.d'.\xi) = \mathsf{pos}(c'.\mu\xi).\mathsf{pos}(d'.\xi)$ si $c'$ contient un nombre pair de négations ou
- $\mathsf{pos}(c'.\mu\xi.d'.\xi) = \mathsf{pos}(c'.\mu\xi).\mathsf{neg}(d'.\neg.\xi)$ si $c'$ contient un nombre impair de négations.

Maintenant, nous savons que $d'$ est un chemin sur la syntaxe d'origine qui ou bien contient un opérateur $\alpha$, ou bien est sans négation, et donc constitué uniquement des opérateurs $\vee, |$ et d'opérateurs de plus petit point fixe. Dans le premier cas, par définition, $\mathsf{pos}(d'.\xi)$ et $\mathsf{neg}(d'.\neg.\xi)$ contiennent un opérateur $\alpha$. Dans le second cas, par définition, il est facile de voir que comme $d$ est sans négation, $\mathsf{pos}(d'.\xi) = \mathsf{pos}(d').\mathsf{pos}(\xi) = \mathsf{pos}(d').\xi$ et $\mathsf{neg}(d'.\neg.\xi) = \mathsf{neg}(d').\mathsf{neg}(\neg\xi) = \mathsf{neg}(d').\xi$. Par ($\star$), nous savons que $\mathsf{pos}(d') = d'$ et $\mathsf{neg}(d')$ est le chemin dual à $d'$, donc composé uniquement des opérateurs $\wedge, \|$ et d'opérateurs de plus grand point fixe, ce qu'il fallait démontrer

□

## 7.2 Sous-classes d'automates et satisfiabilité pour les logiques LSrd et LSrpd

Nous montrons ici que le problème de satisfiabilité des logiques LSrd et LSrpd est décidable. Nous allons utiliser pour cela l'équivalence entre automates à contraintes numériques et formules de la logique spatiale sans quantification. Plus précisément, nous montrons que pour toute formule $\phi$ dans le fragment LSrd, on peut construire un ACN sans étoile effectif qui reconnaît exactement les arbres satisfaisant la formule $\phi$. De la même façon, pour toute formule $\phi$ dans le fragment LSrpd, on peut construire un ACN semilinéaire effectif qui reconnaît exactement les arbres satisfaisant la formule $\phi$. Ensuite nous utilisons simplement le fait que le vide est décidable pour les ACN sans étoile et semilinéaires effectifs.

Commençons par rappeler quelques notations et notions qui ont été introduites dans les chapitres précédents. Pour toute formule de la logique spatiale sans quantification $\phi$, $\mathsf{Eq}(\phi)$ est le système d'équations construit à partir de cette formule (voir la section 4.2.4). Dans la suite, nous notons $\mathcal{S}_\phi$ le système d'équations simplifié construit à partir de $\mathsf{Eq}(\phi)$ (voir la section 4.3.1), et $\mathcal{S}_\phi^{\mathrm{num}}$ est le système d'équations numérique construit à partir de $\mathcal{S}_\phi$ (voir la section 6.2.2).

Rappelons également qu'un ACN $A = (\Sigma, Q, \Delta, F)$ est dit sans étoile (respectivement semilinéaires) si pour toute transition $(q, \alpha, D)$ dans $\Delta$, l'ensemble d'étiquettes $\alpha$ est fini ou cofini et l'ensemble de multiensembles $D \subseteq \mathbb{N}^Q$ est sans étoile (respectivement semilinéaire). Un ACN sans étoile, respectivement semilinéaire, est dit effectif si tous les ensembles de multiensembles apparaissant dans la définition de l'automate sont représentées comme une union finie d'ensembles linéaires, eux-mêmes représentés par leurs bases et périodes respectives.

Rappelons finalement comment sont définis les multiensembles apparaissant dans les transitions de l'automate correspondant à une formule logique. Par définition, pour toute transition

$(q, \alpha, D)$ de $A$, l'ensemble de multiensembles $D$ est de la forme

$$\bigcap_{\xi \in V} Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}, \mathsf{var}(\xi)) \cap \bigcap_{\xi \in V'} \mathbb{N}^Q \setminus Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}_{\phi}, \mathsf{var}(\xi))$$

où $Q$ est l'ensemble d'états de l'automate, $V, V'$ sont des sous ensembles de $\{\mathsf{Vars}(\mathcal{S}^{\text{num}}_{\phi})\}$, pour chaque $\xi$ dans $V$ ou dans $V'$, $\mathsf{var}(\xi)$ est une variable dans $\mathsf{Vars}(\mathcal{S}^{\text{num}}_{\phi})$ et $Sol_{\mathcal{R}}(\mathcal{S}^{\text{num}}_{\phi})$ est la solution du système numérique $\mathcal{S}^{\text{num}}_{\phi}$. Comme les ensembles de multiensembles sans étoile et les ensembles de multiensembles semilinéaires sont effectivement clos par intersection et par complémentaire, nous avons le suivant

**Lemme 7.5** *Avec les notations ci-dessus,*

- *Si, pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{num}_{\phi})$, $Sol_{\mathcal{R}}(\mathcal{S}^{num}_{\phi}, \xi)$ est un ensemble de multiensembles sans étoile effectif, alors $A$ est un ACN sans étoile effectif.*
- *Si, pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{num}_{\phi})$, $Sol_{\mathcal{R}}(\mathcal{S}^{num}_{\phi}, \xi)$ est un ensemble de multiensembles semilinéaire effectif, alors $A$ est un ACN semilinéaire effectif.*

Donc, pour montrer que l'ACN correspondant à une formule LSrd est un ACN sans étoile effectif et que l'ACN correspondant à une formule LSrpd est un ACN semilinéaire effectif, il suffit de montrer le résultat suivant :

**Lemme 7.6** *Avec les notations ci-dessus,*

- *Si $\phi$ est une formule de la logique LSrd, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{num}_{\phi})$, l'ensemble $Sol_{\mathcal{R}}(\mathcal{S}^{num}_{\phi}, \xi)$ est un ensemble de multiensembles sans étoile qui peut être effectivement calculé.*
- *Si $\phi$ est une formule de la logique LSrpd, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{num}_{\phi})$, l'ensemble $Sol_{\mathcal{R}}(\mathcal{S}^{num}_{\phi}, \xi)$ est un ensemble de multiensembles semilinéaire qui peut être effectivement calculé.*

Le reste de la section est consacré à la preuve de ce lemme.

### 7.2.1 Une méthode abstraite de résolution d'un système d'équations

Nous voulons montrer que les solutions des systèmes d'équations numériques construits à partir d'une formule LSrd ou LSrpd sont respectivement des ensembles sans étoile et semilinéaires et qu'elles peuvent être effectivement calculées. Comme nous l'avons vu dans la section précédente, cela nous permettra de conclure à la satisfiabilité des logiques LSrd et LSrpd.

Pour la résolution des systèmes d'équations, nous allons utiliser des propriétés qu'ont les systèmes numériques lorsqu'ils ont été construits à partir d'une formule des deux fragments de la logique qui nous intéressent. Cependant, ces propriétés ne s'expriment pas de manière globale pour tout le système d'équations, mais caractérisent des groupes d'équations de celui-ci. C'est pourquoi, pour la résolution du système d'équations, nous allons utiliser une méthode qui permet d'exprimer la solution d'un système d'équations à partir des solutions de sous systèmes extraits de celui-ci. Dans ce qui suit nous présentons cette méthode. Notons qu'il s'agit ici d'une méthode générale, puisqu'elle peut être utilisée pour n'importe quel système d'équations satisfaisant certaines propriétés (que nous allons annoncer). Nous en faisons ici une présentation abstraite, dans le sens où nous ne nous intéressons pas à la représentation de la solution du

système d'équations. Par ailleurs, nous illustrons les différentes étapes de cette méthode dans l'exemple 7.2.

Fixons d'abord ce qu'on entend par un sous système d'équations extrait de $\mathcal{S}$ (ou encore système d'équations extrait de $\mathcal{S}$). Il s'agit d'un système d'équations construit à partir d'une ou plusieurs équations de $\mathcal{S}$ non nécessairement contiguës.

**Définition 7.7 (Système extrait)** Soit $\mathcal{S}$ un système d'équations et $V$ un sous-ensemble de $\{\mathsf{Vars}(\mathcal{S})\}$. Le *système extrait de $\mathcal{S}$ induit par $V$* est le sous système de $\mathcal{S}$ contenant exactement les équations pour les variables appartenant à $V$ dans le même ordre que dans $\mathcal{S}$.

Par exemple, si $\mathcal{S}$ est le système d'équations à point fixe

$$\xi_1 \overset{\kappa_1}{=} f_1, \xi_2 \overset{\kappa_2}{=} f_2, \ldots \xi_{2n} \overset{\kappa_{2n}}{=} f_{2n}$$

et $V = \{\xi_i \mid i \text{ est pair}\}$, alors le système extrait de $\mathcal{S}$ induit par $V$ est

$$\xi_2 \overset{\kappa_2}{=} f_2, \xi_4 \overset{\kappa_4}{=} f_4, \ldots \xi_{2n} \overset{\kappa_{2n}}{=} f_{2n}.$$

Soit maintenant $\langle E, \wedge_E, \vee_E \rangle$ un treillis quelconque et soit $\mathcal{S}$ un système d'équations à points fixes sur la séquence de variables $x_1, \ldots, x_k$ et tel que les parties droites d'équations dans $\mathcal{S}$ sont des fonctions monotones de $E^k$ dans $E$. Considérons une partition de $\{\mathsf{Vars}(\mathcal{S})\}$ en $n$ ensembles, que nous notons $V^{(1)}, \ldots, V^{(n)}$. Considérons la relation binaire $\rightsquigarrow_{\mathcal{S}}$ (en notation infixe) entre les ensembles $V^{(1)}, \ldots, V^{(n)}$ appelée *relation de dépendance* et définie par : $V^{(j)} \rightsquigarrow_{\mathcal{S}} V^{(j')}$ si $j \neq j'$ et il existe une variable $x$ dans $V^{(j)}$ et une variable $x'$ dans $V^{(j')}$ telles que dans $\mathcal{S}$, la partie droite de l'équation pour la variable $x$ est une fonction que dépend effectivement de la variable $x'$. Voir l'exemple 7.2 qui illustre cette relation de dépendance.

Supposons maintenant que le graphe de la relation $\rightsquigarrow_{\mathcal{S}}$ est acyclique. Soit $V^{(j)}$ un ensemble qui n'a pas de successeurs par la relation $\rightsquigarrow_{\mathcal{S}}$. Alors nous savons que dans $\mathcal{S}$, les parties droites des équations pour les variables dans $V^{(j)}$ sont des fonctions qui ne dépendent que des variables dans $V^{(j)}$. Considérons alors $\mathcal{S}'$, le système extrait de $\mathcal{S}$ induit par $V^{(j)}$ ; $\mathcal{S}'$ est alors un système d'équations sur l'ensemble de variables $V^{(j)}$. Pour toute variable $x_i$ dans $V^{(j)}$, soit $h'_i$ la solution de $\mathcal{S}'$. Par définition de la solution d'un système d'équations, il est facile de voir que pour toute variable $x_i$ dans $V^{(j)}$, $h'_i$ est également la solution de $\mathcal{S}$ pour la variable $x_i$. Alors, pour connaître la solution de $\mathcal{S}$ pour la variable $x_i$ dans $V^{(j)}$, il suffit de pouvoir résoudre le système extrait $\mathcal{S}'$.

Soit maintenant $V^{(j)}$ un ensemble qui a des successeurs par $\rightsquigarrow_{\mathcal{S}}$, et soit $V$ l'ensemble de variables qui appartiennent à un des ensembles successeurs de $V^{(j)}$, càd $V$ est l'ensemble

$$V = \bigcup_{V^{(j)} \rightsquigarrow_{\mathcal{S}} V^{(j')}} V^{(j')}.$$

Pour toute variable $x_i$ dans $V$, notons par $h_i$ la solution de $\mathcal{S}$ pour la variable $x_i$. Considérons le système d'équations $\mathcal{S}'$ obtenu à partir de $\mathcal{S}$ en remplaçant dans celui-ci toute fonction $f(\mathsf{Vars}(\mathcal{S}))$ apparaissant en partie droite d'équation par la fonction $f'$ sur les variables $\{\mathsf{Vars}(\mathcal{S})\} \setminus V$ obtenue à partir de $f$ en instanciant chaque variable $x_i$ de $V$ par la valeur $h_i$. Par le corollaire 4.6, nous savons que les systèmes $\mathcal{S}'$ et $\mathcal{S}$ ont la même solution.

Soit finalement $\mathcal{S}''$ le système d'équations extrait de $\mathcal{S}'$ enduit par $V^{(j)}$. Par définition de $\mathcal{S}'$ et de $\mathcal{S}''$, on sait que dans $\mathcal{S}''$, toute fonction apparaissant en partie droite d'équation ne dépend

que des variables dans $V^{(j)}$. Donc, $\mathcal{S}''$ est un système d'équations clos. Pour toute variable $x_i$ dans $V^{(j)}$, soit $h''_i$ la solution de $\mathcal{S}''$ pour la variable $x_i$. Par définition de la solution d'un système d'équations, on peut voir que pour tout $x_i$ dans $V^{(j)}$, $h''_i$ est également la solution de $\mathcal{S}'$ pour la variable $x_i$, et par ce qui a été dit à propos de $\mathcal{S}'$, c'est également la solution de $\mathcal{S}$ pour la variable $x_i$.

**Exemple 7.2** Soit le système d'équations $\mathcal{S}$ ci-dessous.

$$\begin{array}{lll} x_1 & \overset{\kappa_1}{=} & f_1(x_1, x_2, x_4, x_6), \\ x_2 & \overset{\kappa_2}{=} & f_2(x_2, x_3, x_5), \\ x_3 & \overset{\kappa_3}{=} & f_3(x_3, x_5), \\ x_4 & \overset{\kappa_4}{=} & f_4(x_3, x_4, x_5), \\ x_5 & \overset{\kappa_5}{=} & f_5(x_3, x_5), \\ x_6 & \overset{\kappa_6}{=} & f_6(x_1, x_2, x_4, x_6) \end{array}$$

Nous admettons que chaque fonction apparaissant en partie droite d'équation ne dépend que des variables qui ont été explicitement données ; par exemple la partie droite de la première équation du système est une fonction qui dépend des variables $x_1, x_2, x_4, x_6$ et ne dépend pas des variables $x_3, x_5$.

Considérons les ensembles de variables

$$V^{(1)} = \{x_1, x_6\} \qquad V^{(2)} = \{x_2\} \qquad V^{(3)} = \{x_3, x_5\} \qquad V^{(4)} = \{x_4\}$$

Ces quatre ensembles forment une partition de $\{\mathsf{Vars}(\mathcal{S})\}$. Les couples d'ensembles de variables mis en relation par $\rightsquigarrow_{\mathcal{S}}$ sont :

$$V^{(1)} \rightsquigarrow_{\mathcal{S}} V^{(4)}, \qquad V^{(1)} \rightsquigarrow_{\mathcal{S}} V^{(2)}, \qquad V^{(4)} \rightsquigarrow_{\mathcal{S}} V^{(3)}, \qquad V^{(2)} \rightsquigarrow_{\mathcal{S}} V^{(3)} \quad (7.1)$$

Cette relation nous renseigne sur les dépendances entre les variables du système d'équations. Par exemple, comme $V^{(1)}$ est en relation avec $V^{(4)}$ et $V^{(2)}$ mais pas avec $V^{(3)}$, nous savons que la partie droite d'équation d'une variable $x$ dans $V^{(1)}$ peut dépendre des variables $V^{(1)} \cup V^{(2)} \cup V^{(4)}$ mais dépend d'aucune variable dans $V^{(3)}$.

L'ensemble de variables $V^{(3)}$ est sans successeur pour $\rightsquigarrow_{\mathcal{S}}$. Le système extrait de $\mathcal{S}$ induit par $V^{(3)}$ est le système à deux équations suivant :

$$\begin{array}{lll} x_3 & \overset{\kappa_3}{=} & f_3(x_3, x_5), \\ x_5 & \overset{\kappa_5}{=} & f_5(x_3, x_5) \end{array}$$

Nous pouvons voir que ce système d'équations est clos ; càd toutes les variables apparaissant en partie droite d'équation sont des variables du système. Soit $\langle h'_3, h'_5 \rangle$ la solution de ce système. Soit $\langle h_1, h_2, h_3, h_4, h_5, h_6 \rangle$ la solution de $\mathcal{S}$. Alors $h'_3 = h_3$ et $h'_5 = h_5$.

L'ensemble de variable $V^{(1)}$ a des successeurs par la relation $\rightsquigarrow_{\mathcal{S}}$, notamment les ensembles $V^{(2)}$ et $V^{(4)}$. Soit $V$ l'ensemble $V^{(2)} \cup V^{(4)}$ ; dans notre exemple $V = \{x_2, x_4\}$. Soient $h_2$ et $h_4$ les solutions de $\mathcal{S}$ pour les variables $x_2$ et $x_4$ respectivement. Soit $\mathcal{S}'$ le système d'équations obtenu à partir de $\mathcal{S}$ en injectant les valeurs $h_2$ et $h_4$ dans les fonctions apparaissant en partie

droite d'équation dans $\mathcal{S}$, càd $\mathcal{S}'$ est le système

$$\begin{array}{lll} x_1 & \stackrel{\kappa_1}{=} & f_1(x_1, h_2, h_4, x_6), \\ x_2 & \stackrel{\kappa_2}{=} & f_2(h_2, x_3, x_5), \\ x_3 & \stackrel{\kappa_3}{=} & f_3(x_3, x_5), \\ x_4 & \stackrel{\kappa_4}{=} & f_4(x_3, h_4, x_5), \\ x_5 & \stackrel{\kappa_5}{=} & f_5(x_3, x_5), \\ x_6 & \stackrel{\kappa_6}{=} & f_6(x_1, h_2, h_4, x_6) \end{array}$$

Par le corollaire 4.6, nous savons que $\mathcal{S}$ et $\mathcal{S}'$ ont la même solution.

Soit finalement $\mathcal{S}''$ le système extrait de $\mathcal{S}'$ induit par $V^{(1)}$ :

$$\begin{array}{lll} x_1 & \stackrel{\kappa_1}{=} & f_1(x_1, h_2, h_4, x_6), \\ x_6 & \stackrel{\kappa_6}{=} & f_6(x_1, h_2, h_4, x_6) \end{array}$$

On peut voir que $\mathcal{S}''$ est un système clos. Soit $\langle h_1'', h_6'' \rangle$ la solution de $\mathcal{S}''$. Par définition de la solution d'un système d'équations nous pouvons voir que $h_1''$ est également la solution de $\mathcal{S}'$ pour la variable $x_1$ et $h_6''$ est également la solution de $\mathcal{S}'$ pour la variable $x_6$. Par ce qui a été dit sur la solution de $\mathcal{S}'$, nous en déduisons que $h_1''$ et $h_6''$ sont respectivement la solution de $\mathcal{S}$ pour la variable $x_1$ et la solution de $\mathcal{S}$ pour la variable $x_6$.

On voit que la solution de $\mathcal{S}$ pourrait être calculée de proche en proche, en calculant d'abord la solution des systèmes extraits induits par les ensembles $V^{(j)}$ et en suivant l'ordre partiel défini par la relation acyclique $\rightsquigarrow_{\mathcal{S}}$. Cette méthode de calcul de la solution d'un système d'équations peut s'avérer utile si les solutions des systèmes extraits induits par les ensembles $V^{(j)}$ sont plus faciles à calculer que la solution du système dans sa totalité. Résumons cette méthode de calcul pour le système $\mathcal{S}$ de manière informelle, sous la forme d'un algorithme abstrait (dans lequel on ne se préoccupe pas de la représentation des objets manipulés) :

- soit *solutions* une liste de couples de la forme $(x, h)$, où $x$ est une variable dans $\mathsf{Vars}(\mathcal{S})$ et $h$ appartient à $E$ et qui ne peut contenir deux couples $(x, h)$ et $(x, h')$ ayant la même première composante. La liste *solutions* est initialement vide ;
- pour chacun des ensembles $V^{(j)}$ pris dans un ordre « remontant » le graphe de la relation $\rightsquigarrow_{\mathcal{S}}$, faire
  - dans $\mathcal{S}$, pour toute fonction $f(x_1, \ldots, x_k)$ apparaissant en partie droite d'équation, remplacer $f(x_1, \ldots, x_k)$ par la fonction $f(y_1, \ldots, y_k)$ où pour tout $i$ dans $1..k$, $y_i = h_i$ si $(x_i, h_i)$ est dans la liste *solutions* et $y_i = x_i$ si la liste *solutions* ne contient pas de couple dont la première composante est $x_i$ ;
  - soit $\mathcal{S}'$ le système d'équations extrait de $\mathcal{S}$ induit par $V^{(j)}$. Nous savons que $\mathcal{S}'$ est un système clos, càd les parties droites d'équations dans $\mathcal{S}'$ sont des fonctions qui ne dépendent que des variables dans $V^{(j)}$ ;
  - résoudre le système $\mathcal{S}'$ ; pour chaque variable $x_i$ dans $V^{(j)}$, soit $h_i'$ la solution de $\mathcal{S}'$ pour cette variable ;
  - pour chaque variable $x_i$ dans $V^{(j)}$, ajouter le couple $(x_i, h_i')$ à la liste *solutions* (on sait qu'avant l'ajout, *solutions* ne contient pas de couple dont la première composante est $x_i$, ceci parce que les ensembles $V^{(1)}, \ldots, V^{(n)}$ forment une partition de $\mathsf{Vars}(\mathcal{S})$).

A la fin du traitement itératif, la liste *solutions* contient le couple $(x_i, h_i)$ pour chaque $x_i$ dans $\mathsf{Vars}(\mathcal{S})$, et $h_i$ est alors la solution de $\mathcal{S}$ pour la variable $x_i$. Nous illustrons dans l'exemple 7.3 les différentes étapes de cette méthode.

**Exemple 7.3** Nous montrons les différentes étapes de la méthode de résolution sur le système d'équations $\mathcal{S}$ défini dans l'exemple 7.2. Initialement, la liste *solutions* est vide. Nous allons parcourir les ensembles $V^{(1)}, \ldots, V^{(4)}$ dans l'ordre suivant : d'abord $V^{(3)}$, ensuite $V^{(2)}$ suivi par $V^{(4)}$ et finalement $V^{(1)}$. Cet ordre de parcours est compatible avec l'ordre partiel défini par la relation $\rightsquigarrow_{\mathcal{S}}$ ci-dessus (7.1). Nous commençons par l'itération pour $V^{(3)}$, en remplaçant dans $\mathcal{S}$ les occurrences des variables présentes dans *solutions* par leur solution. La liste solutions étant vide pour l'instant, le système d'équations $\mathcal{S}$ reste inchangé. Ensuite on construit le système extrait de $\mathcal{S}$ induit par $V^{(3)}$ ; c'est le système suivant

$$\begin{aligned} x_3 &\overset{\kappa_3}{=} f_3(x_3, x_5), \\ x_5 &\overset{\kappa_5}{=} f_5(x_3, x_5) \end{aligned}$$

Soit $\langle h_3, h_5 \rangle$ la solution de ce système. On ajoute à *solutions* les couples $(x_3, h_3)$ et $(x_5, h_5)$. L'itération pour $V^{(3)}$ est terminée, on peut continuer par $V^{(2)}$. Pour la première étape, on injecte dans le système $\mathcal{S}$ les valeurs présentes dans *solutions* ; alors $\mathcal{S}$ devient

$$\begin{aligned} x_1 &\overset{\kappa_1}{=} f_1(x_1, x_2, x_4, x_6), \\ x_2 &\overset{\kappa_2}{=} f_2(x_2, h_3, h_5), \\ x_3 &\overset{\kappa_3}{=} f_3(h_3, h_5), \\ x_4 &\overset{\kappa_4}{=} f_4(h_3, x_4, h_5), \\ x_5 &\overset{\kappa_5}{=} f_5(h_3, h_5), \\ x_6 &\overset{\kappa_6}{=} f_6(x_1, x_2, x_4, x_6) \end{aligned}$$

Le système extrait de $\mathcal{S}$ induit par $V^{(2)}$ est le système à une solution $x_2 \overset{\kappa_2}{=} f_2(x_2, h_3, h_5)$. Soit $\langle h_2 \rangle$ la solution de ce système. On ajoute à *solutions* la couple $(x_2, h_2)$. La prochaine itération est pour $V^{(4)}$. On commence par remplacer dans $\mathcal{S}$ les occurrences de variables par la solution correspondante, lorsque celle-ci est contenue dans *solutions*. Alors le système $\mathcal{S}$ devient

$$\begin{aligned} x_1 &\overset{\kappa_1}{=} f_1(x_1, h_2, x_4, x_6), \\ x_2 &\overset{\kappa_2}{=} f_2(h_2, h_3, h_5), \\ x_3 &\overset{\kappa_3}{=} f_3(h_3, h_5), \\ x_4 &\overset{\kappa_4}{=} f_4(h_3, x_4, h_5), \\ x_5 &\overset{\kappa_5}{=} f_5(h_3, h_5), \\ x_6 &\overset{\kappa_6}{=} f_6(x_1, h_2, x_4, x_6) \end{aligned}$$

Le système extrait de $\mathcal{S}$ induit par $V^{(4)}$ est le système à une équation $x_4 \overset{\kappa_4}{=} f_4(h_3, x_4, h_5)$. Soit $\langle h_4 \rangle$ la solution de ce système. On ajoute à *solutions* la couple $(x_4, h_4)$. La dernière itération est pour $V^{(1)}$. Remplacer dans $\mathcal{S}$ les variables par la valeur correspondante modifie $\mathcal{S}$ en le système suivant :

$$\begin{aligned} x_1 &\overset{\kappa_1}{=} f_1(x_1, h_2, h_4, x_6), \\ x_2 &\overset{\kappa_2}{=} f_2(h_2, h_3, h_5), \\ x_3 &\overset{\kappa_3}{=} f_3(h_3, h_5), \\ x_4 &\overset{\kappa_4}{=} f_4(h_3, h_4, h_5), \\ x_5 &\overset{\kappa_5}{=} f_5(h_3, h_5), \\ x_6 &\overset{\kappa_6}{=} f_6(x_1, h_2, h_4, x_6) \end{aligned}$$

Le système extrait de $\mathcal{S}$ induit par $V^{(1)}$ est alors

$$\begin{array}{rcl} x_1 & \overset{\kappa_1}{=} & f_1(x_1, h_2, h_4, x_6), \\ x_6 & \overset{\kappa_6}{=} & f_6(x_1, h_2, h_4, x_6) \end{array}$$

Soit $\langle h_1, h_6 \rangle$ la solution de ce système. On ajoute à *solutions* les couples $(x_1, h_1)$ et $(x_6, h_6)$.

Après cette dernière itération, la liste *solution* contient les couples $(x_3, h_3)$, $(x_5, h_5)$, $(x_2, h_2)$, $(x_4, h_4)$, $(x_1, h_1)$ et $(x_6, h_6)$. Nous savons alors que pour toute variable $x_i$, $h_i$ est la solution du système $\mathcal{S}$ pour cette variable.

Comme nous l'avons vu, pour que cette méthode de résolution puisse donner lieu à un algorithme de résolution d'un système $\mathcal{S}$, le système $\mathcal{S}$ doit satisfaire certaines conditions. Plus particulièrement, on doit pouvoir partitionner l'ensemble de variables $\mathsf{Vars}(\mathcal{S})$ en $k$ sous ensembles, soient $V^{(1)}, \ldots, V^{(k)}$ de telle manière que :

**C1** la relation $\rightsquigarrow_{\mathcal{S}}$ est acyclique ;

**C2** le système d'équations extrait enduit par chacun des ensembles de variables $V^{(j)}$ puisse être résolu. De plus, la résolution de chacun des sous-systèmes extraits permettent de présenter les solutions de telle manière que ces solutions peuvent être réintégrées dans le système d'équations $\mathcal{S}$.

Dans ce qui suit, nous allons montrer que lorsque $\mathcal{S}$ est le système d'équations numériques correspondant à une formule des fragments LSrd et LSrpd ces conditions sont satisfaites.

### 7.2.2 Résolution effective du système lorsque $\phi$ est une formule des fragments qu'on considère

Dans la section précédente nous avons présenté une méthode abstraite de résolution d'un système d'équations $\mathcal{S}$. Cette méthode peut donner lieu à un algorithme de résolution si $\mathcal{S}$ satisfait les deux conditions **C1** et **C2** ci-dessus. Nous montrons ici que ces deux conditions sont satisfaites lorsque $\mathcal{S}$ est un système d'équations numériques associé à une formule des fragments LSrd ou LSrpd.

Commençons par fixer quelques notations. Dans la suite, considérons la formule $\phi$ que nous supposons appartenir à un des fragments LSrd ou LSrpd. Soit $\mathcal{S}_\phi^{\text{num}}$ le système numérique construit à partir de la formule $\phi$ par la méthode de construction décrite dans la section 6.2.2. Notons également par $Q_\phi$ l'ensemble des états de l'automate reconnaissant le langage $[\![\phi]\!]$, automate défini dans la section 6.2. Rappelons que les solutions du système d'équations $\mathcal{S}_\phi^{\text{num}}$ sont des ensembles de multiensembles sur $Q_\phi$. Considérons finalement la signature

$$\rho^* = \{\bot, \vee, |, *\} \bigcup_{q \in Q_\phi} \{C_{\{1_q\}}\}$$

où $*$ est un symbole unaire (utilisé en notation postfixée) et les autres symboles sont comme dans $\rho$. La $\mu$-interprétation $\mathcal{R}$ est définie pour $\rho^*$ comme pour $\rho$, avec l'interprétation du symbole $*$ qui est donnée par

$$(D*)^{\mathcal{R}} = \bigcup_{n \in \mathbb{N}} \{\{\!\{q_1, \ldots, q_n\}\!\} \mid \forall i \in 1..n.\, q_i \in D\}.$$

Dans la suite, nous aurons à considérer des systèmes d'équations sur $fonct(\rho, \mathcal{X}_r)$ dans lesquels les occurrences de certaines variables sont remplacées par des termes clos dans $fonct(\rho^*, \mathcal{X}_r)$. Pour cela, nous allons définir une notation. Soit $\gamma$ une application de domaine (fini) $V \subset \mathcal{X}_r$, qui à chaque variable de $V$ associe un terme fonctionnel clos sur $\rho^*$. Si $\mathcal{S}$ est un système d'équations sur $fonct(\rho, \mathcal{X}_r)$ et $V$ est l'ensemble des variables apparaissant en partie droite d'équation dans $\mathcal{S}$ mais qui n'appartiennent pas à $\mathsf{Vars}(\mathcal{S})$, alors $\gamma\mathcal{S}$ est le système d'équations obtenu à partir de $\mathcal{S}$ en remplaçant dans celui-ci toute occurrence d'une variable $\xi \in V$ par $\gamma(\xi)$. Dans ce cas, $\gamma\mathcal{S}$ est un système d'équations sur $fonct(\rho \cup \rho^*, \mathcal{X}_r)$. Une telle application $\gamma$ serait également appelée substitution, de par son utilisation.

Pour montrer que les conditions **C1** et **C2** sont satisfaites lorsque $\phi$ est une formule des fragments LSrd et LSrpd, nous passons par ces trois étapes :

**(E1)** Nous proposons d'abord un partitionnement des variables de $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$ pour lequel la relation $\rightsquigarrow_{\mathcal{S}_\phi^{\text{num}}}$ est acyclique ; soit $V^{(1)}, \ldots, V^{(k)}$ ce partitionnement.

**(E2)** Nous montrons que pour tout ensemble de multiensembles semilinéaires $D$, il existe un terme clos $f$ sur $fonct(\rho^*, \mathcal{X}_r)$ tel que $f^{\mathcal{R}} = D$. Le terme $f$ n'est pas unique. Un terme clos satisfaisant cette condition sera noté $f_D$.

**(E3)** Soit $V^{(j)}$ un des ensembles parties de $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$, soit $\mathcal{S}_{V^{(j)}}$ le système extrait de $\mathcal{S}_\phi^{\text{num}}$ induit par $V^{(j)}$, soit $V$ l'ensemble des variables apparaissant en partie droite d'équation dans $\mathcal{S}'$ et qui n'appartiennent pas à $V^{(j)}$, et soit $\gamma$ une substitution de domaine $V$. Nous montrons que :
- pour toute variable $\xi$ dans $V^{(j)}$, la solution de $\gamma\mathcal{S}_{V^{(j)}}$ pour $\xi$ est un ensemble semilinéaire et
- on peut effectivement calculer un terme clos $f$ sur $\rho^*$ tel que $f^{\mathcal{R}} = Sol_{\mathcal{R}}(\gamma\mathcal{S}_{V^{(j)}}, \xi)$.

De plus, dans le cas particulier où $\phi$ est une formule du fragment LSrd et pour toute variable $\xi' \in V$, l'ensemble $(\gamma(\xi'))^{\mathcal{R}}$, interprétation du terme image de $\xi'$ par $\gamma$, est un ensemble sans étoile, alors les solutions de $\gamma\mathcal{S}_{V^{(j)}}$ sont aussi des ensembles sans étoile.

Ces trois propriétés nous garantissent que les conditions **C1** et **C2** sont satisfaites pour le système $\mathcal{S}_\phi^{\text{num}}$, et donc que le système d'équations $\mathcal{S}_\phi^{\text{num}}$ peut effectivement être résolu.

### Partitionnement de $\mathsf{Vars}(\mathcal{S})$

Nous présentons ici un partitionnement de l'ensemble variables $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$ dont la construction est syntaxique à partir de $\mathcal{S}_\phi^{\text{num}}$. Ceci correspond à l'étape **(E1)** que nous avons annoncée ci-dessus.

Soit la relation binaire $\longrightarrow_{\mathcal{S}_\phi^{\text{num}}}$ entre variables du système $\mathcal{S}_\phi^{\text{num}}$ : pour $\xi, \xi'$ appartenant à $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$ et $f$ étant la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_\phi^{\text{num}}$, nous disons que $\xi$ *dépend de* $\xi'$ *dans* $\mathcal{S}_\phi^{num}$, noté $\xi \longrightarrow_{\mathcal{S}_\phi^{\text{num}}} \xi'$, si la variable $\xi'$ apparaît libre dans $f$. Nous notons par $\longrightarrow^+_{\mathcal{S}_\phi^{\text{num}}}$ la clôture transitive de la relation $\longrightarrow_{\mathcal{S}_\phi^{\text{num}}}$ et par $\longrightarrow^*_{\mathcal{S}_\phi^{\text{num}}}$ sa clôture transitive et réflexive.

La relation de dépendance $\longrightarrow_{\mathcal{S}_\phi^{\text{num}}}$ permet de définir des classes d'équivalence pour les variables : pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$ telle que $\xi \longrightarrow^+_{\mathcal{S}_\phi^{\text{num}}} \xi$, la *classe d'équivalence de* $\xi$ *pour* $\longrightarrow_{\mathcal{S}_\phi^{num}}$, notée $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$, est l'ensemble de variables $\xi'$ telles que $\xi \longrightarrow^+_{\mathcal{S}_\phi^{\text{num}}} \xi' \longrightarrow^*_{\mathcal{S}_\phi^{\text{num}}} \xi$.

Autrement dit, la classe d'équivalence $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$ contient exactement les variables $\xi'$ se trouvant sur un cycle pour la relation $\longrightarrow_{\mathcal{S}_\phi^{\text{num}}}$ passant par $\xi$. Notons que plusieurs cycles différents peuvent passer par $\xi$, $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$ est alors l'union des variables se trouvant sur n'importe quel de ces cycles. Donc, pour toutes variables $\xi$ et $\xi'$, on a $\xi$ appartient à $(\xi')_{\mathcal{S}_\phi^{\text{num}}}$ si et seulement si $\xi'$ appartient à $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$. Remarquons que la classe d'équivalence n'est pas définie pour les variables $\xi$ n'appartenant pas à un cycle de la relation $\longrightarrow_{\mathcal{S}_\phi^{\text{num}}}$. On dit dans ce cas que la variable $\xi$ *ne définit pas de classe d'équivalence dans* $\mathcal{S}_\phi^{num}$. Par exemple, si la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_\phi^{\text{num}}$ est une constante, alors la variable $\xi$ ne définit pas de classe d'équivalence. Si $\xi$ est une variable qui ne définit pas de classe d'équivalence, la notation $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$ est interdite.

Pour avoir une vue uniforme des classes d'équivalence et des variables ne définissant pas de classe d'équivalence, nous introduisons la notation suivante : $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ est l'ensemble

$$((\xi))_{\mathcal{S}_\phi^{\text{num}}} = \begin{cases} (\xi)_{\mathcal{S}_\phi^{\text{num}}} & \text{si la variable } \xi \text{ définit une classe d'équivalence dans } \mathcal{S}_\phi^{\text{num}}, \\ \{\xi\} & \text{sinon.} \end{cases}$$

L'introduction des ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ mérite une remarque : nous aurions pu obtenir les ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ en modifiant légèrement la définition de $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$. En effet, $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ est l'ensemble de variables $\xi'$ telles que $\xi \longrightarrow^*_{\mathcal{S}_\phi^{\text{num}}} \xi$. La différenciation entre les deux est nécessaire parce que dans la suite nous aurons besoin de distinguer les variables qui définissent une classe d'équivalence des variables qui ne définissent pas de classe d'équivalence. En particulier, un système d'équations dans lequel aucune variable ne définit de classe d'équivalence (càd aucune variable ne se trouve sur un cycle de la relation de dépendance) n'a pas forcément les mêmes propriétés qu'un système d'équations dans lequel certaines classes d'équivalence contiennent une seule variable.

La relation binaire $\rightsquigarrow_{\mathcal{S}_\phi^{\text{num}}}$ entre les ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ pour $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$ est alors définie par : $((\xi))_{\mathcal{S}_\phi^{\text{num}}} \rightsquigarrow_{\mathcal{S}_\phi^{\text{num}}} ((\xi'))_{\mathcal{S}_\phi^{\text{num}}}$ si $\xi \longrightarrow_{\mathcal{S}_\phi^{\text{num}}} \xi'$ et $\xi \notin ((\xi'))_{\mathcal{S}_\phi^{\text{num}}}$. Cette dernière condition implique que les ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ et $((\xi'))_{\mathcal{S}_\phi^{\text{num}}}$ sont disjoints.

**Lemme 7.8** *La relation* $\rightsquigarrow_{\mathcal{S}_\phi^{num}}$ *est acyclique.*

*Preuve* Ceci va de la définition même de $(\xi)_{\mathcal{S}_\phi^{\text{num}}}$ et des ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$, pour $\xi$ variable dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$.

□

Remarquons finalement que chacun des ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$ peut être construit syntaxiquement à partir de $\mathcal{S}_\phi^{\text{num}}$.

Dans la suite, nous considérons le partitionnement de $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$ dont les parties sont les ensembles $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$, pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$. Comme le système $\mathcal{S}_\phi^{\text{num}}$ est clair du contexte, nous écrirons $((\xi))$ à la place de $((\xi))_{\mathcal{S}_\phi^{\text{num}}}$.

**Représentation d'un ensemble semilinéaire sous la forme d'un terme clos**

Nous montrons ici que pour tout ensemble de multiensembles semilinéaires $D$, il existe un terme clos $f_D$ sur $fonct(\rho^*, \mathcal{X}_r)$ tel que $f_D^{\mathcal{R}} = D$. Ceci correspond à l'étape **(E2)** que nous avons énoncées ci-dessus.

L'idée est simple : tout ensemble de multiensembles semilinéaire est l'image de Parikh d'un langage rationnel de mots, tout langage rationnel de mots peut être représenté par une expression rationnelle, et toute expression rationnelle peut être vue comme un terme clos sur $fonct(\rho^*, \mathcal{X}_r)$.

Soit $A$ un alphabet fini. Pour tout mot $w$ dans $A*$, son *image de Parikh* est l'application qui au symbole $a$ dans $A$ associe le nombre d'occurrences de $a$ dans le mot $w$. C'est à dire, l'image de Parikh d'un mot dans $A*$ est un multiensemble sur $A$. L'image de Parikh est étendue aux ensembles de mots : si $L \subseteq A*$ est un ensemble de mots, l'image de Parikh est l'union des images de Parikh des mots constituant $L$.

Pour un alphabet $A$ donné, une expression rationnelle est définie récursivement par :
- $\varnothing$ est une expression rationnelle,
- pour toute lettre $a$ dans $A$, $a$ est une expression rationnelle,
- si $e_1, e_2$ sont des expressions rationnelles, $e_1 \mid e_2$ et $e_1 \vee e_2$ sont aussi des expressions rationnelles ;
- si $e$ est une expression rationnelle, $e*$ est aussi une expression rationnelle.

Dans cette définition, nous avons pris la liberté de représenter la concaténation (.) par le symbole $\mid$ et l'union (+) par le symbole $\cup$. Un *langage rationnel* est un ensemble $L \subseteq A*$ qui peut être défini par une expression rationnelle (dans laquelle $\mid$ est interprété comme la concaténation, $\vee$ est l'union et $*$ est l'opérateur étoile).

Maintenant, il est bien connu que l'image de Parikh d'un langage rationnel $L$ est un ensemble semilinéaire et que tout ensemble semilinéaire est l'image de Parikh d'un langage rationnel $L$. Considérons donc l'ensemble semilinéaire $D$, et soit $L$ un langage rationnel sur l'alphabet $Q_\phi$ tel que $D$ est l'image de Parikh de $L$. Considérons une expression rationnelle $e$ qui définit le langage $L$. Considérons le terme $f$ obtenu en remplaçant dans $e$ les occurrences de $\varnothing$ par $\perp$ et les occurrences d'une lettre $q$ par la constante $C_{\{1_q\}}$ ; $f$ est un terme clos sur $fonct(\rho^*, \mathcal{X}_r)$. Il n'est pas très difficile de voir que $f^{\mathcal{R}}$ est exactement $D$.

Donc, il suffit de prendre le terme $f$ pour $f_D$. Remarquons que le terme $f_D$ ainsi défini n'est pas unique ; nous utilisons la notation $f_D$ pour un (quelconque) terme clos satisfaisant la condition $f_D^{\mathcal{R}} = D$.

Notons finalement que dans le cas particulier où $D$ est un ensemble sans étoile, nous savons que $D$ est l'image de Parikh d'un langage rationnel $L$ sans étoile, et donc l'expression rationnelle définissant $L$ est équivalente à une expression rationnelle sans étoile, mais utilisant le complémentaire.

### 7.2.3 Type de solution et résolution des systèmes extraits

Nous montrons dans cette section que pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$, si $\mathcal{S}_{((\xi))}$ est le système extrait de $\mathcal{S}_\phi^{\text{num}}$ induit par $((\xi))$, si $V$ est l'ensemble des variables apparaissant en partie droite d'équation dans $\mathcal{S}_{((\xi))}$ mais n'appartenant pas à $((\xi))$ et que $\gamma$ est une substitution de domaine $V$ qui à tout $\xi' \in V$ associe un terme fonctionnel clos sur $\rho^*$, alors

- pour toute variable $\xi'$ dans $((\xi))$, l'ensemble $Sol_{\mathcal{R}}(\mathcal{S}_{((\xi))}, \xi')$ est semilinéaire et
- on peut effectivement calculer un terme clos $f$ tel que $f^{\mathcal{R}} = Sol_{\mathcal{R}}(\mathcal{S}_{((\xi))}, \xi')$.

De plus, dans le cas particulier où $\phi$ est une formule du fragment LSrd et pour toute variable $\xi' \in V$, l'ensemble $(\gamma(\xi'))^{\mathcal{R}}$ est sans étoile, alors les solutions de $\gamma\mathcal{S}_{V(j)}$ sont aussi des ensembles sans étoile. Ceci correspond à l'étape **(E3)** que nous avons énoncée plus haut.

Nous allons distinguer les deux cas, suivant que $\phi$ est une formule du fragment LSrd ou du fragment LSrpd. Et effet, les arguments qui nous permettront de conclure dans les deux cas ne sont pas le mêmes. Dans le premier cas, nous utiliserons le fait que $\gamma\mathcal{S}_{((\xi))}$ est forcément un système à une unique équation (voir le lemme 7.9 ci-dessous), et dans le second cas nous utilisons un rapprochement entre $\gamma\mathcal{S}_{((\xi))}$ et une grammaire algébrique.

### Le cas LSrd

Dans cette section nous supposons que $\phi$ est une formule du fragment LSrd et que $\gamma$ est une substitution qui à toute variable $\xi'$ dans $V$ associe un terme clos $f$ dont l'interprétation est un ensemble sans étoile. Nous allons montrer qu'alors pour toute variable dans $((\xi))$, la solution de $\gamma\mathcal{S}_{((\xi))}$ pour cette variable est un ensemble sans étoile.

Il s'avère que, lorsque $\phi$ est une formule du fragment LSrd, $\gamma\mathcal{S}_{((\xi))}$ est un système à une unique solution de la forme

$$\xi \stackrel{\kappa}{=} f$$

où $f$ est un terme clos.

Ceci découle du lemme suivant :

**Lemme 7.9** *Si $\phi$ est une formule du fragment LSrd, alors aucune des variables ne définit de classe d'équivalence.*

*Preuve* Voir la section A.3 de l'annexe A.

□

Par définition même de l'ensemble $((\xi))$ et du système $\mathcal{S}_{((\xi))}$, le lemme 7.9 implique naturellement que $\gamma\mathcal{S}_{\xi}$ est à une seule équation. Il n'est pas difficile non plus de voir que le terme en partie droite de cette équation est un terme clos. En effet, par définition, le fait que la variable $\xi$ ne définit pas de classe d'équivalence dans $\mathcal{S}_{\phi}^{\text{num}}$ implique que la variable $\xi$ ne dépend pas d'elle même. Autrement dit, la variable $\xi$ n'apparaît pas dans la partie droite de l'équation pour $\xi$ dans $\mathcal{S}_{\phi}^{\text{num}}$. Il nous reste à montrer que l'interprétation de ce terme par $\mathcal{R}$ est un ensemble sans étoile.

Rappelons d'abord qu'un ensemble de multiensembles est sans étoile s'il peut être décrit par une expression utilisant des ensembles finis, la somme d'ensembles de multiensembles (qui correspond à l'interprétation de l'opérateur | par $\mathcal{R}$), l'union et le complémentaire. Il nous suffit de montrer que $f^{\mathcal{R}}$ est équivalent à une telle expression. Nous allons le montrer en utilisant la façon dont $f$ a été construit. Soit $f'$ la partie droite de l'équation pour la variable $\xi$ dans le système d'équations $\mathcal{S}_{\phi}^{\text{num}}$. Alors, par définition, $f$ est obtenu à partir de $f'$ en remplaçant dans celui-ci chaque variable libre $\xi'$ par un terme clos, soit $f_{\xi'}$, tel que $f_{\xi'}^{\mathcal{R}}$ est un ensemble sans étoile. Par définition de $\mathcal{S}_{\phi}^{\text{num}}$, nous savons que $f'$ est de la forme $\xi' \vee \xi''$ ou $\xi' \wedge \xi''$ ou $\xi' \,|\, \xi''$ ou $\xi' \,\|\, \xi''$ ou $C_{\{1_q\}}$ pour $q$ dans $Q_{\phi}$. Donc, $f$ est de la forme $f_{\xi'} \vee f_{\xi''}$ ou $f_{\xi'} \wedge f_{\xi''}$ ou $f_{\xi'} \,|\, f_{\xi''}$ ou

$f_{\xi'} \parallel f_{\xi''}$ ou $C_{\{1_q\}}$. Dans chacun de ces cas, nous savons que $f_{\xi'}$ et $f_{\xi''}$ peuvent s'écrire sous la forme d'une expression utilisant des ensembles finis, la somme, l'union et le complémentaire, puisque pour tous ensembles de multiensembles $D, D'$, on a

$$(D \wedge D')^{\mathcal{R}} = \complement(\complement D \cup \complement D') \quad \text{et} \quad (D \parallel D')^{\mathcal{R}} = \complement(\complement D \mid \complement D').$$

**Le cas LSrpd**

Nous supposons ici que $\phi$ est une formule du fragment LSrpd. Nous allons montrer qu'alors pour toute variable dans $(\!(\xi)\!)$, la solution de $\gamma\mathcal{S}_{(\!(\xi)\!)}$ pour cette variable est un ensemble semi-linéaire. Nous utilisons un rapprochement entre le système extrait $\gamma\mathcal{S}_{(\!(\xi)\!)}$ est une grammaire algébrique.

Soit $\mathcal{S}$ un système d'équations (non à points fixes) sur $fonct(\rho^*, V)$. La plus petite solution de $\mathcal{S}$ est un tuple d'ensembles de taille $|\mathsf{Vars}(\mathcal{S})|$ ; pour chaque variable $\xi'$ dans $\mathsf{Vars}(\mathcal{S})$ nous notons par $Sol_{\mu,\mathcal{R}}(\mathcal{S}', \xi')$ la composante de la solution correspondant à cette variable. On peut voir que

**Fait 7.1** Chacun des ensembles $Sol_{\mu,\mathcal{R}}(\mathcal{S}', \xi')$, pour $\xi'$ dans $\mathsf{Vars}(\mathcal{S})$, est l'image de Parikh d'un langage algébrique. De plus, cette solution peut effectivement être calculée.

Sans rentrer dans les détails, ceci peut être facilement conclu de [Autebert et al., 1997] par exemple. L'idée est que le système d'équations $\mathcal{S}'$ peut être vu comme une grammaire algébrique sur le monoide commutatif libre.

Montrons maintenant comment on peut utiliser ce résultat pour résoudre le système d'équations $\gamma\mathcal{S}_{(\!(\xi)\!)}$. Pour cela, nous commençons par mettre en évidence une propriété du système d'équations $\mathcal{S}_\phi^{\text{num}}$ qui découle du fait que $\phi$ est une formule du fragment LSrpd.

**Lemme 7.10** *Si $\phi$ est une formule LSrpd, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{num})$, le système d'équations $\mathcal{S}_{(\!(\xi)\!)}$ extrait de $\mathcal{S}_\phi^{num}$ induit par $(\!(\xi)\!)$ satisfait une de ces deux conditions :*

*(i) toutes les équations dans $\mathcal{S}_{(\!(\xi)\!)}$ sont des équations à plus petit point fixe et toute partie droite d'équation dans $\mathcal{S}_{(\!(\xi)\!)}$ utilise uniquement les opérateurs $\vee$, $\mid$ et les opérateurs constants $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ et des constantes $C_{\{1_q\}}$ pour $q$ dans $Q_\phi$.*

*(ii) toutes les équations dans $\mathcal{S}_{(\!(\xi)\!)}$ sont des équations à plus grand point fixe et toute partie droite d'équation dans $\mathcal{S}_{(\!(\xi)\!)}$ utilise uniquement les opérateurs $\wedge$, $\parallel$ et les opérateurs constants $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ et des constantes $C_{\{1_q\}}$ pour $q$ dans $Q_\phi$.*

*Preuve* Voir la section A.3 de l'annexe A.

□

Suivant les deux possibilités prévues par le lemme 7.10. Si le système $\mathcal{S}_{(\!(\xi)\!)}$ satisfait la condition (i) de ce lemme, alors $\gamma\mathcal{S}_{(\!(\xi)\!)}$ est presque un système d'équations sur $fonct(\rho^*, \mathcal{X}_\mathrm{r})$, à ceci près que les constantes $\top$, $\mathbf{0}$ et $\overline{\mathbf{0}}$ peuvent apparaître en partie droite d'équation. Or, nous savons que l'interprétation de chacune de ces constantes est un ensemble semilinéaire, et donc il existe un terme clos sur $\rho^*$ dont l'interprétation coïncide avec l'interprétation de cette constante. Soit donc $\mathcal{S}'_{(\!(\xi)\!)}$ obtenu à partir de $\mathcal{S}_{(\!(\xi)\!)}$ en remplaçant dans $\mathcal{S}_{(\!(\xi)\!)}$ toute occurrence d'une des

constantes $\top$, $\mathbf{0}$ ou $\overline{\mathbf{0}}$ par un terme clos sur $\rho^*$ qui lui est équivalent. Alors $\mathcal{S}'_{(\!(\xi)\!)}$ est un système d'équations sur $fonct(\rho^*, \mathcal{X}_\mathrm{r})$ qui est équivalent à $\mathcal{S}_{(\!(\xi)\!)}$ sur toutes les variables, et de plus $\gamma\mathcal{S}'_{(\!(\xi)\!)}$ est équivalent à $\gamma\mathcal{S}_{(\!(\xi)\!)}$. De plus, toutes les équations de $\gamma\mathcal{S}'_{(\!(\xi)\!)}$ sont à plus petit point fixe. Alors la solution de $\gamma\mathcal{S}'_{(\!(\xi)\!)}$ est égale à la plus petite solution de $\gamma\mathcal{S}'_{(\!(\xi)\!)}$ considéré comme un système d'équations sans points fixes. Par le fait 7.1, nous concluons que la solution de $\gamma\mathcal{S}'_{(\!(\xi)\!)}$ peut effectivement être calculée et toutes ses composantes sont des ensembles semilinéaires.

Maintenant, considérons le cas où le système d'équations $\mathcal{S}_{(\!(\xi)\!)}$ satisfait la condition (ii) du lemme 7.10. Dans ce cas, le système $\gamma\mathcal{S}_{(\!(\xi)\!)}$ ne peut pas être transformé en un système d'équations sur $fonct(\rho^*, \mathcal{X}_\mathrm{r})$ aussi facilement que dans le cas précédent. Néanmoins, nous allons construire un système d'équations $\mathcal{S}''_{(\!(\xi)\!)}$ sur $fonct(\rho^*, \mathcal{X}_\mathrm{r})$ tel que la solution de $\gamma\mathcal{S}_{(\!(\xi)\!)}$ est le complémentaire de la solution de $\gamma\mathcal{S}''_{(\!(\xi)\!)}$ ; càd, pour toute variable $\xi'$ dans $\gamma\mathcal{S}_{(\!(\xi)\!)}$, il existe une variable $\xi''$ dans $\gamma\mathcal{S}''_{(\!(\xi)\!)}$ telle que $Sol_\mathcal{R}(\gamma\mathcal{S}_{(\!(\xi)\!)}, \xi') = \complement Sol_\mathcal{R}(\gamma\mathcal{S}''_{(\!(\xi)\!)}, \xi'')$. Intuitivement, $\mathcal{S}''_{(\!(\xi)\!)}$ est en quelque sorte le système dual de $\mathcal{S}_{(\!(\xi)\!)}$ : le système obtenu en remplaçant chaque opérateur par son dual et chaque constante par son complémentaire.

Soit donc le système d'équations $\mathcal{S}''_{(\!(\xi)\!)}$ obtenu à partir de $\mathcal{S}_{(\!(\xi)\!)}$ syntaxiquement de la manière suivante (le résultat de cette construction est présentée dans l'exemple 7.4) :

- si $\mathsf{Vars}(\mathcal{S}'_{(\!(\xi)\!)})$ est la séquence $\xi_1, \ldots, \xi_n$, alors $\mathsf{Vars}(\mathcal{S}''_{(\!(\xi)\!)})$ est la séquence $\xi'_1, \ldots, \xi'_n$ (càd les ensembles de variables des deux systèmes sont disjoints et il y a une bijection entre les deux) ;
- toute occurrence de la variable $\xi_i$ dans $\mathcal{S}'_{(\!(\xi)\!)}$ est remplacée par la variable $\xi'_i$, pour tout $i$ dans $1..n$ ;
- toute constante $C_{\{1_q\}}$ est remplacée par un terme clos $f$ sur $\rho^*$ tel que $f^\mathcal{R} = \complement\{1_q\}$. Comme $\{1_q\}$ est un ensemble semilinéaire, son complémentaire l'est aussi et donc le terme $f$ peut être effectivement calculé ;
- toute occurrence de la constante $\top$ (respectivement de la constante $\bot$, de la constante $\mathbf{0}$ et de la constante $\overline{\mathbf{0}}$) est remplacée par un terme clos $f$ sur $\rho^*$ tel que $f^\mathcal{R}$ est égal à $\bot^\mathcal{R}$ (respectivement à $\top^\mathcal{R}$, à $\overline{\mathbf{0}}^\mathcal{R}$ et à $\mathbf{0}^\mathcal{R}$). Comme chacune de ces constantes est un ensemble semilinéaire, le terme $f$ peut être effectivement calculé.
- toute occurrence de l'opérateur $\wedge$ est remplacée par l'opérateur $\vee$ ;
- toute occurrence de l'opérateur $\|$ est remplacée par l'opérateur $|$.

Soient maintenant $x_1, \ldots, x_k$ les variables libres de $\mathcal{S}_{(\!(\xi)\!)}$, et donc aussi les variables libres de $\mathcal{S}''_{(\!(\xi)\!)}$. Soit $\langle h_1(x_1, \ldots, x_k), \ldots, h_n(x_1, \ldots, x_k)\rangle$ la plus grande solution de $\mathcal{S}_{(\!(\xi)\!)}$ vu comme un système d'équations sans points fixes ; comme toutes les équations de $\mathcal{S}_{(\!(\xi)\!)}$ sont des équations à plus grand point fixe, c'est donc aussi la solution de $\mathcal{S}_{(\!(\xi)\!)}$ vu comme un système d'équations à points fixes. Soit $\langle g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k)\rangle$ la plus petite solution de $\mathcal{S}''_{(\!(\xi)\!)}$. Grâce à la dualité des opérateurs, il n'est pas très difficile de voir que pour tout $i$ dans $1..n$, la fonction $h_i(x_1, \ldots, x_k)$ est égale à la fonction $\complement g_i(\complement x_1, \ldots, \complement x_k)$. Soit maintenant $\gamma'$ la substitutions de domaine $(\!(\xi)\!)$ (càd de même domaine que $\gamma$) et qui à tout $\xi'$ dans $(\!(\xi)\!)$ associe un terme $f$ tel que $f^\mathcal{R} = \complement(\gamma(\xi'))^\mathcal{R}$. Nous pouvons déduire que pour toute variable $\xi'$ dans $(\!(\xi)\!)$, solution de $\gamma\mathcal{S}_{(\!(\xi)\!)}$ pour $\xi'$ est le complémentaire de la composante de la plus petite solution de $\gamma'\mathcal{S}''_{(\!(\xi)\!)}$ pour $\xi'_i$ :

$$\text{pour tout } \xi' \text{ dans } (\!(\xi)\!),\ Sol_\mathcal{R}(\gamma\mathcal{S}_{(\!(\xi)\!)}, \xi') = \complement Sol_{\mu,\mathcal{R}}(\gamma'\mathcal{S}''_{(\!(\xi)\!)}, \xi')$$

Or, par le fait 7.1, la plus petite solution de $\gamma'\mathcal{S}''_{(\!(\xi)\!)}$ peut être effectivement calculée sous la forme d'un terme clos sur $\rho^*$, et donc aussi son complémentaire.

Les différentes étapes de cette construction sont illustrées dans l'exemple qui suit.

**Exemple 7.4** Si $\mathcal{S}'$ est le système d'équations

$$\xi_1 \stackrel{\nu}{=} \overline{0}$$

alors $\mathcal{S}''$ est le système d'équations

$$\xi'_1 = \bot *$$

Dans ce cas, la solution de $\mathcal{S}'$ est $\langle \mathsf{C}\{\{\!|\}\!\}\rangle$, la plus petite solution de $\mathcal{S}''$ est $\{\{\!|\}\!\}$, et on a bien le fait que l'un est le complémentaire de l'autre dans $\mathbb{N}^{Q_\phi}$.

Si $\mathcal{S}'$ est le système d'équations

$$\xi_1 \stackrel{\nu}{=} \xi_2 \,||\, \xi_3, \quad \xi_2 \stackrel{\nu}{=} \xi_3 \wedge x_1, \quad \xi_3 \stackrel{\nu}{=} \xi_4 \,||\, x_2, \quad \xi_4 \stackrel{\nu}{=} \xi_1 \wedge x_2$$

alors $\mathcal{S}''$ est le système d'équations

$$\xi'_1 = \xi'_2 \,|\, \xi'_3, \quad \xi'_2 = \xi'_3 \vee x_1, \quad \xi'_3 = \xi'_4 \,|\, x_2, \quad \xi'_4 = \xi'_1 \vee x_2$$

Soit

$$\langle h_1(x_1, x_2), h_2(x_1, x_2), h_3(x_1, x_2), h_4(x_1, x_2)\rangle$$

la solution de $\mathcal{S}'$ et soit

$$\langle g_1(x_1, x_2), g_2(x_1, x_2), g_3(x_1, x_2), g_4(x_1, x_2)\rangle$$

la plus petite solution de $\mathcal{S}''$, où les $h_i(x_1, x_2)$ et les $g_i(x_1, x_2)$ sont des fonctions à deux variables sur les sous-ensembles de $\mathbb{N}^{Q_\phi}$.

Maintenant, si le système d'équations $\mathcal{S}_\xi$ est obtenu à partir de $\mathcal{S}'$ en remplaçant les occurrences de la variable $x_1$ (respectivement de la variable $x_2$) par le terme clos sur $f_1$ sur $\rho^*$ (respectivement le terme clos $f_2$ sur $\rho^*$) tel que $f_1^{\mathcal{R}} = D_1$ (respectivement $f_2^{\mathcal{R}} = D_2$), alors nous avons

$$h_i(D_1, D_2) = \mathsf{C}g_i(\mathsf{C}D_1, \mathsf{C}D_2)$$

pour $i$ étant 1, 2, 3 ou 4 et

$$\langle h_1(D_1, D_2), h_2(D_1, D_2), h_3(D_1, D_2), h_4(D_1, D_2)\rangle$$

est la solution de $\mathcal{S}_\xi$.

Finalement, si $\mathcal{S}''_\xi$ est le système d'équations

$$\xi'_1 = \xi'_2 \,|\, \xi'_3, \quad \xi'_2 = \xi'_3 \vee f'_1, \quad \xi'_3 = \xi'_4 \,|\, x_2, \quad \xi'_4 = \xi'_1 \vee f'_2$$

tel que $f'_1$ et $f'_2$ sont des termes clos sur $\rho^*$ avec $f_1'^{\mathcal{R}} = \mathsf{C}D_1$ et $f_2'^{\mathcal{R}} = \mathsf{C}D_2$, alors la plus petite solution de $\mathcal{S}''_\xi$ est

$$\langle \mathsf{C}\,h_1(D_1, D_2), \mathsf{C}\,h_2(D_1, D_2), \mathsf{C}\,h_3(D_1, D_2), \mathsf{C}\,h_4(D_1, D_2)\rangle$$

et nous pouvons donc en déduire la solution de $\mathcal{S}_\xi$.

Nous avons ainsi complété la preuve du lemme 7.6 qui établissait que

- Si $\phi$ est une formule de la logique LSrd, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$, l'ensemble $Sol_{\mathcal{R}}(\mathcal{S}_\phi^{\text{num}}, \xi)$ est un ensemble de multiensembles sans étoile qui peut être effectivement calculé.
- Si $\phi$ est une formule de la logique LSrpd, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}_\phi^{\text{num}})$, l'ensemble $Sol_{\mathcal{R}}(\mathcal{S}_\phi^{\text{num}}, \xi)$ est un ensemble de multiensembles semilinéaire qui peut être effectivement calculé.

Comme nous l'avons vu en début de cette section, ceci nous permet de conclure que l'ACN correspondant à une formule LSrd est un ACN sans étoile, et l'ACN correspondant à une formule LSrpd est un ACN semilinéaire.

**Proposition 7.11** – *Si $\phi$ est une formule LSrd close, alors on peut construire l'automate à contraintes numériques sans étoile $A_\phi$ effectif tel que le langage de $A_\phi$ est l'ensemble $[\![\phi]\!]$.*
– *Si $\phi$ est une formule LSrpd close, alors on peut construire l'automate à contraintes numériques semilinéaire $A_\phi$ effectif tel que le langage de $A_\phi$ est l'ensemble $[\![\phi]\!]$.*

## 7.3 Encodage des automates vers des formules

Nous montrons dans cette section que les automates à contraintes numériques sans étoile peuvent être encodés dans la logique LSrd et les automates à contraintes numériques semilinéaires peuvent être encodés dans la logique LSrpd.

Soit $\tau''$ la signature $\{\mathbf{0}, \top, |, \vee, *\} \cup \{\alpha\}_{\alpha \subseteq \Sigma \text{ fini ou cofini}}$, et soit la $\mu$-interprétation $\mathcal{T}$ de $\tau''$ sur le treillis $\langle \wp(\mathcal{A}_\Sigma), \vee, \wedge \rangle$ où tous les symboles sont interprétés comme d'habitude et l'interprétation du symbole étoile est :

$$(S*)^{\mathcal{T}} = \bigcup_{n \in \mathbb{N}} \underbrace{S|^{\mathcal{T}} \cdots |^{\mathcal{T}} S}_{n \text{ fois}}$$

Comme nous l'avons déjà remarqué pour l'opérateur $*$ dans la logique, pour tout terme $f$ dans $fonct(\tau'', \mathcal{X}_{\text{r}})$, nous avons $f^{\mathcal{T}} = (\mu\xi.f \mid \xi \vee \mathbf{0})^{\mathcal{T}}$. Notons que tout système d'équations sur $fonct(\tau'', \mathcal{X}_{\text{r}})$ admet une solution lorsque les parties droites d'équations sont interprétées par $\mathcal{T}$. Si $\mathcal{S}$ est un système d'équations sur $fonct(\tau'', \mathcal{X}_{\text{r}})$, on peut construire la formule $\mathsf{formule}(\mathcal{S})$ de la même façon que pour les systèmes d'équations sur $fonct(\tau, \mathcal{X}_{\text{r}})$, avec $[\![\mathsf{formule}(\mathcal{S})]\!] = Sol_{\mathcal{T}}(\mathcal{S}, \mathsf{dern}(\mathcal{S}))$. Dans ce cas $\mathsf{formule}(\mathcal{S})$ est une formule sur la syntaxe d'origine sans négation et utilisant l'opérateur étoile. La notion de chemin peut également être étendue pour prendre en compte le symbole $*$. Remarquons finalement que l'opérateur étoile est autorisé dans les formules à récursion positive / descendante, mais pas dans les formules à récursion descendante. Plus précisément,

**Fait 7.2** Tout terme clos sur $fonct(\tau'', \mathcal{X}_{\text{r}})$ peut être transformé en une formule LSrpd équivalente.

Soit $A = (\Sigma, Q, \Delta, F)$ un automate à contraintes numériques semilinéaires ou sans étoile effectif, et soit $Q = \{q_1, \ldots, q_n\}$. Nous commençons par construire un système d'équations $\mathcal{S}_A$ sur $fonct(\tau'', \mathcal{X}_{\text{r}})$ tel que $[\![\mathsf{formule}(\mathcal{S}_A)]\!] = \mathcal{L}(A)$.

Soit $\mathcal{D}_A$ l'ensemble des multiensembles $\mathbb{N}^Q$ qui apparaissent dans la relation de transition de l'automate ou bien en tant que condition d'acceptation :

$$\mathcal{D}_A = \{D \subseteq \mathbb{N}^Q \mid \text{ ils existent } \alpha, q \text{ tels que } (q, \alpha, D) \in \Delta\} \cup \{F\}.$$

Soit $\mathcal{X}_Q$ l'ensemble de variables $\{\xi_q \mid q \in Q\}$ et $\mathcal{X}_{\mathcal{D}_A}$ l'ensemble de variables $\{\xi_D \mid D \in \mathcal{D}_A\}$. L'ensemble des variables de $\mathcal{S}_A$ est $\mathcal{X}_Q \cup \mathcal{X}_{\mathcal{D}_A}$. L'ordre des ces variables dans $\mathsf{Vars}(\mathcal{S}_A)$ est n'importe quel ordre tel que les variables de $\mathcal{X}_Q$ précèdent les variables de $\mathcal{X}_{\mathcal{D}_A}$ et $\xi_F$ est la dernière variable dans $\mathsf{Vars}(\mathcal{S}_A)$. Pour tout état de l'automate $q$, l'équation pour $\xi_q$ dans $\mathcal{S}_A$ est

$$\xi_q \stackrel{\mu}{=} \bigvee_{\alpha, D \mid (q,\alpha,D)\in\Delta} \alpha[\xi_D].$$

Pour tout ensemble $D$ dans $\mathcal{D}_A$ l'équation pour $\xi_D$ dans $\mathcal{S}_A$ est

$$\xi_D \stackrel{\mu}{=} \mathsf{form}(D)$$

où la formule $\mathsf{form}(D)$ est une formule de la logique dont la définition est donnée ci-dessous. Comme par hypothèse l'automate $A$ est effectif, l'ensemble $D$ est représenté comme une union d'ensembles linéaires. Soit $D = L_1 \cup \cdots \cup L_k$, où chacun des $L_i$ est un ensemble linéaire, alors

$$\mathsf{form}(D) = \bigvee_{i\in 1..k} \mathsf{form}(L).$$

Maintenant, pour $L$ étant l'ensemble linéaire $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$, la formule $\mathsf{form}(L)$ est

$$\mathsf{form}(L) = \mathsf{form}(\mathbf{b}) \,|\, \mathsf{form}{*}(\mathbf{p}_1) \,|\, \cdots \,|\, \mathsf{form}{*}(\mathbf{p}_k).$$

Pour tout multiensemble $\mathbf{d}$, les formules $\mathsf{form}(\mathbf{d})$ et $\mathsf{form}{*}(\mathbf{d})$ sont définies par

$$\begin{aligned} \mathsf{form}(\mathbf{d}) &= \mathbf{d}(q_1) \cdot \xi_{q_1} \,|\, \cdots \,|\, \mathbf{d}(q_n) \cdot \xi_{q_n} \\ \mathsf{form}{*}(\mathbf{d}) &= \mathsf{form}(\mathbf{d}) * . \end{aligned}$$

où pour tout entier naturel $m$ et toute formule $\phi$, $m \cdot \phi$ est un raccourci pour la formule $\underbrace{\phi \,|\, \cdots \,|\, \phi}_{m \text{ fois}}$.

Avec cette notation, pour toute formule $\phi$, $0.\phi$ est la formule $\mathbf{0}$.

Un premier résultat que nous montrons est que le système d'équations ainsi défini est équivalent à l'automate $A$ dans le sens suivant :

**Lemme 7.12** *Pour tout automate à contraintes semilinéaires (et donc pour tout automate à contraintes sans étoile) $A$, le système d'équations $\mathcal{S}_A$ vérifie*

$$\mathcal{L}(A) = Sol_{\mathcal{T}}(\mathcal{S}_A, \xi_F).$$

*Preuve* Ce lemme est conséquence de la définition du langage d'un automate. Soit $\mathcal{S}'$ le système d'équations obtenu à partir de $\mathcal{S}_A$ en enlevant les opérateurs de point fixe, càd $\{\mathsf{Vars}(\mathcal{S}')\} = \{\mathsf{Vars}(\mathcal{S}_A)\}$ et pour toute équation $\xi \stackrel{\mu}{=} f$ dans $\mathcal{S}_A$, l'équation correspondant à la variable $\xi$ dans $\mathcal{S}'$ est $\xi = f$. ($\mathcal{S}'$ n'est pas un système d'équations à point fixe ; pour la différence entre les systèmes d'équations et les systèmes d'équations à point fixe, voir la section 4.1, page 68). Nous savons que la solution de $\mathcal{S}_A$ est égale à la plus petite solution de $\mathcal{S}'$. Pour toute variable $\xi$ dans $\{\mathsf{Vars}(\mathcal{S}')\}$, nous notons par $Sol_{\mu,\mathcal{T}}(\mathcal{S}', \xi)$ la composante de la plus petite solution de $\mathcal{S}'$ correspondant à la variable $\xi$.

Nous allons montrer les deux propriétés suivantes :

(i) si pour tout ensemble $D$ dans $\mathcal{D}$ on a $Sol_{\mu,\mathcal{T}}(\mathcal{S}', \xi_D) = \mathcal{L}_A(D)$, alors pour tout état de l'automate $q$ on a $Sol_{\mu,\mathcal{T}}(\mathcal{S}', \xi_q) = \mathcal{L}_A(q)$ ;

(ii) si pour tout état de l'automate $q$ on a $Sol_{\mu,\mathcal{T}}(\mathcal{S}', \xi_q) = \mathcal{L}_A(q)$, alors pour tout ensemble $D$ dans $\mathcal{D}$ on a $Sol_{\mu,\mathcal{T}}(\mathcal{S}', \xi_D) = \mathcal{L}_A(D)$ ;

D'après les définitions de $\mathcal{L}_A(q)$, $\mathcal{L}_A(D)$ et $\mathcal{L}(A)$, le lemme découle facilement de (i) et (ii).

Le point (i) est conséquence immédiate des définitions de $\mathcal{L}_A(q)$, $\mathcal{L}_A(D)$ et $\mathcal{L}(A)$, et le point (ii) est également facile à montrer à partir de ces définitions et de l'interprétation des opérateurs logiques.

□

Comme formule($\mathcal{S}_A$) est un terme dans $fonct(\tau'', \mathcal{X}_r)$ et $[\![\text{formule}(\mathcal{S}_A)]\!] = Sol_{\mathcal{T}}(\mathcal{S}_A, \xi_F)$, par le fait 7.2 et le lemme précédent il suit immédiatement que

**Lemme 7.13** *Pour tout automate à contraintes numériques semilinéaires $A$, il existe une formule close $\phi$ dans le fragment LSrpd telle que $\mathcal{L}(A) = [\![\phi]\!]$.*

Ce résultat peut même être renforcé. Comme nous l'avons remarqué, formule($\mathcal{S}_A$) est dans une formule sur la syntaxe d'origine sans négation et utilisant l'opérateur étoile. Or, toute sous formule $\phi*$ de formule($\mathcal{S}_A$) peut être remplacée par la formule équivalente $\mu\xi.\phi \mid \xi \vee \mathbf{0}$, où $\xi$ est une variable n'apparaissant pas dans formule($\mathcal{S}_A$). Nous en déduisons que formule($\mathcal{S}_A$) est équivalente à une formule sur la syntaxe d'origine sans négation, d'où le suivant

**Lemme 7.14** *Pour tout automate à contraintes numériques semilinéaires $A$, il existe une formule close $\phi$ sur la syntaxe d'origine et sans négation telle que $\mathcal{L}(A) = [\![\phi]\!]$.*

Le lemme 7.13 établit que les automates à contraintes numériques semilinéaires peuvent être encodés dans le fragment LSrpd de la logique. Il nous reste à montrer le résultat similaire pour les automates à contraintes numériques sans étoile et le fragment LSrd. La preuve dans ce cas est moins immédiate. Nous allons d'abord montrer que la formule formule($()\mathcal{S}_A$) est une formule gardée utilisant l'opérateur étoile, et nous allons ensuite montrer que l'opérateur étoile peut être éliminé de cette formule.

Dans ce but, nous introduisons un troisième fragment de la logique spatiale, qui est un fragment de LSrpd et dont LSrd est un fragment.

**Définition 7.15 (Fragment à récursion descendante avec étoile)** Une formule $\phi$ est dite à *récursion descendante avec étoile* si $\phi$ est une formule sur la syntaxe d'origine enrichie avec l'opérateur étoile $*$ et telle que pour toute sous formule $\mu\xi.\phi'$ de $\phi$ et pour tout chemin $c.\xi$ dans $\phi'$, le chemin $c$ contient au moins une occurrence d'opérateur d'emboîtement $\alpha$.

Le *fragment à récursion descendante avec étoile* de la logique est constitué des formules à récursion descendante avec étoile et est noté LSrd*

Nous commençons par transformer le système d'équations $\mathcal{S}_A$ en un système $\mathcal{S}''_A$ équivalent tel que la formule formule($\mathcal{S}''_A$) est une formule à récursion descendante avec étoile. Nous montrons ensuite que formule($\mathcal{S}''_A$) l'étoile peut être éliminée de cette formule pour obtenir une formule à récursion descendante, ce qui signifie que formule($\mathcal{S}''_A$) est une formule dans le fragment LSrd.

Considérons le système d'équations $\mathcal{S}'_A$ défini à partir de $\mathcal{S}_A$ comme suit. Pour tout $q$ état de $A$ et pour tout $D$ dans $\mathcal{D}_A$, soient $f_q$ et $f_D$ les parties droites des équations pour les variables $\xi_q$ et $\xi_D$ dans $\mathcal{S}_A$. Soit également $\Gamma_{\mathcal{S}_A,\mathcal{X}_Q}$ la substitution de domaine $\mathcal{X}_Q$ qui à $\xi_q$ associe le terme $f_q$, ceci pour tout $q$ état de $A$. Remarquons que l'ordre dans lequel les variables apparaissent dans $\Gamma$ est sans importance puisque aucun des $f_q$ ne contient de variable de $\mathcal{X}_Q$ libres. Alors, pour la définition de $\mathcal{S}'_A$ :

- $\mathsf{Vars}(\mathcal{S}'_A) = \mathsf{Vars}(\mathcal{S}_A)$ ;
- pour tout $q$ état de $A$, l'équation pour la variable $\xi_q$ dans $\mathcal{S}'_A$est la même que l'équation pour la variable $\xi_q$ dans $\mathcal{S}_A$ ;
- pour tout $D$ dans $\mathcal{D}_A$, la partie droite de l'équation pour $\xi_D$ dans $\mathcal{S}'_A$ est $\Gamma_{\mathcal{S}_A,\mathcal{X}_Q}(f_D)$.

Par la proposition 4.5, il est facile de voir que le système $\mathcal{S}'_A$ est équivalent au système $\mathcal{S}_A$ sur l'ensemble de variables $\mathsf{Vars}(\mathcal{S}_A)$. Donc, par le lemme 7.12, $\mathcal{L}(A) = Sol_{\mathcal{T}}(\mathcal{S}'_A, \xi_F)$. Soit $\mathcal{S}''_A$ le système extrait de $\mathcal{S}'_A$ induit par $\mathcal{X}_{\mathcal{D}_A}$. Remarquons que $\mathcal{S}''_A$ est un système clos : càd toute variable apparaissant en partie droite d'équation dans $\mathcal{S}''_A$ est une variable de $\mathcal{X}_{\mathcal{D}_A}$. Donc, par la proposition 4.4, nous avons que $\mathcal{L}(A) = Sol_{\mathcal{T}}(\mathcal{S}''_A, \xi_F)$. Maintenant, par définition, $\xi_F$ est la dernière variable de $\mathsf{Vars}(\mathcal{S}''_A)$, et donc $\mathcal{L}(A) = [\![\mathsf{formule}(\mathcal{S}''_A)]\!]$.

Montrons maintenant que $\mathsf{formule}(\mathcal{S}''_A)$ est une formule à récursion descendante avec étoile.

**Lemme 7.16** *Si $\mu\xi.\phi$ est sous formule de* $\mathsf{formule}(\mathcal{S}''_A)$ *et $c.\xi$ est un chemin dans $\phi$, alors $c$ contient au moins un opérateur $\alpha$ (pour $\alpha \subseteq \Sigma$).*

*Preuve* Voir la section A.3 de l'annexe A.

□

Notons que ce résultat est valide pour $A$ étant un automate à contraintes numériques semilinéaires. Nous pouvons donc en conclure encore une caractérisation du lien qu'il existe entre les automates à contraintes semilinéaires et les formules de la logique :

**Lemme 7.17** *Pour tout automate à contraintes numériques semilinéaires $A$, il existe une formule close $\phi$ dans LSrd* et sans négation telle que $\mathcal{L}(A) = [\![\phi]\!]$.*

Dans la suite, nous considérons le cas où $A$ est un automate à contraintes numériques sans étoile. Le lemme suivant établit un principe de préservation des sous apparaissant en partie droite de $\mathcal{S}''_A$ et les sous formules de $\mathsf{formule}(\mathcal{S}''_A)$.

**Lemme 7.18** *Si $\phi$ est sous formule de* $\mathsf{formule}(\mathcal{S}''_A)$*, alors $\phi$ est de la forme $\Gamma(\phi')$ où $\phi'$ est une sous formule d'une partie droite d'équation dans $\mathcal{S}''_A$ et $\Gamma$ est une substitution de domaine inclus dans $\mathcal{X}_{\mathcal{D}_A}$ et qui à tout $\xi_D$ dans $dom(\Gamma)$ associe une formule de la forme $\mu\xi_D.f$.*

Nous omettons la preuve de ce lemme, qui est conséquence facile de la définition de $\mathsf{formule}(\mathcal{S}''_A)$ (section 4.2.3, page 73).

Nous pouvons maintenant annoncer le résultat concernant le lien entre les automates à contraintes numériques sans étoile et le fragment LSrd.

**Lemme 7.19** *Si $A$ est un automates à contraintes numériques sans étoile, alors* $\mathsf{formule}(\mathcal{S}''_A)$ *est équivalente à une formule LSrd.*

*Preuve* D'après le lemme 7.16, nous savons que formule($\mathcal{S}''_A$) est à récursion descendante à étoile. Nous allons montrer que l'étoile peut être éliminée et que la formule résultante est dans LSrd. Cela veut dire que l'élimination de l'étoile ne peut pas se faire en utilisant l'équivalence $\phi* = \mu\xi.\phi \mid \xi \vee \mathbf{0}$.

Commençons par montrer que les sous formules à étoile de formule($\mathcal{S}''_A$) ont une forme particulière. Soit $\phi*$ une sous formule de formule($\mathcal{S}''_A$). Alors, par le lemme 7.18, nous savons que $\phi*$ est de la forme $\Gamma(\phi_1)$, où $\phi_1$ est sous formule d'une partie droite d'équation dans formule($\mathcal{S}''_A$) et $\Gamma$ est une substitution de domaine inclus dans $\mathcal{X}_{\mathcal{D}_A}$ et qui à tout $\xi_D$ dans $dom(\Gamma)$ associe une formule de la forme $\mu\xi_D.f$. De part la forme de la substitution $\Gamma$, il est facile de voir que $\phi_1$ est de la forme $\phi_2*$ et $\phi* = \Gamma(\phi_2)*$. Or, par définition du système d'équations $\mathcal{S}''_A$, on peut voir que dans ce cas la formule $\phi_2$ est de la forme

$$\mathbf{d}(q_1) \cdot \psi_1 \mid \cdots \mid \mathbf{d}(q_n) \cdot \psi_k \tag{7.2}$$

où pour chaque $i$ dans $1..k$, $\psi_i$ est de la forme

$$\bigvee_{\alpha,D \mid (q,\alpha,D) \in \Delta} \alpha[\xi_D]$$

pour un certain état $q$ de l'automate $A$ et où $\mathbf{d}$ est un multiensemble dans $\mathbb{N}^Q$ où $\mathbf{d}$ est une période d'un ensemble linéaire $L$ tel que $L$ participe dans la définition d'un des ensemble $D'$ apparaissant dans une des transitions de l'automate $A$. Or, comme $A$ est un ACN sans étoile, nous savons que $\mathbf{d}$ est de la forme $1_q$ où $q$ est un état dans $Q$. C'est à dire, dans la formule (7.2) ci-dessus, un seul des $\mathbf{d}(q_i)$ est égal à un et tous les autres sont égaux à zéro. Nous en déduisons que $\phi_2$ est équivalente à une formule qui est une disjonction finie de formules de la forme $\alpha[\xi_D]$ ; soit $\phi_2$ équivalente à la formule

$$\alpha_1[\xi_{D_1}] \vee \cdots \vee \alpha_k[\xi_{D_k}].$$

Montrons que dans ce cas, $\phi_2*$ est équivalente à une formule sans étoile.

**Fait 7.3** Pour tous ensembles d'étiquettes $\alpha_1, \ldots, \alpha_n$ et pour toutes formules $\phi_1, \ldots, \phi_n$, la formule $(\alpha_1[\phi_1] \vee \cdots \vee \alpha_n[\phi_n])*$ est équivalente à la formule $\alpha_1[\phi_1]* \mid \cdots \mid \alpha_n[\phi_n]*$.

**Fait 7.4** Pour tout ensemble d'étiquettes $\alpha$ et toute formule $\phi$, la formule $\alpha[\phi]*$ est équivalente à la formule $\neg(\overline{\alpha}[\top] \mid \top) \wedge \neg(\alpha[\neg\phi] \mid \top)$.

Les preuves de ces deux faits sont conséquence facile de la définition d'interprétation des formules.

Maintenant, rappelant que $\phi = \Gamma(\phi_2)*$, montrons que l'étoile peut être éliminée de la formule $\phi$. Par les faits 7.3 et 7.4, il est évident que l'étoile peut être éliminée de la formule $\phi_2$. Alors il suffirait de montrer que l'étoile peut être éliminée de la formule $\Gamma(\xi)$ pour tout $\xi$ dans le domaine de $\Gamma$. Encore une fois, nous pouvons appliquer les fait 7.3 et 7.4, puisque nous savons que $\Gamma(\xi)$ est de la forme $\mu\xi.\phi'$ et $\phi'$ est la partie droite d'une équation dans $\mathcal{S}''_A$. Nous en déduisons que toute sous formule $\phi*$ de formule($\mathcal{S}''_A$) est équivalente à une formule $\phi'$ sans étoile, et donc, par récurrence sur la structure de formule($\mathcal{S}''_A$), on obtient que formule($\mathcal{S}''_A$) est équivalente à une formule $\Phi$ sans étoile. Il nous reste à montrer que $\Phi$ est à récursion gardée. Rappelant que formule($\mathcal{S}''_A$) est à récursion gardée avec étoile, c'est facile à voir d'après la façon

dont l'élimination de l'étoile a été définie : une élimination d'étoile consistait à remplacer une formule de la forme $\alpha[\psi]$ par la formule $\neg(\overline{\alpha}[\top] \mid \top) \wedge \neg(\alpha[\neg\psi] \mid \top)$, et il est évident que cette transformation n'introduit pas des récursions non gardées.

□

Comme conséquence immédiate de ce lemme, nous avons

**Corollaire 7.20** *Pour tout automate à contraintes numériques sans étoile $A$, il existe une formule $\phi$ close de la logique LSrd telle que $\mathcal{L}(A) = [\![\phi]\!]$.*

Des résultats de ce chapitre et cette section en particulier nous déduisons les équivalences suivantes entre fragments de la logique spatiale et automates à contraintes numériques :

**Théorème 7.21** *La classe de langages définissables par un automates à contraintes numériques semilinéaires est exactement la classe de langages définissables par la logique LSrpd.*

**Théorème 7.22** *La classe de langages définissables par un automates à contraintes numériques sans étoile est exactement la classe de langages définissables par la logique LSrd.*

De ces résultats et de la décidabilité du test du vide des automates à contraintes numériques sans étoile et semilinéaires (voir le lemme 2.30) nous déduisons que

**Théorème 7.23** *Les fragments de la logique LSrpd et LSrd ont un problème de satisfiabilité décidable.*

Finalement, des lemmes 7.14 et 7.17 et du théorème 7.21 nous déduisons

**Proposition 7.24** *Ces trois fragments syntaxiques de la logique spatiale sont équivalents :*
- *LSrpd,*
- *le fragment constitué des formules sans négation,*
- *le fragment constitué des formules sans négation, à récursion gardée et avec étoile.*

## 7.4 Équivalences des fragments de la logique spatiale avec MSO et PMSO

Nous terminons ce chapitre en montrant que la logique LSrpd est équivalente à la logique PMSO et que la logique LSrd est équivalente à MSO. Ces résultats sont conséquence facile des résultats établis dans ce chapitre, du théorème 3.1 établissant que les ACN sans étoile sont équivalents à la logique MSO et du théorème 3.2 établissant que les ACN semilinéaires sont équivalents à la logique PMSO.

**Théorème 7.25** *La logique LSrpd est équivalente à la logique PMSO.*

**Théorème 7.26** *La logique LSrd est équivalente à la logique MSO.*

De plus, le passage d'une formule MSO à une formule LSrd et inversement est effectif ; de même pour le passage d'une formule PMSO et une formule LSrpd et inversement. Cependant, cette transformation demande d'utiliser les automates à contraintes numériques et n'est pas du tout naturelle, en plus de provoquer une explosion de la taille dans la formule dans les deux sens. Nous ne voyons pas à ce jour de manière directe de transformer une formule MSO en une formule LSrd ou inversement.

## Conclusion

Nous avons établi dans ce chapitre plusieurs résultats importants sur l'expressivité de la logique spatiale. Tout d'abord, nous avons identifié deux fragments décidables de la logique : les fragments LSrd et LSrpd. Ce sont les premiers résultats de décidabilité de la logique avec récursion. Il est important à noter que ces deux fragments sont définis comme des restrictions syntaxiques de la logique, ce qui rend facile à identifier si une formule appartient ou non à un de ces fragments.

Un autre résultat important est l'équivalence entre un fragment de la logique spatiale (LSrd) et la logique MSO. Comme nous l'avons déjà souligné, la logique MSO est très populaire pour plusieurs raisons, et sert souvent de référence. Même si la correspondance entre les deux logiques n'est pas très intuitive (passage par les ACN) et de ce fait ne donne pas forcément une meilleure compréhension du mécanisme de récursion dans la logique spatiale, ce résultat est important en soi. D'autant plus que, d'un point de vue syntaxique, la logique spatiale et la logique MSO sont très différentes et cette correspondance aurait été plus difficile à établir sans les automates.

L'autre résultat d'équivalence que nous présentons ici est l'équivalence entre les logiques LSrpd et PMSO. Ce résultat est certainement moins surprenant, notamment grâce aux travaux [Dal Zilio and Lugiez, 2003] [Dal Zilio et al., 2004] qui avaient déjà établi un lien entre les formules de la logique spatiale et des automates semblables aux automates de Presburger. Mais nous établissons ici une équivalence stricte. Deux résultats plus surprenants sont les résultats d'équivalence entre LSrpd avec des fragments syntaxiques encore plus restrictifs pour la récursion. Notamment, LSrpd n'est pas plus expressive que le fragment de la logique sans négation, ni le fragment sans négation, à récursion gardée et à étoile. Ce constat renforce l'intuition que nous avons donnée dans le chapitre 5 sur la source d'indécidabilité de la logique spatiale : la possibilité de combiner des conjonction et des disjonctions dans les formules récursives.

# Troisième partie

# Model checking de la logique spatiale

Rappelons que le problème de model checking est, « étant donné un arbre $t$ et une formule $\phi$, décider si $t$ satisfait $\phi$ ». Savoir répondre à ce problème et connaître sa complexité est important pour plusieurs raisons. D'une part, le model checking peut être utilisé pour l'évaluation de requêtes. Considérons un arbre $t$ et la formule $\phi$ à $k$ variables libres $x_1, \ldots, x_k$. Cette formule peut être considérée comme une requête dont le but est d'extraire $k$ valeurs d'étiquettes de $t$ : le tuple d'étiquettes $a_1, \ldots, a_k$ est extrait si l'arbre $t$ satisfait la formule $\phi$ sous la valuation $\rho$ qui associe $a_i$ à la variable $x_i$. Si la formule $\phi$ est close, on parle d'une requête booléenne : le problème de model checking « $t$ satisfait il $\phi$ » permet de vérifier si l'arbre $t$ satisfait une propriété qui est définie par la formule $\phi$. Ces propriétés peuvent être de différentes natures, comme par exemple existence d'une clé primaire dans une base de données (représentée comme un arbre), un document a-t-il un type (la formule $\phi$ définit un type dont les valeurs sont les arbres satisfaisant cette formule), etc. D'autre part, la décidabilité du problème de model checking et sa complexité sont utilisés comme des « mesures » du pouvoir d'expression d'une logique et permettent de comparer l'expressivité de différentes logiques.

Dans cette partie de la thèse nous étudions le problème de model checking de la logique spatiale, et plus précisément sa décidabilité et sa complexité pour différents fragments de la logique.

# Chapitre 8

# Un algorithme de model checking pour la logique spatiale

La décidabilité du problème de model checking pour la logique spatiale n'est pas triviale. Il est évident que, pour une formule (close) $\phi$ donnée, il est impossible de calculer (càd représenter de manière finie en mémoire) l'ensemble $[\![\phi]\!]$ pour deux raisons : le point fixe ne converge pas nécessairement en un nombre fini d'itérations et la quantification est interprétée comme une union infinie. C'est pourquoi nous adoptons ici une approche de model checking local dans l'esprit des propriétés de la satisfiabilité des formules données dans le lemme 3.3. Dans cette optique, la première difficulté que nous avons citée est résolue par la possibilité de « déplier » les formules à point fixe :

$$t, \rho, \delta \models \mu\xi.\phi \text{ si et seulement si } t, \rho, \delta \models \phi\langle \xi \rightarrow \mu\xi.\phi \rangle.$$

Pour garantir l'arrêt de cette méthode pour le model checking du $\mu$-calcul modal, [Winskel, 1991] enrichit les formules de point fixe avec une mémoire finie. Dans notre cas cela se traduit par une extension des formules de point fixe : les formules à point fixe seront de la forme $\mu\xi(M).\phi$, où $M$ est un ensemble fini d'arbres.

En ce qui concerne la quantification, prenons l'exemple des formules de quantification d'arbres. Dans une approche de model checking local, la satisfaction de la formule $\exists X.\phi$ par l'arbre $t$ s'écrit comme

$$t, \rho, \delta \models \exists X.\phi \text{ si et seulement s'il existe un arbre } s \text{ dans } \mathcal{A}_\Sigma \text{ tel que } t, \rho[X \rightarrow s], \delta \models \phi.$$

L'énumération de tous les arbres $s$ dans $\mathcal{A}_\Sigma$ est impossible. Nous résolvons ce problème en montrant que la satisfaction des formules à quantification vérifie la propriété suivante : il existe un ensemble fini et calculable d'arbres $S_t$ qui ne dépend que de $t$ et tel que

$$t, \rho, \delta \models \exists X.\phi \text{ si et seulement s'il existe un arbre } s \text{ dans } S_t \text{ tel que } t, \rho[X \rightarrow s], \delta \models \phi.$$

Nous montrons également un résultat similaire pour les formules à quantification sur les étiquettes.

Le model checking des formules à point fixe soulève également un problème de complexité. Considérons la formule $\mu\xi.\xi \mid \xi$. Un dépliage « naïf » tel que décrit précédemment résulte en

une croissance exponentielle de la taille du problème. En effet, un premier dépliage nous donne la formule $(\mu\xi.\xi \mid \xi) \mid (\mu\xi.\xi \mid \xi)$, et chaque dépliage successif double sa taille. Pour éviter ce coût en espace et d'après les idées de [Stirling and Walker, 1991], nous allons utiliser une méthode permettant d'effectuer les dépliages uniquement lorsque c'est nécessaire et qui va nous permettre d'avoir un algorithme s'exécutant en espace polynomial.

La suite du chapitre est organisée comme suit : nous commençons par donner quelques définitions dans la section 8.1. Nous présentons ensuite dans la section 8.2 deux propriétés de la satisfiabilité des formules à quantification qui vont nous garantir que le model checking peut se faire en ne considérant qu'un ensemble fini d'arbres et qu'un ensemble fini d'étiquettes pour la valuation des formules à quantification. Dans la section 8.3 nous présentons notre algorithme de model checking pour la logique spatiale, et la section 8.4 est consacrée à la preuve de correction de cet algorithme. Nous terminons par la section 8.5 qui présente la complexité en espace de l'algorithme de model checking proposé.

## 8.1 Définitions

Dans ce chapitre nous avons besoin de considérer une extension de la logique dans laquelle la syntaxe des formules à point fixe est donnée par $\mu\xi(M).\phi$, où $M$ est un ensemble fini d'arbres. Pour les valuations $\rho$ et $\delta$, l'interprétation de la formule $\mu\xi(M).\phi$ est

$$[\![\mu\xi(M).\phi]\!]_{\rho,\delta} = \bigcap\{S \mid S \supseteq [\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap \complement M\}.$$

Cette extension va être utilisée par l'algorithme de model checking que vous allons proposer : dans la formule $\mu\xi(M).\phi$, l'ensemble $M$ est utilisé comme une mémoire finie dont le rôle sera précisé dans la suite. Notons néanmoins que l'algorithme de model checking que nous allons définir ne prendra en entrée que des formules $\phi$ telles que pour toute sous formule $\mu\xi(M).\phi'$ de $\phi$, $M$ est l'ensemble vide. Il est clair que ces formules ont la même sémantique que les formules définies par la syntaxe originale de la logique présentée à la section 3.2.

**Définition 8.1 (Composant d'arbre)** L'arbre $t$ est *composant* de l'arbre $t'$, noté $t \sqsubseteq t'$, si (le multiensemble) $t$ est inclus dans (le multiensemble) $t'$ ou bien il existe $a[t'']$ un élément d'arbre qui est élément de $t'$ et $t$ est composant de $t''$ ; $t$ est *composant strict* de $t'$, noté $t \sqsubset t'$, si $t$ est un composant de $t'$ et $t$ est différent de $t'$.

Il est facile de voir que la relation de composant définit un ordre partiel sur les arbres.

Notons que la notion de composant d'arbre diffère de la notion de sous-arbre considérée habituellement pour arbres vu comme termes sur une signature finie. Dans le cas classique, un sous-arbre peut être défini de façon unique par un nœud de l'arbre. En utilisant la même intuition, un composant d'arbre est identifié par un *ensemble d'arêtes* parmi les arêtes successeurs d'un nœud de l'arbre. Pour éviter toute confusion, nous utilisons ici le terme « composant » plutôt que le terme « sous-arbre ».

**Définition 8.2 (Taille d'un arbre)** La *taille* d'un arbre $t = \langle N, E, \mathsf{etq}\rangle$, notée $taille(t)$, est le cardinal de l'ensemble des nœuds de l'arbre $N$. Pour un ensemble d'arbres $M$, sa taille $taille(M)$ est $\sum_{t\in M} taille(t)$.

Nous aurons besoin de désigner l'ensemble des variables libres d'étiquettes et d'arbres ou les variables libres de récursion d'une formule considérée seule ou relativement à une valuation $\rho$ ou $\delta$ des valuations de variables. Dans les définitions qui suivent, $\phi$ désigne une formule, $\rho$ une valuation des variables d'étiquettes et des variables d'arbres et $\delta$ une valuation des variables de récursion.

**Définition 8.3 (Variables libres relativement à une valuation)** Soit $\phi$ une formule, $\rho$ une valuation d'étiquettes et d'arbres et $\delta$ une valuation de variables de récursion. Alors $\mathsf{libres}(\phi, \rho)$ désigne l'ensemble de variables d'étiquettes et d'arbres qui sont libres dans $\phi$ et ne sont pas définies par la valuation $\rho$ et $\mathsf{libres}(\phi, \delta)$ désigne l'ensemble de variables de récursion libres dans $\phi$ et non définies par $\delta$ :

$$\begin{aligned}\mathsf{libres}(\phi, \rho) &= (\mathsf{libres}(\phi) \cap (\mathcal{X}_\mathrm{e} \cup \mathcal{X}_\mathrm{a})) \setminus dom(\rho),\\ \mathsf{libres}(\phi, \delta) &= (\mathsf{libres}(\phi) \cap \mathcal{X}_\mathrm{r}) \setminus dom(\delta).\end{aligned}$$

On dit que la formule $\phi$ est *close par* $\rho$ si l'ensemble $\mathsf{libres}(\phi, \rho)$ est vide et on dit que $\phi$ est *close par* $\delta$ si l'ensemble $\mathsf{libres}(\phi, \delta)$ est vide.

Pour un arbre, un ensemble d'arbres, une formule ou encore une valuation de variables, nous aurons besoin de désigner l'ensemble d'étiquettes qui apparaissent dans cet arbre, ensemble d'arbres, formule ou valuation.

**Définition 8.4 (Étiquettes)** L'ensemble des étiquettes d'un arbre $t$, noté $etiq(t)$ est défini récursivement sur la structure de $t$ par :

$$\begin{aligned}etiq(\{\!|e_1, \ldots, e_k|\!\}) &= \bigcup_{i \in 1..k} etiq(e_i) && \text{pour } e_1, \ldots, e_k \in \mathsf{EArbres}_\Sigma,\\ etiq(a[t']) &= \{a\} \cup etiq(t') && \text{pour } a \in \Sigma \text{ et } t' \in \mathcal{A}_\Sigma\end{aligned}$$

L'ensemble des étiquettes d'un ensemble d'arbres $M$, noté $etiq(M)$, est défini par

$$etiq(M) = \bigcup_{t \in M} etiq(t).$$

L'ensemble des étiquettes d'une formule $\phi$, noté $etiq(\phi)$, est défini récursivement sur la structure de $\phi$ par :

$$\begin{aligned}&etiq(\mathbf{0}) = etiq(\top) = etiq(X) = etiq(\xi) = \varnothing,\\ &etiq(\neg\phi) = etiq(\exists X.\phi) = etiq(\exists x.\phi) = etiq(\phi),\\ &etiq(\eta[\phi]) = (\{\eta\} \cap \Sigma) \cup etiq(\phi),\\ &etiq(\phi \,|\, \phi') = etiq(\phi \vee \phi') = etiq(\phi) \cup etiq(\phi'),\\ &etiq(\eta = \eta') = \{\eta, \eta'\} \cap \Sigma,\\ &etiq(\mu\xi(M).\phi) = etiq(M) \cup etiq(\phi).\end{aligned}$$

L'ensemble d'étiquettes de la valuation de variables d'étiquettes et variables d'arbres $\rho$, noté $etiq(\rho)$, est défini par

$$etiq(\rho) = \bigcup_{x \in dom(\rho) \cap \mathcal{X}_\mathrm{e}} \{\rho(x)\} \cup \bigcup_{X \in dom(\rho) \cap \mathcal{X}_\mathrm{a}} etiq(\rho(X)).$$

L'ensemble d'étiquettes de la valuation de variables de récursion $\delta$, noté $etiq(\delta)$, est défini par

$$etiq(\delta) = \bigcup_{\xi \in dom(\delta)} etiq(\delta(\xi)).$$

L'ensemble des variables de la formule $\phi$, la valuation de variables d'étiquettes et variables d'arbres $\rho$ et la valuation de variables de récursion $\delta$, noté $etiq(\phi, \rho, \delta)$, est défini par

$$etiq(\phi, \rho, \delta) = etiq(\phi) \cup etiq(\rho) \cup etiq(\delta).$$

Nous écrirons parfois $\rho(\eta)$ lorsque la nature de $\eta$ (càd étiquette ou variable d'étiquette) est inconnue, avec la convention que $\rho(\eta) = \eta$ lorsque $\eta$ est une étiquette.

**Définition 8.5 (Variables de récursion)** Pour une formule $\phi$, l'ensemble de variable de récursion (libres ou liées), noté $varrec(\phi)$, est défini récursivement sur la structure de $\phi$ par :

$varrec(\mathbf{0}) = varrec(\top) = varrec(X) = varrec(\eta = \eta') = \varnothing,$
$varrec(\neg\phi) = varrec(\exists X.\phi) = varrec(\exists x.\phi) = varrec(\eta[\phi]) = varrec(\mu\xi(M).\phi) = varrec(\phi),$
$varrec(\phi \mid \phi') = varrec(\phi \vee \phi') = varrec(\phi) \cup varrec(\phi'),$
$varrec(\xi) = \{\xi\}.$

Pour une formule $\phi$, nous désignons par $varrec(\phi)$ l'ensemble de variable de récursion (libres ou liées) apparaissant dans $\phi$.

Le problème de model checking tel que nous allons le considérer dans cette section est défini par :

> Étant donné un arbre $t$, une formule $\phi$ et une valuation $\rho$ telle que $\phi$ est close par $\rho$, décider si $t$ satisfait $\phi$ sous la valuation $\rho$.

## 8.2 Propriétés de la satisfiabilité pour les formules à quantification

Dans cette section nous prouvons quelques propriétés de la satisfiabilité des formules à quantification qui nous permettront d'obtenir un algorithme de model checking. Nous établissons dans la proposition 8.7 que pour tester si l'arbre $t$ satisfait la formule $\exists X.\phi$ il suffit de considérer un nombre fini d'arbres pour la valuation de la variable $X$. De la même façon, la proposition 8.9 établit que pour tester si $t$ satisfait $\exists x.\phi$, il suffit de considérer un nombre fini d'étiquettes pour la valuation de $x$. Les lemmes 8.6 et 8.8 annoncent des résultats plus généraux sur les propriétés de la satisfiabilité et seront utilisés pour les preuves des propositions 8.7 et 8.9.

**Lemme 8.6** *Soient $t$ un arbre, $\rho$ et $\delta$ des valuations de variables et $\phi$ une formule tels que $\phi$ est close par $\delta$ et* libres$(\phi, \rho) \subseteq \{X\}$*, pour une certaine variable d'arbre $X$. Soient $s$ et $s'$ deux arbres qui ne sont pas composants de $t$. Alors $t, \rho[X \to s], \delta \models \phi$ si et seulement si $t, \rho[X \to s'], \delta \models \phi$.*

*Preuve* Voir la section A.4 de l'annexe A.

□

Comme conséquence immédiate du précédent lemme, nous avons

**Proposition 8.7** *Soit $t$ un arbre et $\exists X.\phi$ une formule close par la valuation $\rho$ et sans variables de récursion libres. Soit $b$ une étiquette qui n'appartient pas à $etiq(t)$. Alors $t, \rho \models \exists X.\phi$ si et seulement s'il existe un arbre $s$ dans $\{s' \mid s' \sqsubseteq t\} \cup \{\{\!|b[\{\!|\}\!|]|\!\}\}$ tel que $t, \rho[X \rightarrow s], \delta \models \phi$.*

*Preuve* Supposons qu'il existe un arbre $s$ dans $\{s' \mid s' \sqsubseteq t\} \cup \{\{\!|b[\{\!|\}\!|]|\!\}\}$ tel que $t, \rho[X \rightarrow s] \models \phi$. Alors, par les propriétés de la satisfiabilité, $t, \rho \models \exists X.\phi$. Supposons maintenant que $t, \rho \models \exists X.\phi$. Alors il existe un arbre $s$ tel que $t, \rho[X \rightarrow s] \models \phi$. Si $s$ est un composant de $t$, alors la conclusion est immédiate. Sinon, comme l'étiquette $b$ n'est pas dans $etiq(t)$, l'arbre $\{\!|b[\{\!|\}\!|]|\!\}$ n'est pas composant de $t$ et donc, par le lemme 8.6 et toute valuation vide $\delta$, $t, \rho[X \rightarrow \{\!|b[\{\!|\}\!|]|\!\}], \delta \models \phi$.

□

Un résultat similaire existe sur sur la quantification sur les étiquettes, néanmoins il ne s'annonce pas aussi simplement. C'est pourquoi nous commençons par introduire quelques notions, qui seront nécessaire pour énoncer ce résultat. Pour toute étiquettes $a, b$ et tout arbre $t$, $t\langle a \leftrightarrow b\rangle$ désigne l'arbre obtenu à partir de $t$ en échangeant les étiquettes $a$ et $b$. Formellement,

$$\begin{aligned}
\{\!|e_1, \ldots, e_k|\!\}\langle a \leftrightarrow b\rangle &= \{\!|e_1\langle a \leftrightarrow b\rangle, \ldots, e_k\langle a \leftrightarrow b\rangle|\!\} \\
a[t']\langle a \leftrightarrow b\rangle &= b[t'\langle a \leftrightarrow b\rangle] \\
b[t']\langle a \leftrightarrow b\rangle &= a[t'\langle a \leftrightarrow b\rangle] \\
c[t']\langle a \leftrightarrow b\rangle &= c[t'\langle a \leftrightarrow b\rangle] \qquad \text{pour } c \neq a \text{ et } c \neq b,
\end{aligned}$$

où les $e_i$ sont des éléments d'arbres et $t'$ est un arbre quelconque. Soit $\rho_{ab}$ une valuation qui à chaque variable d'étiquette associe $a$ ou bien $b$, càd pour tout $x$ dans $dom(\rho_{ab}) \cap \mathcal{X}_e$, $\rho_{ab}(x) \in \{a, b\}$. Alors on note $\overline{\rho_{ab}}$ la valuation définie sur le même domaine que $\rho_{ab}$ et qui échange les étiquettes $a$ et $b$ dans les valeurs (étiquettes ou arbres) données par $\rho_{ab}$. Formellement,

- $dom(\overline{\rho_{ab}}) = dom(\rho_{ab})$ ;
- pour tout $x$ dans $dom(\overline{\rho_{ab}}) \cap \mathcal{X}_e$,

$$\overline{\rho_{ab}}(x) = \begin{cases} a & \text{si } \rho_{ab}(x) = b \\ b & \text{si } \rho_{ab}(x) = a \end{cases}$$

- pour tout $X$ dans $dom(\overline{\rho_{ab}}) \cap \mathcal{X}_a$, $\overline{\rho_{ab}}(X) = \rho_{ab}(X)\langle a \leftrightarrow b\rangle$.

Pour deux valuations $\rho$ et $\rho'$ dont les domaines sont disjoints, nous notons par $\rho\rho'$ la valuation de domaine $dom(\rho) \cup dom(\rho')$ et qui à chaque $\chi$ dans $dom(\rho)$ associe $\rho(\chi)$ et à chaque $\chi$ dans $dom(\rho')$ associe $\rho'(\chi)$.

**Lemme 8.8** *Soient $t$ un arbre, $\rho$ et $\delta$ des valuations et $\phi$ une formule tels que $\phi$ est close par $\delta$ et* libres$(\phi, \rho) \subseteq \{x_1, \ldots, x_k, X_1, \ldots, X_{k'}\}$, *où $x_1, \ldots, x_x$ sont des variables d'étiquettes et $X_1, \ldots, X_{k'}$ sont des variables d'arbres. Soient $a$ et $b$ des étiquettes qui n'appartiennent pas à $etiq(t)$ ni à $etiq(\phi, \rho, \delta)$ et soit $\rho_{ab}$ une valuation de domaine $\{x_1, \ldots, x_k, X_1, \ldots, X_{k'}\}$ et telle que pour chaque $i$ dans $1..k$, $\rho_{ab}(x_i) \in \{a, b\}$. Alors $t, \rho\rho_{ab}, \delta \models \phi$ si et seulement si $t, \rho\overline{\rho_{ab}}, \delta \models \phi$.*

*Preuve* Voir la section A.4 de l'annexe A.

□

Comme conséquence immédiate du précédent lemme, nous avons :

**Proposition 8.9** *Soit $t$ un arbre et $\exists x.\phi$ une formule close par la valuation $\rho$ et sans variables de récursion libres. Soit $A$ un ensemble fini d'étiquettes tel que $etiq(\phi, \rho) \subseteq A$ et soit $a$ une étiquette qui n'appartient pas à $etiq(t) \cup A$. Alors $t, \rho \models \exists x.\phi$ si et seulement s'il existe une étiquette $b$ dans l'ensemble $etiq(t) \cup A \cup \{a\}$ pour laquelle $t, \rho[x \rightarrow b] \models \phi$.*

*Preuve* Supposons d'abord que l'étiquette $b$ existe, alors par définition même de la satisfiabilité des formules, $t, \rho \models \exists x.\phi$. Supposons maintenant que $t, \rho \models \exists x.\phi$. Alors, par définition de la satisfiabilité, il existe une étiquette $b$ telle que $t, \rho[x \rightarrow b] \models \phi$. Si $b$ est dans l'ensemble $etiq(t) \cup A \cup \{a\}$, la conclusion est immédiate. Sinon, soit $\rho_{ab}$ la valuation de domaine $\{x\}$ et telle que $\rho_{ab}(x) = a$. Comme les étiquettes $a$ et $b$ n'appartiennent pas à $etiq(t) \cup A$, par définition de $A$ elles n'appartiennent non plus à $etiq(t) \cup etiq(\phi, \rho, \delta)$, où $\delta$ est la valuation de domaine vide. Alors du lemme 8.8 nous déduisons que $t, \rho\rho_{ab} \models \phi$ si et seulement si $t, \rho\overline{\rho_{ab}} \models \phi$. Il suffit alors de remarquer que $\rho\rho_{ab} = \rho[x \rightarrow a]$ et $\rho\overline{\rho_{ab}} = \rho[x \rightarrow b]$.

□

## 8.3 Algorithme de model checking

Comme nous l'avons déjà remarqué, si le model checking de la formule à point fixe $\mu\xi.\phi$ est fait en remplaçant explicitement dans $\phi$ les occurrences de la variable de récurrence $\xi$ par la formule $\mu\xi.\phi$, la taille des formules croît de façon exponentielle lorsque $\phi$ contient de multiples occurrences de la variable $\xi$. Pour éviter cette explosion en taille nous allons faire ce remplacement uniquement lorsque c'est nécessaire et nous allons stocker les substitutions à être appliquées à une formule dans une structure à part qui contiendra une occurrence unique de la formule $\mu\xi(M).\phi$ pour toutes les occurrences de la variable $\xi$ apparaissant dans $\phi$.

Une *substitution* $\Gamma$ est une séquence finie de la forme $[\xi_1 \rightarrow \phi_1] \cdots [\xi_n \rightarrow \phi_n]$, où les $\xi_i$ sont des variables de récursion et les $\phi_i$ sont des formules logiques. De plus, nous imposons que $\xi_i \neq \xi_j$ si $i \neq j$. La séquence vide est notée $\epsilon$.

Pour la formule logique $\phi$ et la substitution $\Gamma$, l'application de $\Gamma$ à $\phi$ est une formule notée $\Gamma(\phi)$ et définie récursivement sur la structure de $\Gamma$ comme suit :

- $\epsilon(\phi) = \phi$ ;
- $[\xi \rightarrow \phi']\Gamma(\phi) = (\Gamma(\phi))\langle \xi \rightarrow \phi' \rangle$ (nous rappelons que pour toutes formules $\phi, \phi'$, $\phi\langle \xi \rightarrow \phi' \rangle$ est la formule obtenue en remplaçant dans $\phi$ les occurrences libres de la variable $\xi$ par $\phi'$).

Le *domaine* de la substitution $\Gamma$, noté $dom(\Gamma)$, est défini par : $dom(\epsilon) = \varnothing$ et $dom([\xi \rightarrow \phi]\Gamma) = \{\xi\} \cup dom(\Gamma)$.

On note par $\Gamma\langle \xi \rightarrow \phi \rangle$ la substitution définie par :

- si $\xi$ n'est pas dans le domaine de $\Gamma$, alors $\Gamma\langle \xi \rightarrow \phi \rangle = \Gamma[\xi \rightarrow \phi]$ ;
- si $\xi$ est dans le domaine de $\Gamma$ et $[\xi \rightarrow \phi']$ est élément de $\Gamma$, alors $\Gamma\langle \xi \rightarrow \phi \rangle$ est la substitution identique à $\Gamma$ mais dans laquelle $[\xi \rightarrow \phi']$ a été remplacé par $[\xi \rightarrow \phi]$.

L'ensemble des étiquettes apparaissant dans la substitution $\Gamma$, noté $etiq(\Gamma)$, est défini par : $etiq(\epsilon) = \varnothing$, $etiq([\xi \rightarrow \phi]\Gamma) = etiq(\phi) \cup etiq(\Gamma)$. Pour toute formule $\phi$, toute valuation $\rho$ et toute substitution $\Gamma$, nous notons par $etiq(\phi, \rho, \Gamma)$ l'ensemble d'étiquettes $etiq(\phi) \cup etiq(\rho) \cup etiq(\Gamma)$.

On présente sur la figure 1 un algorithme de model checking check$(t, \phi, \rho, \Gamma)$, où $t$ est un arbre, $\phi$ est une formule logique, $\rho$ est une valuation de variables d'étiquettes et $\Gamma$ est une substitution. L'algorithme est correct dans le sens où pour tout arbre $t$, toute formule $\phi$ et valuation $\rho$ telles que $\phi$ est close par $\rho$ et de plus toute sous formule $\mu\xi(M).\phi'$ de $\phi$ vérifie $M = \varnothing$, check$(t, \phi, \rho, \epsilon)$ termine et retourne vrai si $t \models \phi$ et faux sinon.

## 8.4 Preuve de correction de l'algorithme

Nous donnons d'abord quelques invariants de l'algorithme (proposition 8.11), ensuite nous montrons sa terminaison (proposition 8.13) et finalement sa correction (théorème 8.14).

### 8.4.1 Invariants de l'algorithme

On dit que la substitution $\Gamma$ est *close par préfixe* si $\Gamma$ est la substitution vide $\epsilon$ ou bien pour $\Gamma = [\xi_1 \to \phi_1] \cdots [\xi_n \to \phi_n]$, si la variable $\xi$ est libre dans la formule $\phi_i$ (pour $i$ dans $1..n$), alors il existe $j < i$ tel que $\xi_j = \xi$.

Il est facile de voir à partir de cette définition que

**Propriété 8.10** *Si la substitution $\Gamma$ est close par préfixe et les variables de récursion libres de la formule $\phi$ appartiennent au domaine de $\Gamma$, alors la formule $\Gamma(\phi)$ est sans variables de récursion libres.*

**Proposition 8.11** *Soit $\phi_I$ une formule sans variables de récursion libres et soit $\rho_I$ une valuation telle que $\phi_I$ est close par $\rho_I$. Pour tout appel récursif* check$(t, \phi, \rho, \Gamma)$ *généré par l'évaluation de* check$(t_I, \phi_I, \rho_I, \epsilon)$*, les propriétés suivantes sont vérifiées :*

*(i) la formule $\phi$ est close par $\rho$ ;*

*(ii) $\Gamma$ est close par préfixe ;*

*(iii) $\phi$ est sous formule de $\phi_I$ ;*

*(iv) $t$ est composant de $t_I$ ;*

*(v) les variables de récursion libres de $\phi$ appartiennent à $dom(\Gamma)$ ;*

*(vi) $dom(\Gamma)$ est inclus dans $varrec(\phi_I)$ ;*

*(vii) pour tout $\xi$ du domaine de $\Gamma$, si $[\xi \to \phi']$ est l'élément de $\Gamma$ correspondant à $\xi$, alors la formule $\phi'$ est de la forme $\mu\xi(M).\phi$ telle que $\mu\xi(\varnothing).\phi$ est sous formule de $\phi_I$. De plus, l'ensemble $M$ est égal à $\{t_1, \ldots, t_n\}$, où les arbres $t_1, \ldots, t_n$ vérifient $t \sqsubseteq t_n \sqsubset t_{n-1} \sqsubset \cdots \sqsubset t_1 \sqsubseteq t_I$ ;*

*(viii) pour $\Gamma = [\xi_1 \to \mu\xi_1(M_1).\phi_1] \cdots [\xi_k \to \mu\xi_k(M_k).\phi_k]$ et pour tous $i, j$ dans $1..k$, si $\xi_j$ est libre dans $\phi_i$, alors $\phi_i$ est sous formule de $\phi_j$.*

*Preuve* Remarquons d'abord que, comme dans la formule $\phi_I$ deux occurrences distinctes de l'opérateur de plus petit point fixe lient deux variables différentes, pour le point (vii) de la proposition il existe une unique formule $\phi$ telle que $\mu\xi(\varnothing).\phi$ est sous formule de $\phi_I$.

La preuve se fait par induction sur la structure de l'arbre d'exécution de l'algorithme check, sauf pour le point (viii) dont la preuve est donnée en dernier.

check$(t, \phi, \rho, \Gamma)$ is
  case $\phi$ of
    $\mathbf{0}$ : retourner $t = \{\!|\}\!|$ ;
    $\eta[\phi']$ : retourner $t = \{\!|\rho(\eta)[t']|\!\}$ et check$(t', \phi', \rho, \Gamma)$ ;
    $\phi' \mid \phi''$ : pour chaque $t', t''$ tel que $t = t' \uplus t''$
      faire
        si check$(t', \phi', \rho, \Gamma)$ et check$(t'', \phi'', \rho, \Gamma)$
        alors retourner vrai ;
        fin si ;
      fin pour ;
      retourner faux ;
    $\top$ : retourner vrai ;
    $\neg\phi'$ : retourner non check$(t, \phi', \rho, \Gamma)$ ;
    $\phi' \vee \phi''$ : retourner check$(t, \phi', \rho, \Gamma)$ ou check$(t, \phi'', \rho, \Gamma)$ ;
    $\exists x.\phi'$ : soit $a$ une étiquette n'appartenant ni à $etiq(t)$ ni à $etiq(\phi', \rho, \Gamma)$ dans
      pour chaque $b \in etiq(t) \cup etiq(\phi', \rho, \Gamma) \cup \{a\}$
      faire
        si check$(t, \phi', \rho[x \rightarrow b], \Gamma)$
        alors retourner vrai ;
        fin si ;
      fin pour ;
      retourner faux ;
      fin soit ;
    $X$ : retourner $t = \rho(X)$ ;
    $\exists X.\phi'$ : soit $a$ une étiquette n'appartenant pas à $etiq(t)$ dans
      pour chaque $s \in \{s' \mid s' \sqsubset t\} \cup \{\{\!|a[\{\!|\}\!|]|\!\}\}$
      faire
        si check$(t, \phi', \rho[X \rightarrow s], \Gamma)$
        alors retourner vrai ;
        fin si ;
      fin pour ;
      retourner faux ;
      fin soit ;
    $\eta = \eta'$ : retourner $\rho(\eta) = \rho(\eta')$ ;
    $\mu\xi(M).\phi'$ : si $t \in M$ alors retourner faux ;
      sinon retourner check$(t, \phi', \rho, \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle)$ ;
      fin si ;
    $\xi$ : si $\xi \notin dom(\Gamma)$
      alors retourner erreur ;
      fin si ;
      soit $[\xi \rightarrow \mu\xi(M).\phi']$ élément de $\Gamma$ dans
        si $t \in M$ alors retourner faux ;
        sinon retourner check$(t, \phi', \rho, \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle)$ ;
        fin si ;
      fin soit ;
  fin case ;

FIG. 1 – Algorithme de model checking.

Cas de base: On montre que les propriétés sont vérifiées pour $\mathsf{check}(t, \phi_I, \rho_I, \epsilon)$. Le point (i) est vrai par hypothèse. Les points (ii), (vi) et (vii) sont vérifiés puisque la substitution $\Gamma$ est vide et pour le point (v), par hypothèse la formule $\phi_I$ n'admet pas de variables de récursion libres. Les points (iii) et (iv) sont triviaux.

Cas d'induction: Pour le pas d'induction, nous considérons tous les appels récursifs possibles dans l'algorithme. Montrons d'abord le point (iv) qui est indépendant des autres. Supposons que l'appel $\mathsf{check}(t, \phi, \rho, \Gamma)$ génère un appel récursif direct à $\mathsf{check}(t', \phi', \rho', \Gamma')$[1] et que $t$ est composant de $t_I$. Pour les cas où $\phi$ est l'une des formules $\eta[\phi']$ ou $\phi' \mid \phi''$, la propriété est facile à déduire de l'hypothèse de récurrence et de la transitivité de la relation de composant. Pour tous les autres cas, $t' = t$ et la propriété est vraie par hypothèse de récurrence.

Prouvons maintenant les autres points. Faisons d'abord quelques remarques générales. Si $\rho = \rho'$ et les formules $\phi$ et $\phi'$ ont les mêmes ensembles de variables d'étiquettes et d'arbres libres, alors le point (i) Cette condition n'est pas vérifiée pour $\phi$ étant une des formules $\exists x.\phi'$, $\exists X.\phi'$, $\xi$ ou $\mu\xi(M).\phi'$. Si $\Gamma = \Gamma'$ et les formules $\phi$ et $\phi'$ ont les mêmes ensembles de variables de récursion libres, alors le point (v) est vérifié. Cette condition exclut le cas où $\phi$ est l'une des formules $\xi$ ou $\mu\xi(M).\phi'$. Si $\Gamma = \Gamma'$, alors les points (ii), (vi) et (vii) sont vérifiés. Cette condition exclut les cas où $\phi$ est l'une des formules $\xi$ ou $\mu\xi(M).\phi'$. Finalement, si $\phi'$ est sous formule de $\phi$, alors le point (iii) est vérifié par transitivité de la relation de sous formule. Cette condition n'est pas vérifié pour $\phi$ étant la formule $\xi$.

Considérons maintenant les cas qui ne sont pas couverts par ces remarques.

Si $\phi = \exists x.\phi'$, l'appel récursif est $\mathsf{check}(t, \phi', \rho[x \to a], \Gamma))$, on doit prouver le point (i). Il est facile de voir que $\mathsf{libres}(\phi', \rho) = \mathsf{libres}(\phi, \rho) \cup \{x\}$ et, comme $dom(\rho') = dom(\rho) \cup \{x\}$, le point (i) est vérifié.

Si $\phi = \exists X.\phi'$, l'appel récursif est $\mathsf{check}(t, \phi, \rho[X \to s], \Gamma))$, encore une fois nous devons montrer le point (i) et la preuve est similaire au cas précédent.

Si $\phi = \mu\xi(M).\phi'$, l'appel récursif est $\mathsf{check}(t, \phi', \rho, \Gamma\langle\xi \to \mu\xi(M \cup \{t\}).\phi'\rangle)$. On doit montrer les points (ii), (v), (vi) et (vii). Montrons d'abord que $\Gamma$ est close par préfixe. Soit $\Gamma = [\xi_1 \to \phi_1] \cdots [\xi_n \to \phi_n]$. Si $\xi \in dom(\Gamma)$, soit $\xi = \xi_i$ : par hypothèse de récurrence (point (vii)), $\phi_i = \mu\xi(M_i).\phi_i'$ et $\mu\xi(\varnothing).\phi_i'$ est sous formule de $\phi_I$. D'autre part, par hypothèse de récurrence (point (iii)), $\phi'$ et sous formule de $\phi_I$ et donc $\phi_i'$ et $\phi'$ sont la même formule, ce qui implique que $\Gamma$ est close par préfixe. Si $\xi \notin dom(\Gamma)$, alors $\Gamma' = \Gamma[\xi \to \mu\xi(M \cup \{t\}).\phi']$. De plus, par hypothèse de récurrence (point (v)), les variables de récursion libres de $\mu\xi(M).\phi'$ appartiennent à $dom(\Gamma)$, ce qui implique que $\Gamma$ est close par préfixe et termine la preuve du point (ii). Le point (v) est conséquence immédiate de l'hypothèse de récurrence. Le point (vi) est vérifié puisque, par hypothèse de récurrence (point (iii)) $\phi$ est sous formule de $\phi_I$ et donc la variable $\xi$ est dans $varrec(\phi_I)$. Il nous reste à montrer le point (vii). Par hypothèse de récurrence (point (v)), $\phi = \mu\xi(\varnothing).\phi'$ est sous formule de $\phi_I$ et donc $M = \varnothing$ et $M \cup \{t\} = \{t\}$. Par hypothèse de récurrence (point (iv)), $t$ est composant de $t_I$, ce qui est suffisant pour conclure pour le point (vii).

Si $\phi = \xi$, l'appel récursif est $\mathsf{check}(t, \phi', \rho, \Gamma\langle\xi \to \mu\xi(M \cup \{t\}).\phi'\rangle)$ et l'algorithme garantit que $[\xi \to \mu\xi(M).\phi']$ fait partie de $\Gamma$. Nous devons prouver les points (ii), (v), (vii) et (iii). Soit $\Gamma = [\xi_1 \to \phi_1] \cdots [\xi_n \to \phi_n]$, et nous savons que $\xi \in dom(\Gamma)$ ; soit $\xi = \xi_i$. En utilisant

[1] càd $\mathsf{check}(t', \phi', \rho', \Gamma')$ est fils de $\mathsf{check}(t, \phi, \rho, \Gamma)$ dans l'arbre d'exécution

la définition de $\Gamma' = \Gamma\langle\xi \to \mu\xi(M \cup \{t\}).\phi'\rangle$ et l'hypothèse de récurrence (point (ii)), il est facile de voir que $\Gamma$ est close par préfixe, ce qui montre le point (ii). De plus, $\Gamma$ étant close par préfixe et en utilisant que $\phi = \phi_i$, les variables de récursion libres de $\mu\xi(M).\phi'$ appartiennent à $\{\xi_1, \ldots, \xi_{i-1}\}$ et donc le point (v) est vérifié. Pour le point (vi), il suffit de voir que $dom(\Gamma) = dom(\Gamma')$. Montrons maintenant le point (vii). Par hypothèse de récurrence (point (iii)), $\mu\xi(\varnothing).\phi'$ est sous formule de $\phi_I$ et $M = \{t_1, \ldots, t_k\}$ et tel que $t$ est composant de $t_i$ pour tout $i$ dans $1..k$. De plus, par hypothèse de récurrence (point (vii)), $t_k \sqsubseteq \cdots \sqsubseteq t_1 \sqsubset t_I$ et comme $t \notin M$, $t$ est composant strict de $t_k$, ce qui permet de conclure.

Montrons maintenant le point (viii). D'après le point (vii), pour tout $i$ dans $1..k$, $\mu\xi_i(\varnothing).\phi_i$ est sous formule de $\phi_I$. Donc, si $\phi_i$ contient une occurrence libre de $\xi_j$, alors $\phi_i$ est sous l'opérateur $\mu\xi_j$ dans $\phi_I$ et alors $\phi_i$ est sous formule de $\phi_j$.

□

### 8.4.2 Terminaison de l'algorithme

La preuve de terminaison de l'algorithme utilise une relation d'ordre Noethérienne entre les appels récursifs de l'algorithme. Nous définissons d'abord cette relation d'ordre.

Soit $<_{\text{sf}}$ la relation de sous formule stricte, càd $\phi <_{\text{sf}} \phi'$ si la formule $\phi$ est sous formule stricte de la formule $\phi'$.

Pour chaque arbre $t$, nous définissons la relation binaire $<'_t$ (en notation infixe) sur les ensembles finis d'arbres comme suit : $M' <'_t M$ si les ensembles $M$ et $M'$ vérifient l'une de ces conditions :

(i) $M = \{t_1, \ldots, t_k\}$ et $M' = \{t_1, \ldots, t_k, t_{k+1}\}$ et $t_{k+1} \sqsubset t_k$ ;

(ii) $M = \{t_1, \ldots, t_k\}$ et $M' = \{t_{k+1}\}$ et $t_{k+1} \sqsubseteq t_k$,

où les arbres $t_1, \ldots, t_k$ (pour $k$ éventuellement nul) vérifient $t_k \sqsubset t_{k-1} \sqsubset \cdots \sqsubset t_1 \sqsubseteq t$.

On étend la relation $<'_t$ sur les substitutions par : $\Gamma' <'_t \Gamma$ si les substitutions $\Gamma, \Gamma'$ vérifient l'une des conditions suivantes :

(i) $dom(\Gamma) \subseteq dom(\Gamma')$ ;

(ii) $dom(\Gamma) = dom(\Gamma')$ et il existe une variable $\xi$ dans $dom(\Gamma)$ telle que, si $[\xi \to \mu\xi(M').\phi]$ et $[\xi \to \mu\xi(M).\phi]$ sont éléments de $\Gamma'$ et $\Gamma$ respectivement, alors $M' <_t M$ et pour tout $\xi' \neq \xi$ dans $dom(\Gamma)$, pour $[\xi' \to \psi]$ élément de $\Gamma$ et $[\xi' \to \psi']$ élément de $\Gamma'$, on a $\psi = \psi'$.

Soient $t_I$, $\phi_I$ et $\rho_I$ un arbre, une formule et une valuation respectivement avec la condition que $\phi_I$ est close par $\rho_I$. Nous définissons la relation $<_{t_I}$ entre les instances de l'algorithme check générées par l'évaluation de $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$ par : $\mathsf{check}(t', \phi', \rho', \Gamma') <_{t_I} \mathsf{check}(t, \phi, \rho, \Gamma)$ si (i) $\Gamma' <'_{t_I} \Gamma$ ou bien (ii) $\Gamma' = \Gamma$ et $\phi' <_{\text{sf}} \phi$.

**Lemme 8.12** *Soient $t_I$ un arbre et soient $\phi_I$ un formule et $\rho_I$ une valuation telles que $\phi_I$ est close par $\rho_I$. La relation $<_{t_I}$ est une relation d'ordre Noethérienne sur l'ensemble des instances de l'algorithme* check *générées par l'évaluation de* $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$.

*Preuve* Pour chaque ensemble d'arbres $M$ de la forme $\{t_1, \ldots, t_k\}$ et tel que $t_k \sqsubset \cdots \sqsubset t_1 \sqsubseteq t_I$, soit $\mathcal{M}_{\text{ens}}(M)$ le nombre naturel défini par : si $k = 1$, alors $\mathcal{M}_{\text{ens}}(M) = 2 * taille(t_1) - 1$ et si $k > 1$, alors $\mathcal{M}_{\text{ens}}(M) = 2 * taille(t_k)$.

Montrons que $<'_{t_I}$ est une relation d'ordre sur les substitutions $\Gamma$ telles que $dom(\Gamma)$ est inclus dans $varrec(\phi_I)$, ce qui par la proposition 8.11 (point (vi)) est effectivement le cas pour les substitutions $\Gamma$ telles que $\mathsf{check}(t, \phi, \rho, \Gamma)$ est instance de l'algorithme check générée par l'évaluation de $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$. Pour un tel $\Gamma$, soit $\mathcal{M}_{\text{subs}}(\Gamma)$ le couple dans $\mathbb{N}^2$ défini par

$$\mathcal{M}_{\text{subs}}(\Gamma) = \left( |varrec(\phi)| - |dom(\Gamma)| \, , \sum_{\xi \in varrec(\phi)} \mathcal{M}'_{\text{ens}}(\Gamma, \xi) \right)$$

où, pour tout $\xi$ dans $varrec(\phi)$, $\mathcal{M}'_{\text{ens}}(\Gamma, \xi)$ est défini par :

(i) si $\xi$ est dans $dom(\Gamma)$ et $[\xi \rightarrow \mu\xi(M).\phi']$ est l'élément correspondant, alors $\mathcal{M}'_{\text{ens}}(\Gamma, \xi) = \mathcal{M}_{\text{ens}}(M)$ ;

(ii) si $\xi$ n'est pas dans $dom(\Gamma)$, alors $\mathcal{M}_{\text{ens}}(\Gamma, \xi) = 2 * taille(t_I)$.

Il est facile de voir que la relation $<'_{t_I}$ fait décroître au moins un des éléments du couple $\mathcal{M}_{\text{subs}}(\Gamma)$, et comme le nombre de tels couples comparables (càd pouvant être mis en relation) par $<'_{t_I}$ est fini, nous en déduisons que $<'_{t_I}$ est un ordre partiel. Par conséquent, $<_{t_I}$ est un ordre lexicographique composé des relations d'ordre $<'_{t_I}$ et $<_{\text{sf}}$.

Montrons maintenant que $<_{t_I}$ est un ordre Noethérien pour l'ensemble des instances de l'algorithme check générées par l'évaluation de $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$. Nous savons que $<'_{t_I}$ fait décroître $\mathcal{M}_{\text{subs}}(\Gamma)$. D'autre part, $<_{\text{sf}}$ fait décroître la taille des formules, càd $\phi <_{\text{sf}} \phi'$ si et seulement si $taille(\phi) < taille(\phi')$. Donc, pour chaque instance $\mathsf{check}(t, \phi, \rho, \Gamma)$ générée par l'évaluation de $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$, $<_{t_I}$ fait décroître le couple d'entiers $(\mathcal{M}_{\text{subs}}(\Gamma), taille(\phi))$. Et comme $\mathcal{M}_{\text{subs}}(\epsilon)$ et $taille(\phi_I)$ sont des nombres naturels, la plus longue chaîne décroissante pour l'ordre $<_{t_I}$ a une longueur inférieure au produit de ces deux nombres. Ceci prouve que la relation $<_{t_I}$ est Noethérienne.

□

**Proposition 8.13 (Terminaison de l'algorithme)** *Soient $\phi_I$ une formule sans variables de récursion libres et $\rho_I$ une valuation telles que $\phi_I$ est close par $\rho_I$. Alors pour tout arbre $t_I$, l'évaluation de* $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$ *s'arrête et retourne* vrai *ou* faux.

*Preuve* On sait par la proposition 8.11 (point (v)) que pour tout appel récursif $\mathsf{check}(A, \phi, \rho, \Gamma)$ généré par l'évaluation de $\mathsf{check}(A_I, \phi_I, \rho_I, \epsilon)$, les variables de récursion libres de la formule $\phi$ appartiennent à $dom(\Gamma)$. Donc, la valeur erreur ne peut pas calculée comme résultat de l'algorithme.

Pour la terminaison de l'algorithme, il suffit de montrer que si $\mathsf{check}(t', \phi', \rho', \Gamma')$ est un appel récursif généré par l'évaluation de $\mathsf{check}(t, \phi, \rho, \Gamma)$, alors $\mathsf{check}(t', \phi', \rho', \Gamma') <_{t_I} \mathsf{check}(t, \phi, \rho, \Gamma)$. La plupart des cas sont immédiats puisque $\Gamma = \Gamma'$ et $\phi$ est sous formule stricte de $\phi$. Considérons les cas restants.

Si $\phi = \xi$, alors $\phi'$ est tel que $[\xi \rightarrow \mu\xi(M).\phi']$ est élément de $\Gamma$ et $\Gamma' = \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle$. Alors $dom(\Gamma') = dom(\Gamma)$ et par le point (vii) de la proposition 8.11 il est facile de voir que $M \cup \{t\} <_{t_I} M$, donc $\Gamma' <_{t_I} \Gamma$.

Si $\phi = \mu\xi(M).\psi$, alors $\phi' = \psi$ et $\Gamma' = \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\psi\rangle$. Si $\xi$ n'appartient pas au domaine de $\Gamma$, alors $dom(\Gamma)$ est inclus dans $dom(\Gamma')$ et la conclusion est immédiate par définition de $<_{t_I}$. Si $\xi$ appartient au domaine de $\Gamma$, soit $[\xi \rightarrow \psi']$ l'élément de $\Gamma$. Nous savons par

le point (vii) de la proposition 8.11 que $\psi'$ est égal à $\mu\xi(\{t_1, \ldots, t_n\}).\psi$, où les arbres $t_1, \ldots, t_n$ vérifient $t \sqsubseteq t_n \sqsubset t_{n-1} \sqsubset \cdots \sqsubset t_i \sqsubseteq t_I$. D'autre part, par le point (iii) de la proposition 8.11 nous savons que $\mu\xi(M).\psi$ est sous formule de $\phi_I$ et alors $M = \varnothing$. Donc $M \cup \{t\} = \{t\}$ et, par définition, $\{t\} <_{t_I} \{t_1, \ldots, t_n\}$. La conclusion est immédiate par définition de $<_{t_I}$.

□

### 8.4.3 Correction de l'algorithme

Cette section est consacrée à la preuve du théorème qui suit.

**Théorème 8.14 (Correction de l'algorithme)** *Soient $\phi_I$ une formule sans variables de récursion libres et $\rho_I$ une valuation telles que $\phi_I$ est close par $\rho_I$. Alors pour tout arbre $t_I$, l'algorithme* check$(t_I, \phi_I, \rho_I, \epsilon)$ *retourne* vrai *si et seulement si $A_I, \rho_I \models \phi_I$ est vrai.*

Nous commençons par donner quelques résultats préliminaires avant de donner la preuve du théorème (page 184).

Le lemme qui suit est une adaptation du résultat de Winskel [Winskel, 1991] établi pour le $\mu$-calcul avec un opérateur de plus grand point fixe.

**Lemme 8.15** *Si $F$ est une fonction monotone de $\wp(\mathcal{A}_\Sigma)$ dans $\wp(\mathcal{A}_\Sigma)$, alors pour tout arbre $t$, $t$ appartient à $\mu\xi.F(\xi)$ si et seulement si $t$ appartient à $F(\mu\xi.(F(\xi) \cap \complement\{t\}))$.*

*Preuve* Soit $G$ la fonction $\xi \mapsto F(\xi) \cap \complement\{t\}$. Notons que $G$ est monotone. Nous montrons que $t \notin \mu\xi.F(\xi) \Leftrightarrow t \notin F(\mu\xi.G(\xi))$, ce qui est équivalent à l'énoncé du lemme.

Supposons d'abord que $t \notin \mu\xi.F(\xi)$. Nous avons la suite d'implications suivante :

$$
\begin{array}{lcl}
t \notin \mu\xi.F(\xi) & \Rightarrow & \\
(\mu\xi.F(\xi)) \cap \complement\{t\} = \mu\xi.F(\xi) & \Rightarrow & \text{(plus petit point fixe pour } F\text{)} \\
F(\mu\xi.F(\xi)) \cap \complement\{t\} = \mu\xi.F(\xi) & \Rightarrow & \text{(définition de } G\text{)} \\
G(\mu\xi.F(\xi)) = \mu\xi.F(\xi) & \Rightarrow & \\
\mu\xi.F(\xi) \text{ est point fixe de } G & \Rightarrow & \\
\mu\xi.G(\xi) \subseteq \mu\xi.F(\xi) & \Rightarrow & \text{monotonie de } F \\
F(\mu\xi.G(\xi)) \subseteq F(\mu\xi.F(\xi)) & \Rightarrow & \\
F(\mu\xi.G(\xi)) \subseteq \mu\xi.F(\xi) & \Rightarrow & (t \notin \mu\xi.F(\xi)) \\
t \notin F(\mu\xi.G(\xi)) & &
\end{array}
$$

Supposons maintenant que $t \notin F(\mu\xi.G(\xi))$. Nous avons la suite d'implications suivante :

$$
\begin{array}{lcl}
t \notin F(\mu\xi.G(\xi)) & \Rightarrow & \\
F(\mu\xi.G(\xi)) \cap \complement\{t\} = F(\mu\xi.G(\xi)) & \Rightarrow & \text{(définition de } G\text{)} \\
G(\mu\xi.G(\xi)) = F(\mu\xi.G(\xi)) & \Rightarrow & \\
\mu\xi.G(\xi) = F(\mu\xi.G(\xi)) & \Rightarrow & \\
\mu\xi.G(\xi) \text{ est point fixe de } F & \Rightarrow & \\
\mu\xi.F(\xi) \subseteq \mu\xi.G(\xi) & \Rightarrow & \text{monotonie de } F \\
F(\mu\xi.F(\xi)) \subseteq F(\mu\xi.G(\xi)) & \Rightarrow & \\
\mu\xi.F(\xi) \subseteq F(\mu\xi.G(\xi)) & \Rightarrow & (t \notin F(\mu\xi.G(\xi))) \\
t \notin \mu\xi.F(\xi) & &
\end{array}
$$

□

Le lemme qui suit nous permet de montrer que l'utilisation de la substitution $\Gamma$ est correcte vis à vis de la satisfiabilité des formules à variables de récursion libres définie en termes d'une valuation $\delta$.

Rappelons que, d'après la proposition 8.11 (point (vii)), si $\phi_I$ est une formule sans variables de récursion libres et $\rho_I$ une valuation telles que $\phi_I$ est close par $\rho_I$ et si $\mathsf{check}(t, \phi, \rho, \Gamma)$ est un appel récursif généré par l'évaluation de l'algorithme $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$, alors la substitution $\Gamma$ est de la forme $[\xi_1 \to \mu\xi_1(M_1).\phi_1] \cdots [\xi_n \to \mu\xi_n(M_n).\phi_n]$. Dans le lemme qui suit, pour tout $i$ dans $1..n$, nous notons par $\Gamma_i$ le préfixe $[\xi_1 \to \mu\xi_1(M_1).\phi_1] \cdots [\xi_i \to \mu\xi_i(M_i).\phi_i]$ de $\Gamma$. Avec ses notations, $\Gamma_0$ est la substitution vide $\epsilon$.

**Lemme 8.16** *Soient $\phi_I$ une formule sans variables de récursion libres et $\rho_I$ une valuation telles que $\phi_I$ est close par $\rho_I$. Soit* $\mathsf{check}(t, \phi, \rho, \Gamma)$ *un appel récursif généré par l'évaluation de l'algorithme* $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$*, avec $\Gamma = [\xi_1 \to \mu\xi_1(M_1).\phi_1] \cdots [\xi_n \to \mu\xi_n(M_n).\phi_n]$.*

*Soient $i$ dans $1..n$ et $\delta$ une valuation des variables de récursion tels que la formule $\Gamma_i(\phi)$ est close par $\delta$. Alors les ensembles $[\![\Gamma_i(\phi)]\!]_{\rho,\delta}$ et $[\![\Gamma_{i-1}(\phi)]\!]_{\rho,\delta[\xi_i \to [\![\Gamma(\mu\xi_i(M_i).\phi_i)]\!]_\rho]}$ sont égaux.*

*Preuve* Pour alléger l'écriture, nous notons par $D_\rho$ l'ensemble d'arbres $[\![\Gamma(\mu\xi_i(M_i).\phi_i)]\!]_\rho$. La preuve se fait par récurrence sur la structure de la formule $\phi$.

Notons d'abord que, par la proposition 8.11 (point (ii)), $\Gamma$ est close par préfixe et donc pour tout $i$ dans $1..n$ et pour tout $j$ dans $i - 1..n$, on a $\Gamma_j(\mu\xi_i(M_i).\phi_i) = \Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)$ et $\Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)$ est une formule sans variables de récursion libres. Nous utilisons implicitement cette propriété.

Notons également que

($\star$) si la variable $\xi_i$ n'est pas libre dans $\Gamma_{i-1}(\phi)$, alors la propriété est vérifiée,

puisque dans ce cas d'une part $[\![\Gamma_{i-1}(\phi)]\!]_{\rho,\delta[\xi_i \to D]} = [\![\Gamma_{i-1}(\phi)]\!]_{\rho,\delta}$ et d'autre part $\Gamma_{i-1}(\phi) = \Gamma_i(\phi)$.

La preuve se fait par récurrence sur la structure de la formule $\phi$.

Cas de base: Si $\phi$ est l'une des formules $\mathbf{0}$, $\top$, $X$ ou $\eta = \eta'$, clairement ($\star$) s'applique et la propriété est vérifiée.

Si $\phi$ est la formule $\xi$, nous distinguons trois cas.

(i) Si $\xi \notin dom(\Gamma_i)$ alors clairement $\Gamma_{i-1}(\phi) = \xi$, donc ($\star$) s'applique et la propriété est vérifiée.

(ii) Si $\xi \in dom(\Gamma_{i-1})$, soit $\xi = \xi_j$ pour $j \leq i - 1$. Alors $\Gamma_{i-1}(\phi) = \Gamma_{j-1}(\mu\xi_j(M_j).\phi_j)$ qui est une formule sans variables de récursion libres. Donc ($\star$) s'applique et la propriété est vérifiée.

(iii) Si $\xi = \xi_i$, alors $\Gamma_i(\phi) = \Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)$ et $\Gamma_{i-1}(\phi) = \xi_i$. En utilisant la définition de la satisfiabilité, nous obtenons que $[\![\Gamma_{i-1}(\phi)]\!]_{\rho,\delta[\xi_i \to D_\rho]} = [\![\xi_i]\!]_{\rho,\delta[\xi_i \to D_\rho]} = D_\rho$. Alors il s'agit de montrer que $[\![\Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)]\!]_{\rho,\delta} = D_\rho$, ce qui est facile à voir rappelant que $D_\rho = [\![\Gamma(\mu\xi_i(M_i).\phi_i)]\!]_\rho$ et en utilisant le fait que $\Gamma(\mu\xi_i(M_i).\phi_i) = \Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)$ qui est une formule sans variables de récursion libres.

Cas d'induction: Si $\phi$ est la formule $\eta[\phi']$ alors, $\Gamma_j(\eta[\phi']) = \eta[\Gamma_j(\phi')]$ pour $j$ étant $i$ ou $i-1$. Il suffit donc de prouver que $[\![\eta[\Gamma_i(\phi')]]\!]_{\rho,\delta} = [\![\eta[\Gamma_{i-1}(\phi')]]\!]_{\rho,\delta[\xi\to D_\rho]}$. En utilisant la définition de l'interprétation des formules, ceci revient à montrer que $\rho(\eta)[[\![\Gamma_i(\phi')]\!]_{\rho,\delta}] = \rho(\eta)[[\![\Gamma_{i-1}(\phi')]\!]_{\rho,\delta[\xi\to D_\rho]}]$, ce qui est vrai par hypothèse de récurrence.

Si $\phi$ est l'une des formules $\phi' \mid \phi''$, $\neg\phi'$ ou $\phi' \vee \phi''$, comme dans le cas précédent, la preuve se fait simplement en utilisant la définition de l'interprétation des formules et l'hypothèse de récurrence.

Si $\phi$ est la formule $\mu\xi(M).\phi'$, on distingue différents cas en fonction de la variable $\xi$.

(i) Si $\xi \in dom(\Gamma_i)$, soit $\xi = \xi_j$ pour $j \leq i$, alors, par la proposition 8.11 (point (vii)), $\mu\xi_j(M'_j).\phi_j$ et sous formule de $\phi_I$ pour un certain ensemble $M'_j$ et par la proposition 8.11 (point (iii)), $\mu\xi(M).\phi'$ est sous formule de $\phi_I$. Donc, $\phi_j = \phi'$. D'autre part, la formule $\Gamma_{j-1}(\mu\xi_j(M_j).\phi_j)$ n'admet pas de variables de récursion libres, et donc c'est aussi le cas pour $\Gamma_{j-1}(\mu\xi(M).\phi')$. La conclusion est immédiate en rappelant que les formules $\Gamma_i(\mu\xi(M).\phi')$ et $\Gamma_{i-1}(\mu\xi(M).\phi')$ sont égales à $\Gamma_{j-1}(\mu\xi(M).\phi')$

(ii) Si $\xi \notin dom(\Gamma_i)$, alors $\Gamma_j(\mu\xi(M).\phi') = \mu\xi(M).\Gamma_j(\phi')$ pour $j$ étant $i$ ou $i-1$. Il suffit donc de montrer que

$$[\![\mu\xi(M).\Gamma_i(\phi')]\!]_{\rho,\delta} = [\![\mu\xi(M).\Gamma_{i-1}(\phi')]\!]_{\rho,\delta[\xi_i\to D_\rho]},$$

ce qui par définition de l'interprétation des formules revient à montrer que

$$\bigcap\{S \mid S \supseteq [\![\Gamma_i(\phi')]\!]_{\rho,\delta[\xi\to S]} \cap \complement M\} = \bigcap\{S \mid S \supseteq [\![\Gamma_{i-1}(\phi')]\!]_{\rho,\delta[\xi_i\to D_\rho][\xi\to S]} \cap \complement M\}.$$

Comme $\xi \neq \xi_i$, les valuations $\delta[\xi_i \to D_\rho][\xi \to S]$ et $\delta[\xi \to S][\xi_i \to D_\rho]$ sont équivalentes. Pour tout ensemble d'arbres $S$, nous pouvons appliquer l'hypothèse de récurrence sur la formule $\phi'$ et la valuation $\delta[\xi \to S]$, ce qui nous permet de déduire que pour tout ensemble $S$, les ensembles $[\![\Gamma_i(\phi')]\!]_{\rho,\delta[\xi\to S]}$ et $[\![\Gamma_{i-1}(\phi')]\!]_{\rho,\delta[\xi_i\to D_\rho][\xi\to S]}$ sont égaux. La conclusion est maintenant immédiate.

Si $\phi$ est la formule $\exists x.\phi'$, nous distinguons deux cas. Si la variable $\xi_i$ n'apparaît pas libre dans $\exists x.\phi'$, en utilisant que $\Gamma$ est close par préfixe qui est montré dans la proposition 8.11, nous déduisons que $\xi_i$ n'apparaît pas libre dans $\Gamma_{i-1}(\exists x.\phi')$ non plus, donc $(\star)$ s'applique et la propriété est vérifiée.

Regardons maintenant le cas où la variable $\xi_i$ apparaît libre dans $\exists x.\phi'$. Remarquons d'abord que dans ce cas $\exists x.\phi'$ est sous formule de $\phi_i$ puisque, par la proposition 8.11 (point (vii)), $\exists x.\phi'$ et $\mu\xi_i(M'_i).\phi_i$, pour un certain ensemble $M'_i$, sont des sous-formules de $\phi_I$. Maintenant, pour $j$ étant $i$ ou $i-1$, on a $\Gamma_j(\exists x.\phi') = \exists x.\Gamma_j(\phi')$. Donc il s'agit de prouver que

$$[\![\exists x.\Gamma_i(\phi')]\!]_{\rho,\delta} = [\![\exists x.\Gamma_{i-1}(\phi')]\!]_{\rho,\delta[\xi_i\to D_\rho]}.$$

Par définition de l'interprétation des formules, ceci revient de montrer que

$$\bigcup_{a\in\Sigma} [\![\Gamma_i(\phi')]\!]_{\rho[x\to a],\delta} = \bigcup_{a\in\Sigma} [\![\Gamma_{i-1}(\phi')]\!]_{\rho[x\to a],\delta[\xi_i\to D_\rho]}.$$

Par hypothèse de récurrence, nous savons que la partie gauche est égale à

$$\bigcup_{a\in\Sigma} [\![\Gamma_{i-1}(\phi')]\!]_{\rho[x\to a],\delta[\xi_i\to D_{\rho[x\to a]}]}.$$

Donc il suffit de montrer que $D_\rho = D_{\rho[x \to a]}$. Rappelons que par définitions de $D_\rho$ et $D_{\rho[x \to a]}$, ceci revient à montrer que

$$[\![\Gamma(\mu\xi_i(M_i).\phi_i)]\!]_\rho = [\![\Gamma(\mu\xi_i(M_i).\phi_i)]\!]_{\rho[x \to a]}.$$

Il suffit de montrer que la variable $x$ n'apparaît pas libre dans la formule $\Gamma(\mu\xi_i(M_i).\phi_i)$, et comme $\Gamma(\mu\xi_i(M_i).\phi_i) = \Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)$, de montrer que

($\star\star$) la variable $x$ n'apparaît pas libre dans la formule $\Gamma_{i-1}(\mu\xi_i(M_i).\phi_i)$.

Nous allons montrer pour toutes substitutions $\Gamma'$ et $\Gamma''$ telles que $\Gamma_{i-1} = \Gamma'\Gamma''$ que

(i) si la variable $\xi_j$ (pour $j$ dans $1..n$) est libre dans la formule $\Gamma''(\mu\xi_i(M_i).\phi_i)$, alors $\phi_i$ est sous-formule de $\phi_j$ ;

(ii) la variable $x$ n'est pas libre dans la formule $\Gamma''(\mu\xi_i(M_i).\phi_i)$.

Alors la propriété ($\star\star$) correspond au point (ii) ci-dessus pour $\Gamma' = \epsilon$.

Montrons d'abord le point (i). La preuve se fait par récurrence sur la longueur de $\Gamma''$. Pour le cas de base, $\Gamma'' = \epsilon$ et $\Gamma' = \Gamma_{i-1}$, donc la propriété est vérifiée par la proposition 8.11 (point (viii)).

Pour le pas d'induction, soit $\Gamma'' = [\xi_k \to \mu\xi_k(M_k).\phi_k]\Gamma'''$ pour un certain $k$ dans $1..i-1$ et une substitution $\Gamma'''$. Notons que nécessairement $j < k$, puisque dans le cas contraire $\xi_j$ ne serait pas libre dans $\Gamma''(\mu\xi_i(M_i).\phi_i)$ (ceci provient du fait que $\Gamma$ est close par préfixe). Nous distinguons deux cas.

- Si $\xi_j$ est libre dans $\Gamma'''(\mu\xi_i(M_i).\phi_i)$, alors par hypothèse de récurrence $\phi_i$ est sous formule de $\phi_j$.
- Si $\xi_j$ n'est pas libre dans $\Gamma'''(\mu\xi_i(M_i).\phi_i)$, alors (a) $\xi_j$ est libre dans $\mu\xi_k(M_k).\phi_k$ et (b) $\xi_k$ est libre dans $\Gamma'''(\mu\xi_i(M_i).\phi_i)$. Par hypothèse de récurrence, (b) implique que $\phi_i$ est sous formule de $\phi_k$. En utilisant la proposition 8.11, (a) implique que $\phi_k$ et sous formule de $\phi_j$. Alors, par transitivité de la relation de sous formule, $\phi_i$ est sous formule de $\phi_j$.

Montrons maintenant le point (ii). Encore une fois, la preuve se fait par récurrence sur la longueur de $\Gamma''$. Pour le cas de base, $\Gamma'' = \epsilon$ : il faut montrer que $x$ n'est pas libre dans $\mu\xi_i(M_i).\phi_i$, et donc que $x$ n'est pas libre dans $\phi_i$. C'est le cas puisque $\exists x.\phi'$ est sous formule de $\phi_i$ et donc toutes les occurrences de $x$ dans $\phi_i$ sont liées. Pour le pas d'induction, soit $\Gamma'' = [\xi_k \to \mu\xi_k(M_k).\phi_k]\Gamma'''$ pour un certain $k$ dans $1..i-2$ et une substitution $\Gamma'''$. Nous distinguons trois cas.

- La variable $\xi_k$ n'est pas libre dans la formule $\Gamma'''(\mu\xi_i(M_i).\phi_i)$. Alors $\Gamma''(\mu\xi_i(M_i).\phi_i) = \Gamma'''(\mu\xi_i(M_i).\phi_i)$ et la propriété est vérifiée par hypothèse de récurrence.
- La variable $\xi_k$ est libre dans $\Gamma'''(\mu\xi_i(M_i).\phi_i)$, mais la variable $x$ n'est pas libre dans $\phi_k$. En utilisant que $x$ n'est pas libre dans $\Gamma'''(\mu\xi_i(M_i).\phi_i)$, par hypothèse de récurrence, il est évident que $x$ n'est pas libre dans $[\xi_k \to \mu\xi_k(M_k).\phi_k]\Gamma'''(\mu\xi_i(M_i).\phi_i)$.
- La variable $\xi_k$ est libre dans $\Gamma'''(\mu\xi_i(M_i).\phi_i)$ et la variable $x$ est libre dans $\phi_k$. En utilisant le point (i), nous déduisons que $\phi_i$ est sous formule de $\phi_k$. D'autre part, comme $x$ est libre dans $\phi_k$ et que par la proposition 8.11 (points (iii) et (vii)) $\phi_k$ et $\exists x.\phi'$ sont sous formules de $\phi_I$, nous déduisons que $\phi_k$ est sous formule stricte de $\exists x.\phi'$. Comme de plus nous savons que $\exists x.\phi'$ est sous formule de $\phi_i$, nous avons obtenu une contradiction. Ceci nous garantit que ce troisième cas ne peut jamais se produire et termine la preuve du point (ii).

Pour terminer la preuve du lemme, il nous reste à considérer le cas où $\phi$ est la formule $\exists X.\phi$. La preuve de ce cas est identique à la preuve du cas précédent.

□

La proposition qui suit nous permettra de montrer que l'algorithme est correct pour le model checking des formules avec récursion.

**Proposition 8.17** *Soient $\phi_I$ une formule sans variables de récursions libres et $\rho_I$ une valuation telles que $\phi_I$ est close par $\rho_I$. Soit* $\mathsf{check}(t, \mu\xi(M).\phi, \rho, \Gamma)$ *un appel récursif généré par l'évaluation de l'algorithme pour* $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$. *Si $t$ n'est pas dans l'ensemble $M$, alors $t, \rho \models \Gamma(\mu\xi(M).\phi)$ si et seulement si $t, \rho \models \Gamma\langle\xi \to \mu\xi(M \cup \{t\}).\phi\rangle(\phi)$.*

*Preuve* Soient $\Gamma'$ un préfixe de $\Gamma$ tel que la variable $\xi$ n'appartient pas au domaine de $\Gamma'$ et pour tout ensemble $M'$, $\Gamma(\mu\xi(M').\phi) = \Gamma'(\mu\xi(M').\phi)$. Notons que le préfixe $\Gamma'$ existe toujours : si $\xi$ n'appartient pas à $dom(\Gamma)$, alors il suffirait de prendre $\Gamma' = \Gamma$. Si $\xi$ appartient à $dom(\Gamma)$, par la proposition (point (viii)), $\Gamma$ est de la forme $[\xi_1 \to \mu\xi_1(M_1).\phi_1] \cdots [\xi_k \to \mu\xi_k(M_k).\phi_k]$. Soit $i$ dans $1..n$ tel que $\xi = \xi_i$. Alors il suffit de prendre $\Gamma' = \Gamma_i$. Par définition de $\Gamma'$, $\xi$ est l'unique variable libre de $\Gamma'(\phi)$.

Nous notons par $F$ la fonction monotone de $\wp(\mathcal{A}_\Sigma)$ dans $\wp(\mathcal{A}_\Sigma)$ donnée par $F : S \mapsto [\![\Gamma'(\phi)]\!]_{\rho[\xi \to S]}$. Alors, par la définition de l'interprétation des formules, nous avons $\Gamma'(\mu\xi(M).\phi) = \mu S.F(S) \cap M$, où $\mu$ est utilisé ici comme le plus petit point fixe pour les fonctions monotones dans le treillis complet $\langle\wp(\mathcal{A}_\Sigma), \cup, \cap\rangle$.

La suite d'équivalences suivante est vérifiée :

$$
\begin{array}{lcl}
t, \rho \models \Gamma(\mu\xi(M).\phi) & \Leftrightarrow & \text{(par hypothèse sur } \Gamma') \\
t, \rho \models \Gamma'(\mu\xi(M).\phi) & \Leftrightarrow & \text{(par } \xi \notin dom(\Gamma') \\
t, \rho \models \mu\xi(M).\Gamma'(\phi) & \Leftrightarrow & \text{(par définitions de la satisfiabilité et de } F) \\
t \in \mu S.F(S) \cap \complement M & \Leftrightarrow & \text{(par le lemme 8.15 pour la fonction } S \mapsto F(S) \cap \complement M) \\
t \in F(\mu S.((F(S) \cap \complement M) \cap \complement\{t\}) \cap \complement M & \Leftrightarrow & \\
t \in F(\mu S.(F(S) \cap \complement(M \cup \{t\})) \cap \complement M & \Leftrightarrow & \text{(par } t \notin M) \\
t \in F(\mu S.(F(S) \cap \complement(M \cup \{t\}))) & \Leftrightarrow & \text{(par définition de } F) \\
t \in [\![\Gamma'(\phi)]\!]_{\rho,[\xi \to \mu S.(F(S) \cap \complement(M \cup \{t\}))]} & \Leftrightarrow & \text{(par définition de satisfiabilité et de } F) \\
t \in [\![\Gamma'(\phi)]\!]_{\rho,[\xi \to [\![\mu\xi(M \cup \{t\}).\Gamma'(\phi)]\!]_\rho]} & \Leftrightarrow & \text{(par } \xi \notin dom(\Gamma')) \\
t \in [\![\Gamma'(\phi)]\!]_{\rho,[\xi \to [\![\Gamma'(\mu\xi(M \cup \{t\}).\phi)]\!]_\rho]} & \Leftrightarrow & \text{(par hypothèse sur } \Gamma') \\
t \in [\![\Gamma'(\phi)]\!]_{\rho,[\xi \to [\![\Gamma(\mu\xi(M \cup \{t\}).\phi)]\!]_\rho]} & &
\end{array}
$$

En utilisant le lemme 8.16 pour $\Gamma\langle\xi \to \mu\xi(M \cup \{t\})\rangle$ et pour $\delta$ étant la valuation vide, $t \in [\![\Gamma'(\phi)]\!]_{\rho,\delta[\xi \to [\![\Gamma(\mu\xi(M \cup \{t\}).\phi)]\!]_\rho]}$ est équivalent à $t \in [\![\Gamma'[\xi \to \mu\xi(M \cup \{t\}).\phi](\phi)]\!]_\rho$. Par définition de $\Gamma'$ et par propriété de la satisfiabilité, ceci est équivalent à $t, \rho \models \Gamma\langle\xi \to \mu\xi(M \cup \{t\})\rangle(\phi)$.

□

*Preuve du théorème 8.14* Nous montrons que pour tout appel récursif $\mathsf{check}(A, \phi, \rho, \Gamma)$ généré par l'évaluation de $\mathsf{check}(A_I, \phi_I, \rho_I, \epsilon)$, $\mathsf{check}(t, \phi, \rho, \Gamma)$ retourne vrai si et seulement si $t, \rho \models \Gamma(\phi)$. La preuve se fait par récurrence sur l'arbre d'exécution de l'algorithme.

Cas de base: La formule $\phi$ est une parmi $\top$, $\mathbf{0}$, $X$ ou $\eta = \eta'$. Dans tous ces cas, $\Gamma(\phi) = \phi$ et la preuve est immédiate par définition de la satisfiabilité.

Cas d'induction: Si $\phi$ est l'une des formules $\eta[\phi']$, $\phi' \mid \phi''$, $\neg\phi'$ ou $\phi' \vee \phi''$, la preuve est facile en utilisant la définition de la satisfiabilité, l'hypothèse de récurrence et le fait que $\Gamma(\eta[\phi']) = \eta[\Gamma(\phi')]$, $\Gamma(\phi' \mid \phi'') = \Gamma(\phi') \mid \Gamma(\phi'')$, $\Gamma(\neg\phi) = \neg\Gamma(\phi)$ et $\Gamma(\phi' \vee \phi'') = \Gamma(\phi') \vee \Gamma(\phi'')$.

Si $\phi$ est la formule $\exists x.\phi'$, l'algorithme $\mathsf{check}(t, \exists x.\phi', \rho, \Gamma)$ retourne vrai si et seulement si

(i) il existe une étiquette $a$ n'appartenant ni à $etiq(t)$ ni à $etiq(\phi', \rho, \Gamma)$ et

(ii) il existe une étiquette $b$ dans $etiq(t) \cup etiq(\phi', \rho, \Gamma) \cup \{a\}$ pour laquelle $\mathsf{check}(t, \phi', \rho[x \rightarrow b], \Gamma)$ retourne vrai.

Remarquons d'abord que l'étiquette $a$ existe puisque les ensembles $etiq(t)$ et $etiq(\phi', \rho, \Gamma)$ sont finis et nous considérons un ensemble infini d'étiquettes. [2] D'autre part, par hypothèse de récurrence, $\mathsf{check}(t, \phi', \rho[x \rightarrow b], \Gamma)$ retourne vrai si et seulement si

(iii) $t, \rho[x \rightarrow b] \models \Gamma(\phi')$

En remarquant que, par définitions de $etiq(\phi, \rho, \Gamma)$ et $\Gamma(\phi)$, $etiq(\Gamma(\phi), \rho) \subseteq etiq(\phi, \rho, \Gamma)$, en utilisant la proposition 8.9 nous savons que (i),(ii) et (iii) si et seulement si $t, \rho \models \exists x.\Gamma(\phi')$. La conclusion suit immédiatement de $\Gamma(\exists x.\phi') = \exists x.\Gamma(\phi')$.

Si $\phi$ est la formule $\exists X.\phi'$, la preuve est similaire au cas précédent en utilisant la proposition 8.7.

Si $\phi$ est la formule $\mu\xi(M).\phi'$, par définition de l'algorithme, $\mathsf{check}(t, \mu\xi(M).\phi', \rho, \Gamma)$ retourne vrai si et seulement si $t$ n'appartient pas à $M$ et $\mathsf{check}(t, \phi', \rho, \Gamma\langle\xi \rightarrow \mu\xi.(M \cup \{t\}).\phi'\rangle)$ retourne vrai. Par la proposition 8.11 (point (iii)) nous savons que $\mu\xi(M).\phi'$ est sous formule de $\phi_I$ et donc $M = \varnothing$. Par conséquent, $t$ n'appartient pas à $M$. Par hypothèse de récurrence, $\mathsf{check}(t, \phi', \rho, \Gamma\langle\xi \rightarrow \mu\xi.(M \cup \{t\}).\phi'\rangle)$ retourne vrai si et seulement si $t, \rho \models \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle(\phi')$. Par la proposition 8.17, $t, \rho \models \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle(\phi')$ si et seulement si $t, \rho \models \Gamma(\mu\xi(M).\phi')$, ce qu'il fallait démontrer.

Si $\phi$ est la formule $\xi$, soit $[\xi \rightarrow \mu\xi(M).\phi']$ élément de $\Gamma$ (nous savons par la proposition 8.11 (point (v)) que $\xi$ est dans le domaine de $\Gamma$). Alors $\Gamma(\xi) = \Gamma(\mu\xi(M).\phi')$. Maintenant, par définition de l'algorithme, $\mathsf{check}(t, \xi, \rho, \Gamma)$ retourne vrai si et seulement si $t$ n'est pas dans $M$ et $\mathsf{check}(t, \phi', \rho, \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle)$ retourne vrai, ce qui par hypothèse de récurrence est équivalent à

(i) $t \notin M$ et $t, \rho \models \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle(\phi')$.

En supposant (i), par la proposition 8.17 nous avons que $t, \rho \models \Gamma(\mu\xi(M).\phi')$, ce qui est équivalent à $t, \rho \models \Gamma(\xi)$. En supposant $t, \rho \models \Gamma(\xi)$, càd $t, \rho \models \Gamma(\mu\xi(M).\phi')$, par définition de la satisfiabilité $t \notin M$, et donc par la même proposition que précédemment $t, \rho \models \Gamma\langle\xi \rightarrow \mu\xi(M \cup \{t\}).\phi'\rangle(\phi')$.

□

## 8.5 Complexité de l'algorithme

Nous montrons ici que l'algorithme de model checking présenté dans ce chapitre s'exécute en espace polynomial dans la taille du problème de model checking. Nous commençons par

[2] Dans une configuration où $\Sigma$ serait un ensemble fini, l'étiquette $a$ n'existerait pas forcément. Cependant, dans ce cas, le problème de la définition de $[\![]\!]_{\rho,\delta}\exists x.\phi]$ comme une union infinie ne se poserait plus et nous n'aurions plus besoin des propositions 8.9 et 8.7.

définir ce qu'est la taille d'un problème de model checking, ensuite nous présentons une représentation d'un problème de model checking qui est polynomiale dans la taille du problème et finalement nous montrons que l'évaluation de l'algorithme requiert un espace polynomial dans la taille du problème.

### 8.5.1 Taille d'un problème de model checking

La *taille* d'une formule logique $\phi$, notée $taille(\phi)$, est définie récursivement sur la structure de $\phi$ comme suit :

- $taille(\mathbf{0}) = taille(\top) = taille(X) = taille(\xi) = taille(\eta = \eta') = 1$ ;
- $taille(x[\phi]) = taille(\neg\phi) = taille(\exists x.\phi) = taille(\exists X.\phi) = taille(\phi) + 1$ ;
- $taille(\phi \mid \phi') = taille(\phi \vee \phi') = taille(\phi) + taille(\phi') + 1$ ;
- $taille(\mu\xi(M).\phi) = taille(\phi) + taille(M) + 1$.

Les tailles respectives d'une valuation $\rho$ et d'une substitution $\Gamma$ sont définies par (pour tout $a$ dans $\Sigma$, $taille(a) = 1$) :

- $taille(\rho) = \sum_{\chi \in dom(\rho)} taille(\rho(\chi))$ (pour $\chi$ étant une variable d'étiquette ou une variable d'arbre) ;
- $taille(\Gamma) = \sum_{[\xi \to \phi] \text{ dans } \Gamma} taille(\phi)$.

Pour un arbre $t$, une valuation $\rho$ et une formule $\phi$ qui est close par $\rho$, la taille du problème de model checking « est-ce que $t$ satisfait $\phi$ sous la valuation $\rho$ » est $taille(t) + taille(\phi) + taille(\rho)$.

### 8.5.2 Complexité en espace de l'algorithme

Cette section est consacrée à la preuve de la proposition suivante :

**Proposition 8.18** *Soient $t_I$ un arbre et $\phi_I$ une formule et $\rho_I$ une valuation telle que $\phi_I$ est close par $\rho_I$. L'algorithme* $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$ *requiert pour son évaluation un espace polynomial dans $taille(t_I) + taille(\phi_I) + taille(\rho_I)$.*

Soit $AE_I$ l'arbre d'évaluation de l'algorithme $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$ : les nœuds de cet arbre sont des instances du problème de model checking de la forme $\mathsf{check}(t, \phi, \rho, \Gamma)$ et le branchement est défini par les appels récursifs. Soit $\mathbf{T}_I$ la quantité $taille(t_I) + taille(\phi_I) + taille(\rho_I)$. La preuve de la proposition passe par trois étapes : nous montrons d'abord que la longueur de chaque branche de $AE_I$ est bornée de façon polynomiale par $\mathbf{T}_I$. Nous montrons ensuite que la représentation de chacun des nœuds de cet arbre est polynomiale dans $\mathbf{T}_I$ et finalement nous montrons que l'évaluation de l'algorithme peut être effectuée en stockant à tout moment une seule branche de l'arbre $AE_I$ en mémoire.

**Lemme 8.19** *La longueur des branches de l'arbre d'exécution est bornée par $taille(\phi_I)^3 * taille(t_I)$.*

*Preuve* Nous utilisons la relation d'ordre qui a été définie pour montrer la terminaison de l'algorithme (section 8.4.2). Soit $varrec(\phi_I)$ l'ensemble à $n$ éléments $\{\xi_1, \ldots, \xi_n\}$. Pour chaque instance du problème de model checking $\mathsf{check}(t, \phi, \rho, \Gamma)$ générée par l'évaluation de $\mathsf{check}(t_I, \phi_I, \rho_I, \Gamma_I)$,

soit le triplet $\mathcal{M}(t, \phi, \rho, \Gamma) \subseteq \mathbb{N}^3$ défini par :

$$\mathcal{M}(t, \phi, \rho, \Gamma) = (n - \mathsf{Card}(dom(\Gamma)), \sum_{i \in 1..n} taille\_minarbre(\Gamma, \xi_i), taille(\phi))$$

où pour tout $i$ dans $1..n$, $taille\_minarbre(\Gamma, \xi_i)$ est donné par :

(i) si $\xi_i$ appartient à $dom(\Gamma)$ et $[\xi_i \rightarrow \mu\xi_i(M).\phi]$ est élément de $\Gamma$, et soit $t$ le plus petit arbre, pour la relation de composant, contenu dans l'ensemble $M$, alors $taille\_minarbre(\Gamma, \xi_i) = taille(t)$ ;

(ii) si $\xi_i$ n'appartient pas à $dom(\Gamma)$, alors $taille\_minarbre(\Gamma, \xi_i) = taille(t_I)$.

Il est facile de voir que si $\mathsf{check}(t, \phi, \rho, \Gamma) <_{t_I} \mathsf{check}(t', \phi', \rho', \Gamma')$, alors $\mathcal{M}(t, \phi, \rho, \Gamma) < \mathcal{M}(t', \phi', \rho', \Gamma')$, où $<$ désigne la relation d'ordre par composante sur les vecteurs. D'autre part, $\mathcal{M}(t_I, \phi_I, \rho_I, \epsilon) = (n, n * taille(t_I), taille(\phi_I)))$. La conclusion est immédiate en remarquant que $n \leq taille(\phi_I)$.

□

L'instance $\mathsf{check}(t, \phi, \rho, \Gamma)$ du problème de model checking est représentée par ses arguments. Un arbre est représenté par l'ensemble de ses nœuds et des pointeurs étiquetées entre les nœuds définissant la structure arborescente et l'étiquetage des arêtes de l'arbre. Une formule est représentée également par une structure arborescente dont les nœuds sont étiquetées par les opérateurs logiques. Les nœuds de récursion (opérateur $\mu\xi$) se voient associer également une liste d'arbres. Une substitution $\Gamma$ est représentée par une liste de couples de la forme <variable de récursion, formule>. Finalement, une valuation $\rho$ est représentée par une liste de couples de la forme <variable d'arbre, arbre> ou <variable d'étiquette, étiquette>.

**Lemme 8.20** *La représentation des nœuds de l'arbre d'exécution est polynomiale en* $\mathbf{T}_I$.

*Preuve* Pour chaque instance du problème de model checking $\mathsf{check}(t, \phi, \rho, \Gamma)$ dans l'arbre d'évaluation $AE_I$, les tailles de $t$, $\phi$, $\rho$ et $\Gamma$ sont polynomiales dans $T_I$ pour les raisons suivantes : d'après la proposition 8.11 (point (iv)), $t$ est composant de $t_I$ et $\phi$ est sous formule de $\phi_I$. D'après cette même proposition, la taille de $\rho$ est bornée par $taille(\phi_I) + taille(\rho_I)$ et la longueur de $\Gamma$ est bornée par la taille de $\phi_I$ et pour chaque $[\xi \rightarrow \mu\xi(M).\phi']$ élément de $\Gamma$, $\phi'$ est sous formule de $\phi_I$ et la taille de $M$ est bornée par $taille(t_I)^2$. En ce qui est de la représentation effective des instances du problème, il est clair qu'elle est polynomiale dans $\mathbf{T}_I$ pour $t$, $\phi$ et $\Gamma$, puisque le nombre d'opérateurs logiques est constant et le nombre d'étiquettes utilisées dans $t$, $\phi$ et $\Gamma$ est sous ensemble des étiquettes utilisées dans $t_I$ et $\phi_I$. La représentation polynomiale de $\rho$ est moins immédiate à voir, puisque la valuation $\rho$ peut associer à certaines variables des étiquettes qui n'apparaissent pas dans $t_I$ ni dans $\phi_I$ et des arbres qui ne sont pas composants de $t_I$. Notons que la taille de ces arbres est constante : ils sont toujours de la forme $a[\{\!\}]$, où $a$ est une étiquette n'apparaissant ni dans $t_I$, ni dans $\phi_I$ ni dans $\rho_I$. Il s'agit donc de montrer que l'évaluation de l'algorithme nécessite un nombre polynomial d'étiquettes nouvelles (càd n'apparaissant ni dans $t_I$, ni dans $\phi_I$). Ceci est vérifié puisque la taille de $\rho$ et bornée par $taille(\rho_I) * taille(\phi_I)$ et donc au plus $taille(\rho_I) * taille(\phi_I)$ nouvelles étiquettes sont nécessaires pour l'évaluation de l'algorithme. D'après les évaluations qui ont été faites, nous pouvons conclure que la taille de la représentation d'un nœud de l'arbre d'exécution est limitée par $taille(\phi_I) + taille(\rho_I) + (taille(\phi_I) + taille(\rho_I))^2 + (taille(\phi_I) \times taille(\rho_I))^2$.

□

Pour terminer la preuve de la proposition 8.18, nous allons montrer que l'évaluation de $\mathsf{check}(t_I, \phi_I, \rho_I, \epsilon)$ peut être effectuée en stockant à tout moment une seule branche de l'arbre d'évaluation, plus éventuellement, pour chaque nœud, une information supplémentaire permettant d'énumérer tous les nœuds fils du nœud courant non encore évalués.

**Lemme 8.21** *Pour tout nœud* $\mathsf{check}(t, \phi, \rho, \Gamma)$ *de l'arbre d'exécution, on peut énumérer les appels récursifs directs* $\mathsf{check}(t', \phi', \rho', \Gamma')$ *générés par l'évaluation de* $\mathsf{check}(t, \phi, \rho, \Gamma)$ *en utilisant un espace polynomial dans* $\mathbf{T}_I$.

*Preuve* La preuve se fait en étudiant dans la structure de l'algorithme tous les cas qui génèrent des appels récursifs. Dans la majorité des cas, un ou deux appels récursifs sont générés, et donc il suffit de générer tous les appels récursifs (càd construire et garder en mémoire les instances du problème de la forme $(t', \phi', \rho', \gamma')$ correspondant aux appels récursifs). Les seuls cas non triviaux sont pour $\phi$ étant de la forme $\phi' \mid \phi''$ ou $\exists X.\phi'$. Il nous suffit de montrer dans ces cas que la boucle *pour* peut s'exécuter effectivement de façon itérative, en énumérant tous les cas, et que l'appel récursif de chaque itération peut être calculé en utilisant uniquement l'appel courant et l'appel de l'itération précédente. Pour $\phi$ étant la formule $\phi' \mid \phi''$, il s'agit d'énumérer tous les couples d'arbres $t', t''$ tels que $t' \uplus t'' = t$. Ceci revient à énumérer toutes les bipartitions de l'ensemble des nœuds successeurs du nœud racine de l'arbre $t$ et peut donc être fait en utilisant un espace polynomial dans $taille(t)$. Pour $\phi$ étant la formule $\exists X.\phi'$, il s'agit d'énumérer tous les composants de l'arbre $t$. Remarquons qu'un composant de $t$ peut être identifié par un sous ensemble des nœuds de $t$. Tous les sous ensembles de nœuds de $t$ ne définissent pas un composant de $t$ correct, néanmoins étant donné un sous ensemble $S$ de l'ensemble des nœuds de $t$, on peut vérifier s'il correspond effectivement à un composant de $t$. Ceci se fait en

(i) annotant chaque élément $n$ de $S$ par un booléen indiquant si tous les nœuds successeurs de $n$ dans $t$ sont également dans $S$ et

(ii) vérifiant que tous les nœuds $n$ dans $S$ qui ne sont pas nœuds successeurs d'un nœud dans $S$ ont le même nœud père dans $t$.

Il est facile de voir que les étapes (i) et (ii) nécessitent un espace supplémentaire qui est linéaire dans la représentation de $t$. De ce qui a été dit nous déduisons que sur chaque nœud de la branche d'évaluation courante de l'arbre d'évaluation, la taille des informations qu'il nous faut stocker pour permettre la continuation de l'évaluation est au pire de taille d'un nœud de l'arbre d'exécution.

□

Remarquons finalement que, d'après les bornes de complexité qui ont été donné dans les différents lemmes intermédiaires, le polynôme qui borne la complexité en espace de l'algorithme de model checking a un degré 5 pour $taille(\phi_I)$, un degré 3 pour $taille(t_I)$ et un degré 2 pour $taille(\rho_I)$.

# Conclusion

Dans ce chapitre nous avons étudié la complexité du model checking pour la logique spatiale. Nous avons identifié deux difficultés liées au model checking. La première provient de la

quantification dans la logique dont l'interprétation demande de considérer un ensemble potentiellement infini d'objets (arbres ou variables), ce qui empêche la définition d'un algorithme de model checking en utilisant directement la définition de la satisfiabilité des formules logiques. Nous avons montré que du point de vue du problème de model checking, cette quantification peut être remplacée par une quantification sur un domaine fini. La deuxième difficulté est liée à l'opérateur de point fixe qui, considéré de façon naïve peut résulter en un accroissement exponentiel de la taille de la formule. Pour empêcher cette explosion en espace nous avons montré qu'il est possible d'adapter une méthode de model checking du $\mu$-calcul. Ces deux résultats nous ont permis de définir un algorithme de model checking de la logique spatiale qui s'exécute en espace polynomial dans la taille de la formule et de l'arbre. Comme nous allons le voir dans le chapitre suivant, ce résultat est optimal, puisque le problème de model checking de la logique spatiale est PSPACE complet.

# Chapitre 9

# Bornes inférieures pour la complexité du model checking

Dans le présent chapitre nous établissons quelques bornes de la complexité du model checking pour différents fragments de la logique. Comme nous l'avons vu dans le chapitre précédent, les sources de difficulté et de complexité pour le model checking sont la composition, la quantification et la récursion. La composition faisant parti intégrante de la logique spatiale (sans quoi ce ne serait plus une logique spatiale), nous étudions ici ce qui se passe si les deux autres sources de complexité – la récursion et la quantification – sont enlevées.

Nous commençons le chapitre par une section de rappels et définitions d'objets et notions qui seront utilisés par la suite. Nous étudions ensuite (dans la section 9.2) la complexité combinée de deux petits fragments de la logique spatiale, et nous établissons qu'elle est PSPACE-difficile pour tous ces fragments ; il s'agit du fragment mini-LS, déjà introduit et obtenu en enlevant de la logique la récursion et la quantification, et du fragment constitué uniquement des opérateurs booléens et de la quantification. La section 9.3 est consacrée à l'étude de la complexité de données de la logique spatiale. Nous exploitons d'abord l'équivalence entre fragments de la logique et les logiques MSO et PMSO, et les automates à contraintes numériques, pour déterminer la complexité de données de ces fragments. Ensuite nous établissons que la complexité de données est PSPACE difficile pour la logique spatiale sans restriction et difficile pour tous les niveaux de la hiérarchie polynomiale pour la logique sans récursion.

## 9.1 Préliminaires

Commençons par rappeler les définitions des deux fragments de la logique qui vont nous intéresser dans ce chapitre. La logique $\mathrm{LS}_{|\mu}$ est obtenu en enlevant l'opérateur de point fixe de la logique spatiale, et la logique mini-LS est obtenue en enlevant le point fixe et la quantification. Concrètement, cela nous donne les deux fragments syntaxiques suivants :

$$(\mathrm{LS}_{|\mu}) \quad \phi ::= \mathbf{0} \mid \eta[\phi] \mid \phi|\phi \mid \top \mid \neg\phi \mid \phi \vee \phi \mid X \mid \exists X.\phi \mid \exists x.\phi \mid \eta = \eta$$
$$(\text{mini-LS}) \quad \phi ::= \mathbf{0} \mid \eta[\phi] \mid \phi|\phi \mid \top \mid \neg\phi \mid \phi \vee \phi.$$

Nous continuons par le rappel des définitions de quelques classes de complexité et par la définition des formules booléennes quantifiées utilisées dans ce chapitre

### 9.1.1 Quelques classes de complexité

Nous rappelons brièvement quelques classes de complexité. Nous supposons que le lecteur connaît la notion de complétude pour une classe de complexité. Pour une définition plus détaillée, voir par exemple [Papadimitriou, 1995].

Soient $\mathcal{C}, \mathcal{D}$ deux classes de complexité en temps, déterministes ou non. On définit $\mathcal{C}^{\mathcal{D}}$ comme la classe de complexité des problèmes pouvant être résolus par un algorithme de la classe $\mathcal{C}$ qui de plus peut faire des requêtes auprès d'un oracle pouvant répondre à tout problème de la classe $\mathcal{D}$ (la réponse à ces requêtes est donnée en un temps constant).

Soient comme d'habitude P la classe des algorithmes déterministes polynomiaux en temps, NP la classe des algorithmes non déterministes polynomiaux en temps et coNP la classe des algorithmes dont le complémentaire est dans NP. La *hiérarchie polynomiale* est la suite de classes de complexité suivante :

- $\Omega_0 = \Sigma_0 = \Pi_0 = \mathrm{P}$ ;
- $\Omega_{i+1} = \mathrm{P}^{\Sigma_i}, \quad \Sigma_{i+1} = \mathrm{NP}^{\Sigma_i}, \quad \Pi_{i+1} = \mathrm{coNP}^{\Sigma_i}$.

Les suites $\Sigma_i$ et $\Pi_i$, pour $i$ dans $\mathbb{N}$, sont également appelées hiérarchie polynomiale existentielle et hiérarchie polynomiale universelle respectivement. La hiérarchie polynomiale cumulative, notée PH, est la classe $\bigcup_{i\in\mathbb{N}} \Sigma_i$.

Nous notons PSPACE la classe des algorithmes polynomiaux en espace.

Les inclusions suivantes sont connues pour ces différentes classes de complexité :

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE}$$

$$\text{pour tout } i \text{ dans } \mathbb{N}, \quad \Omega_i \subseteq \Omega_{i+1}, \quad \Sigma_i \subseteq \Sigma_{i+1}, \quad \Pi_i \subseteq \Pi_{i+1}.$$

### 9.1.2 Formules booléennes quantifiées

Soit $\mathcal{U}$ un ensemble dénombrable de variables. Une formule booléenne quantifiée (FBQ) est une formule de la logique propositionnelle de la forme $Q_1 u_1 . \cdots . Q_n u_n . G$ où chacun des $Q_i$ est un quantificateur $\exists$ ou $\forall$ et $G$ est une formule de la logique propositionnelle construite sur les variables $\{u_1, \ldots, u_n\} \subseteq \mathcal{U}$.

Le degré d'alternance d'une FBQ $\mathcal{G}$, noté $degalt(\mathcal{G})$, est défini récursivement sur la structure de la formule comme suit (pour $Q, Q'$ étant $\forall$ ou $\exists$ et $u$ et $u'$ étant des variables propositionnelles) :

- si $G$ est une formule sans quantification, alors $degalt(G) = 0$ et $degalt(Qu.G) = 1$ ;
- $degalt(Qu.Q'u'.G) = degalt(Q'u'.G)$ si $Q = Q'$ ;
- $degalt(Qu.Q'u'.G) = 1 + degalt(Q'u'.G)$ si $Q \neq Q'$.

Trois différents problèmes liés à la validité des FBQ sont connus pour être complets pour les classes PSPACE, $\Sigma_n$ et $\Pi_n$ pour tout naturel $n$, comme énoncé dans les deux théorèmes qui suivent.

**Théorème 9.1 ([Stockmeyer, 1976])** *Le problème de validité pour une formule booléenne quantifiée est PSPACE complet.*

**Théorème 9.2 ([Stockmeyer, 1976])** *Le problème de validité pour une formule booléenne quantifiée $\mathcal{G}$ de degré d'alternance $n$ est :*

- *complet pour le $n^{ème}$ niveau de la hiérarchie polynomiale existentielle si $\mathcal{G}$ est de la forme $\exists u.\mathcal{G}'$ ;*
- *complet pour le $n^{ème}$ niveau de la hiérarchie polynomiale universelle si $\mathcal{G}$ est de la forme $\forall u.\mathcal{G}'$.*

**Remarque 9.3** *Les théorèmes 9.1 et 9.2 restent valides pour toute formule booléenne quantifiée en forme prénexe normale disjonctive.*

En utilisant la remarque précédente, dans la suite nous considérons, sans perte de généralité, des formules booléennes quantifiée en forme prénexe normale disjonctive. C'est à dire, une FBQ est une formule **close** de la forme $Q_1 u_1 . \cdots . Q_n u_n . G$, où les $Q_i$ sont des quantificateurs et $G$ est définie par la syntaxe :

$$\begin{array}{ll} G & ::= \theta_1 \vee \cdots \vee \theta_k \\ \theta & ::= l_1 \wedge \cdots \wedge l_r \\ l & ::= u_i \mid \overline{u_i}, \quad i \in \{1, \ldots, n\}. \end{array}$$

Une *valuation des variables propositionnelles* $\gamma$ de domaine $\mathcal{U}' \subseteq \mathcal{U}$ est une application de $\mathcal{U}'$ dans $\{\mathsf{vrai}, \mathsf{faux}\}$. Nous considérons dans la suite uniquement des valuations de domaine fini. Parfois, la notation $u_1 \mapsto \mathsf{b}_1, \cdots, u_n \mapsto \mathsf{b}_n$ sera utilisée pour désigner la valuation $\gamma$ de domaine $\{u_1, \ldots, u_n\}$ qui à $u_i$ associe la valeur booléenne $\mathsf{b}_i$, pour $i$ dans $1..n$.

Soit $\mathcal{G}$ une formule booléenne quantifiée et $\mathcal{G}'$ une sous formule de $\mathcal{G}$ dont les variables libres sont incluses dans $\{u_1, \ldots, u_m\}$, et soit $\gamma$ une valuation de domaine $\{u_1, \ldots, u_m\}$. Alors la valuation $\gamma$ satisfait la formule $\mathcal{G}'$, écrit $\gamma \models \mathcal{G}'$, si :

- $\mathcal{G}' = u$ et $\gamma(u) = \mathsf{vrai}$ ;
- $\mathcal{G}' = \overline{u}$ et $\gamma(u) = \mathsf{faux}$ ;
- $\mathcal{G}' = l_1 \wedge \cdots \wedge l_k$ et pour tout $i$ dans $1..k$, $\gamma \models l_i$ ;
- $\mathcal{G}' = \theta_1 \vee \cdots \vee \theta_k$ et il existe $i$ dans $1..k$ tel que $\gamma \models \theta_i$ ;
- $\mathcal{G}' = \exists u.\mathcal{G}''$ et il existe b dans $\{\mathsf{vrai}, \mathsf{faux}\}$ tel que $\gamma, u \mapsto \mathsf{b} \models \mathcal{G}''$ ;
- $\mathcal{G}' = \forall u.\mathcal{G}''$ et pour tout b dans $\{\mathsf{vrai}, \mathsf{faux}\}$, $\gamma, u \mapsto \mathsf{b} \models \mathcal{G}''$.

Il est évident que la formule booléenne quantifiée $\mathcal{G}$ est valide si et seulement si $\gamma \models \mathcal{G}$ pour toute valuation $\gamma$ (et en particulier pour la valuation $\gamma$ vide).

## 9.2 Borne inférieure pour la complexité combinée

Rappelons que la complexité combinée du problème de model checking $t, \rho \models \phi$ est la complexité du problème en considérant l'arbre $t$, la valuation $\rho$ et la formule $\phi$ comme données du problème. Nous commençons par énoncer deux résultats sur la complexité combinée du model checking pour la logique limitée aux opérateurs booléens (dans la section 9.2.1). Plus précisément, la complexité combinée est PSPACE-difficile pour ce fragment, et le résultat peut être obtenu en utilisant la quantification sur les variables d'étiquettes seule et en utilisant la

quantification sur les variables d'arbres seule.Dans la section 9.2.2 nous étudions la complexité combinée de la logique mini-LS, et nous établissons que celle ci est PSPACE-difficile.

### 9.2.1 La logique sans opérateurs spatiaux

Nous présentons ici deux résultats montrant qu'en présence de chacun des deux types de quantification de la logique (quantification sur les étiquettes et quantification sur les arbres), le model checking est PSPACE complet pour sa complexité combinée. Soient $LS_{\mathrm{Bool}x}$ et $LS_{\mathrm{Bool}X}$ les fragments de la logique suivants :

$$\begin{array}{ll} (LS_{\mathrm{Bool}x}) & \phi ::= \eta = \eta' \mid \neg\phi \mid \phi \vee \phi \mid \exists x.\phi \\ (LS_{\mathrm{Bool}X}) & \phi ::= X \mid \neg\phi \mid \phi \vee \phi \mid \exists X.\phi \end{array}$$

L'intérêt de ses deux fragments est assez limité puisque pour toute formule close $\phi$ de l'un de ses fragments, l'ensemble $[\![\phi]\!]$ est ou bien l'ensemble de tous les arbres, ou bien l'ensemble vide. Néanmoins, les résultats de complexité que nous proposons montrent que les deux types de quantification dans la logique spatiale permettent d'encoder la quantification de la logique propositionnelle classique.

#### Quantification sur les étiquettes

Nous commençons par la logique $LS_{\mathrm{Bool}x}$, en nous contentant de rappeler comment la borne PSPACE difficile pour la complexité combinée de cette logique a été établie dans [Charatonik et al., 2001, Charatonik et al., 2003]. Soit $\mathcal{G} = Q_1 u_1 . \cdots . Q_n u_n . G$ une formule booléenne quantifiée, où $G$ est une formule sans quantification en forme normale disjonctive. A $\mathcal{G}$ on associe la formule $(\!|\mathcal{G}|\!)$ de la logique $LS_{\mathrm{Bool}x}$ définie récursivement sur la structure de $\mathcal{G}$ par :

$$\begin{array}{lcl} (\!|u_i|\!) & = & \mathsf{u}_i = \mathsf{vv} \\ (\!|\overline{u_i}|\!) & = & \mathsf{u}_i = \mathsf{ff} \\ (\!|l_1 \wedge \cdots \wedge l_k|\!) & = & (\!|l_1|\!) \wedge \cdots \wedge (\!|l_k|\!) \\ (\!|\theta_1 \vee \cdots \vee \theta_k|\!) & = & (\!|\theta_1|\!) \vee \cdots \vee (\!|\theta_k|\!) \\ (\!|\exists u_i.\mathcal{G}|\!) & = & \exists \mathsf{u}_i.(\mathsf{u}_i = \mathsf{vv} \vee \mathsf{u}_i = \mathsf{ff}) \wedge (\!|\mathcal{G}|\!) \\ (\!|\forall u_i.\mathcal{G}|\!) & = & \forall \mathsf{u}_i.(\mathsf{u}_i = \mathsf{vv} \vee \mathsf{u}_i = \mathsf{ff}) \rightarrow (\!|\mathcal{G}|\!) \end{array}$$

où vv et ff sont des étiquettes dans $\Sigma$.

**Lemme 9.4 ([Charatonik et al., 2001, Charatonik et al., 2003])** *Pour toute formule booléenne quantifiée $\mathcal{G}$, $\mathcal{G}$ est valide si et seulement si $\{\!|\}$ satisfait $(\!|\mathcal{G}|\!)$.*

**Théorème 9.5 ([Charatonik et al., 2001, Charatonik et al., 2003])** *Le problème de model checking de la logique $LS_{Boolx}$ est PSPACE complet pour sa complexité combinée.*

A proprement parler, dans [Charatonik et al., 2001, Charatonik et al., 2003], les auteurs considèrent des formules booléennes quantifiées en forme prénexe normale conjonctive. Par souci d'uniformité, nous avons présenté ici une adaptation naturelle de leur encodage pour des FBQ en forme prénexe normale disjonctive. De plus, dans *loc. sit.*, le test d'égalité d'étiquettes n'est pas une formule de base de la logique et de plus la satisfiabilté de la logique est définie non pas

de manière ensembliste, mais comme une relation de satisfiabilité (dans l'esprit du lemme 3.3). Néanmoins il est facile de voir pour l'opérateur d'égalité défini dans *loc. sit.* et pour tout arbre $t$ et toute valuation des variables $\rho$, $t \models \eta = \eta'$ si et seulement si $\rho(\eta) = \rho(\eta')$, et donc l'opérateur d'égalité d'étiquettes utilisé dans *loc. sit.* est équivalent à celui de la logique spatiale.

### Quantification sur les arbres

De façon similaire, nous établissons que la complexité combinée de la logique $LS_{\mathrm{Bool}X}$ est également PSPACE-difficile. Soit de nouveau $\mathcal{G} = Q_1 u_1 . \cdots . Q_n u_n . G$ une formule booléenne quantifiée en forme normale disjonctive. Nous lui associons la formule $(\!|\mathcal{G}|\!)$ de la logique $LS_{\mathrm{Bool}X}$ définie récursivement sur la structure de $\mathcal{G}$ par :

$$\begin{array}{lcl}
(\!|u_i|\!) & = & X_i \\
(\!|\overline{u_i}|\!) & = & \neg X_i \\
(\!|l_1 \wedge \cdots \wedge l_k|\!) & = & (\!|l_1|\!) \wedge \cdots \wedge (\!|l_k|\!) \\
(\!|\theta_1 \vee \cdots \vee \theta_k|\!) & = & (\!|\theta_1|\!) \vee \cdots \vee (\!|\theta_k|\!) \\
(\!|\exists u_i . \mathcal{G}|\!) & = & \exists X_i . (\!|\mathcal{G}|\!) \\
(\!|\forall u_i . \mathcal{G}|\!) & = & \forall X_i . (\!|\mathcal{G}|\!)
\end{array}$$

Il s'avère que l'interprétation de la formule $(\!|\mathcal{G}|\!)$ est ou bien l'ensemble de tous les arbres, ou bien l'ensemble vide. Nous montrons plus particulièrement que (le lemme 9.6 ci-dessous) pour tout arbre $t$, $t$ satisfait $(\!|\mathcal{G}|\!)$ si et seulement si la formule $\mathcal{G}$ est valide. Regardons d'abord ce qui se passe sur un exemple.

**Exemple 9.1** Soit $\mathcal{G}$ la formule (valide)

$$\forall u_1 . \forall u_2 . (u_1 \wedge u_2) \vee (u_1 \wedge \overline{u_2}) \vee (\overline{u_1} \wedge u_2) \vee (\overline{u_1} \wedge \overline{u_2}).$$

Alors $(\!|\mathcal{G}|\!)$ est la formule

$$\forall X_1 . \forall X_2 . (X_1 \wedge X_2) \vee (X_1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2) \vee (\neg X_1 \wedge \neg X_2).$$

Par définition, l'interprétation de cette formule est l'ensemble d'arbres (nous notons $\overline{\{t\}}$ l'ensemble $\mathcal{A}_\Sigma \setminus \{t\}$) :

$$\bigcap_{t \in \mathcal{A}_\Sigma} \bigcap_{s \in \mathcal{A}_\Sigma} (\{t\} \cap \{s\}) \cup (\{t\} \cap \overline{\{s\}}) \cup (\overline{\{t\}} \cap \{s\}) \cup (\overline{\{t\}} \cap \overline{\{s\}})$$

ce qui est égal à

$$\bigcap_{t \in \mathcal{A}_\Sigma} \left( \begin{array}{c} \left( \bigcap_{s \in \mathcal{A}_\Sigma, s \neq t} (\{t\} \cap \{s\}) \cup (\{t\} \cap \overline{\{s\}}) \cup (\overline{\{t\}} \cap \{s\}) \cup (\overline{\{t\}} \cap \overline{\{s\}}) \right) \\ \cap \\ \left( (\{t\} \cap \{t\}) \cup (\{t\} \cap \overline{\{t\}}) \cup (\overline{\{t\}} \cap \{t\}) \cup (\overline{\{t\}} \cap \overline{\{t\}}) \right) \end{array} \right)$$

On voit facilement que c'est égal à l'ensemble de tous les arbres $\mathcal{A}_\Sigma$.

Considérons maintenant l'exemple d'une formule non valide : soit $\mathcal{G}$ la formule

$$\forall u_1 . \forall u_2 . (u_1 \wedge u_2) \vee (u_1 \wedge \overline{u_2}) \vee (\overline{u_1} \wedge u_2).$$

On peut voir qu'alors $[\![(\!|\mathcal{G}|\!)]\!]$ est l'ensemble vide.

**Lemme 9.6** *Pour toute formule booléenne quantifiée $\mathcal{G}$ et pour tout arbre t, $\mathcal{G}$ est valide si et seulement si t satisfait $(\!|\mathcal{G}|\!)$.*

*Preuve* Soit $\mathcal{G}$ la formule $Q_1 u_1 . \cdots . Q_n u_n . G$. Nous montrons la propriété plus générale que pour tout $m$ dans $0..n$ et toute valuation des variables d'arbres $\rho$ de domaine $\{X_1, \ldots, X_m\}$, $t, \rho \models (\!|\mathcal{G}'|\!)$ si et seulement si $\gamma_{t,\rho} \models \mathcal{G}'$, où $\mathcal{G}'$ est la sous formule $Q_{m+1} u_{m+1} . \cdots . Q_n u_n . G$ et pour toute valuation $\rho$ de domaine $\{X_1, \ldots, X_m\}$, la valuation $\gamma_{t,\rho}$ est définie par : le domaine de $\gamma_{t,\rho}$ est $\{u_1, \ldots, u_m\}$ et pour tout $u_i$ dans $\{u_1, \ldots, u_m\}$, $\gamma_{t,\rho}(u_i) = \mathsf{vrai}$ si et seulement si $\rho(X_i) = t$. Remarquons que pour toute valuation $\gamma$ de domaine libres $\{u_1, \ldots, u_m\}$, il existe une valuation $\rho$ de domaine $\{X_1, \ldots, X_m\}$ telle que $\gamma = \gamma_{t,\rho}$. Alors l'énoncé du lemme correspond à $m = 0$. La preuve se fait par récurrence sur $n - m$.

Cas de base: Pour $m = n$, par définition on a $\mathcal{G}' = G$. Soit $G = \theta_1 \vee \cdots \vee \theta_k$ Alors, par définition de l'encodage $(\!|G|\!)$ et par les propriétés de la satisfiabilité des formules de la logique LS, il est facile de voir que $t, \rho \models (\!|G|\!)$ si et seulement si pour un certain $I$ dans $1..k$, $t, \rho \models (\!|\theta_I|\!)$. Soit un tel $I$ et soit $\theta_I = l_1 \wedge \cdots \wedge l_r$, alors par les mêmes arguments que précédemment ceci est équivalent à ce que pour chaque $j$ dans $1..r$, $t, \rho \models (\!|l_j|\!)$. Pour chaque $j$ dans $1..r$, soit $h_j$ l'indice tel que $l_j = u_{h_j}$ ou $l_j = \overline{u_{h_j}}$. Rappelons que $(\!|l_j|\!)$ est égal à $X_{h_j}$ si $l_j = u_{h_j}$ et $(\!|l_j|\!)$ est égal à $\neg X_{h_j}$ si $l_j = \overline{u_{h_j}}$. Il est maintenant facile de voir par les propriétés de la satisfiabilité des formules LS et par définition de l'encodage que pour chaque $j$ dans $1..r$, $t, \rho \models (\!|l_j|\!)$ si et seulement si $\rho(X_{h_j}) = t$ lorsque $l_j = u_{h_j}$ et $\rho(X_{h_j}) \neq t$ lorsque $l_j = \overline{u_{h_j}}$. Maintenant, par définition de $\gamma_{t,\rho}$ ceci est équivalent à $\gamma_{t,\rho}(u_{h_j}) = \mathsf{vrai}$ si $l_j = u_{h_j}$ et $\gamma_{t,\rho}(u_{h_j}) = \mathsf{faux}$ si $l_j = \overline{u_{h_j}}$, c'est à dire $\gamma_{t,\rho}$ satisfait $\theta_I$. En rappelant la quantification existentielle sur $I$, c'est équivalent à $\gamma_{t,\rho}$ satisfait $G$.

Cas d'induction: Pour $m < n$, par définition $\mathcal{G}' = Q_{m+1} u_{m+1} . \cdots . Q_n u_n . G$. Soit $\mathcal{G}''$ la formule $Q_{m+2} u_{m+2} . \cdots . Q_n u_n . G$.[1] Par définition de l'encodage, $t, \rho \models (\!|Q_{m+1} u_{m+1} . \mathcal{G}''|\!)$ si et seulement si

(i) $t, \rho \models Q_{m+1} X_{m+1} . (\!|\mathcal{G}''|\!)$.

Supposons que $Q_{m+1} = \exists$. Par satisfiabilité des formules, (i) est vérifié si et seulement si il existe un arbre $t'$ tel que $t, \rho[X_{m+1} \rightarrow t'] \models (\!|\mathcal{G}''|\!)$. Par hypothèse de récurrence c'est équivalent à

(ii) l'existence d'un arbre $t'$ tel que $\gamma_{t,\rho[X_{m+1} \rightarrow t']}$ satisfait $\mathcal{G}''$

D'autre part, par définition de satisfiabilité des formules booléennes, $\gamma_{t,\rho}$ satisfait $\mathcal{G}'$ si et seulement si

(iii) il existe un booléen b dans $\{\mathsf{vrai}, \mathsf{faux}\}$ tel que $\gamma_{t,\rho}[u_{m+1} \rightarrow \mathsf{b}]$ satisfait $\mathcal{G}''$.

Il suffit donc de montrer que (ii) si et seulement si (iii), ce qui est facile à voir par définition de la valuation $\gamma_{t,\rho[X_{m+1} \rightarrow t']}$. La preuve pour $Q_{m+1} = \forall$ est similaire.

□

Comme corollaire facile de ce résultat nous avons

**Corollaire 9.7** *Pour toute formule booléenne quantifiée $\mathcal{G}$, $\mathcal{G}$ est valide si et seulement si $[\![(\!|\mathcal{G}|\!)]\!] = \mathcal{A}_\Sigma$.*

[1] Si $n = m + 1$, nous admettons que $Q_{m+2} u_{m+2} . \cdots . Q_n u_n . G = G$.

Finalement, ceci nous permet de déduire la classe de complexité de la complexité combinée du model checking la logique $LS_{\mathrm{Bool}X}$.

**Théorème 9.8** *Le problème de model checking de la logique $LS_{BoolX}$ est PSPACE complet pour sa complexité combinée.*

*Preuve* Conséquence immédiate du lemme 9.6, du théorème 9.1 et de la proposition 8.18. □

### 9.2.2 Le fragment sans quantification et sans récursion (mini-LS)

Dans la section précédente nous avons vu qu'en présence de quantification, le problème de model checking pour la logique spatiale est PSPACE complet. Ce résultat n'est pas vraiment surprenant puisque la quantification de la logique permet d'encoder de façon assez directe la quantification des formules booléennes quantifiées. Dans ce qui suit nous montrons que l'opérateur de composition spatiale combiné à la négation est également très expressif et permet d'encoder la quantification des FBQ.

Nous présentons ici un encodage du problème de validité des formules booléennes quantifiées dans le problème de model checking de la logique mini-LS. A toute formule booléenne quantifiée $\mathcal{G}$ nous associons

- une formule de la logique mini-LS, notée $(\!|\mathcal{G}|\!)$ et
- un arbre, noté $t_{\mathcal{G}}$

tels que $t_{\mathcal{G}}$ satisfait $(\!|\mathcal{G}|\!)$ si et seulement si $\mathcal{G}$ est valide.

La formule $(\!|\mathcal{G}|\!)$ est définie récursivement sur la structure de $\mathcal{G}$ comme suit :

$$\begin{array}{lcl}
(\!|u_i|\!) & = & \mathsf{u}_i[\mathsf{vv}[\mathbf{0}]] \\
(\!|\overline{u_i}|\!) & = & \mathsf{u}_i[\mathsf{ff}[\mathbf{0}]] \\
(\!|l_1 \wedge \cdots \wedge l_k|\!) & = & (\!|l_1|\!) \mid \top \wedge \cdots \wedge (\!|l_k|\!) \mid \top \\
(\!|\theta_1 \vee \cdots \vee \theta_k|\!) & = & (\!|\theta_1|\!) \vee \cdots \vee (\!|\theta_k|\!) \\
(\!|\exists u_i.\mathcal{G}|\!) & = & \mathsf{u}_i[\top] \mid (\!|\mathcal{G}|\!) \\
(\!|\forall u_i.\mathcal{G}|\!) & = & \mathsf{u}_i[\top] \models\!\!\Rightarrow (\!|\mathcal{G}|\!)
\end{array}$$

où vv, ff et les $\mathsf{u}_i$ pour $i$ dans $1..n$ sont des étiquettes (distinctes) dans $\Sigma$.

L'arbre $t_{\mathcal{G}}$ est donné par

$$t_{\mathcal{G}} = \{\!|e_1^{\mathsf{vrai}}, e_1^{\mathsf{faux}}, \ldots, e_n^{\mathsf{vrai}}, e_n^{\mathsf{faux}}|\!\}$$

où pour chaque $i$ dans $1..n$,

$$e_i^{\mathsf{vrai}} = \mathsf{u}_i[\{\!|\mathsf{vv}[\{\!|\}\!]|\!\}] \qquad\qquad e_i^{\mathsf{faux}} = \mathsf{u}_i[\{\!|\mathsf{ff}[\{\!|\}\!]|\!\}]$$

Soit $\mathcal{G}$ la formule $Q_1 u_1 . \cdots . Q_n u_n . G$. Pour toute valuation $\gamma$ de domaine $\{u_1, \ldots, u_m\}$ pour $m \leq n$, soit $t_{\mathcal{G}}\{\gamma\}$ l'arbre composant de $t_{\mathcal{G}}$ défini par

$$t_{\mathcal{G}}\{\gamma\} = \{\!|e_1^{\gamma(u_1)}, \ldots, e_m^{\gamma(u_m)}, e_{m+1}^{\mathsf{vrai}}, e_{m+1}^{\mathsf{faux}}, \ldots, e_n^{\mathsf{vrai}}, e_n^{\mathsf{faux}}|\!\}.$$

Nous avons donc

$$t_{\mathcal{G}} = \{\!|e^{\mathsf{b}_1}, \ldots, e^{\mathsf{b}_m}_m|\!\} \uplus t_{\mathcal{G}}\{\gamma\},$$

où pour tout $i$ dans $1..m$, $\mathsf{b}_i$ est la valeur booléenne différente de $\gamma(u_i)$. Maintenant, si $\gamma'$ est une des valuations $\gamma, u_{m+1} \mapsto \mathsf{vrai}$ ou $\gamma, u_{m+1} \mapsto \mathsf{faux}$, alors l'arbre $t_{\mathcal{G}}\{\gamma'\}$ est obtenu à partir de $t_{\mathcal{G}}\{\gamma\}$ en « enlevant » de celui-ci l'élément d'arbre $e^{\mathsf{b}}_{m+1}$, où b est la valeur booléenne différente de $\gamma(u_{m+1})$. Intuitivement, c'est cette propriété qui est utilisée dans l'encodage. La formule $(\!|\mathcal{G}|\!)$ permet de « construire » successivement toutes les valuations $\gamma_m$ de domaine $\{u_1, \ldots, u_m\}$, pour $m$ allant de 1 à $n$ et en prenant en compte le type de quantification pour chacune des variables, et de tester la validité de la formule $Q_{m+1}u_{m+1} \cdots Q_n u_n.G$ sous chacune de ses valuations $\gamma_m$. Pour $n = m$, et donc $\gamma_m$ une valuation de domaine $\{u_1, \ldots, u_n\}$, ce problème de validité se réduit en le problème de satisfiabilité « $\gamma$ satisfait $G$ ».

Regardons par exemple le cas où $\mathcal{G}$ est la formule

$$\exists u_1.\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2).$$

Dans ce cas, la formule $(\!|\mathcal{G}|\!)$ est

$$\mathsf{u}_i[\top] \mid (\!|\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!).$$

L'arbre $t_{\mathcal{G}}$ satisfait $(\!|\mathcal{G}|\!)$ s'il peut être présenté comme la composition de deux arbres : $t_{\mathcal{G}} = t' \uplus t''$, tels que $t'$ est de la forme $\{\!|\mathsf{u}_i[s]|\!\}$ et $t''$ satisfait la formule $(\!|\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$. Étant donné la forme de l'arbre $t_{\mathcal{G}}$, nécessairement $t'$ est l'un des arbres $\{\!|\mathsf{u}_1[\{\!|\mathsf{vv}[\{\!|\}\!]]\!\}|\!\}$ ou $\{\!|\mathsf{u}_1[\{\!|\mathsf{ff}[\{\!|\}\!]]\!\}|\!\}$. Dans ce cas, on peut voir que $t''$ est l'un des arbres $t_{\mathcal{G}}\{u_1 \mapsto \mathsf{faux}\}$ ou $t_{\mathcal{G}}\{u_1 \mapsto \mathsf{vrai}\}$ et donc il s'agit de tester si un de ces arbres satisfait la formule $(\!|\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$. Dans la suite, nous supposons donc que nous cherchons à tester si l'arbre $t'' = t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}\}$ satisfait cette formule (pour b étant une valeur booléenne). Par définition, $(\!|\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$ est la formule

$$\mathsf{u}_2[\top] \models\!\!> (\!|(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!).$$

et l'arbre $t''$ est de la forme

$$\{\!|\mathsf{u}_1[\{\!|\mathsf{a}[\{\!|\}\!]]\!\}|\!\}, \mathsf{u}_2[\{\!|\mathsf{vv}[\{\!|\}\!]]\!\}|\!\}, \mathsf{u}_2[\{\!|\mathsf{ff}[\{\!|\}\!]]\!\}|\!\}$$

où a est une étiquette parmi vv et ff. Alors, par définition de la satisfiabilité, $t''$ satisfait la formule $(\!|\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$ si pour toute décomposition de $t''$ sous la forme $t'' = t''_1 \uplus t''_2$, si l'arbre $t''_1$ satisfait la formule $\mathsf{u}_2[\top]$, alors l'arbre $t''_2$ satisfait la formule $(\!|(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$. Les deux décompositions de $t''$ à être considérées sont

- $t''_1 = \{\!|\mathsf{u}_2[\{\!|\mathsf{vv}[\{\!|\}\!]]\!\}|\!\}$ et $t''_2 = t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{faux}\}$ et
- $t''_1 = \{\!|\mathsf{u}_2[\{\!|\mathsf{ff}[\{\!|\}\!]]\!\}|\!\}$ et $t''_2 = t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{vrai}\}$.

Donc, $t''$ satisfait la formule $(\!|\forall u_2.(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$ si et seulement si chacun des arbres $t''_2 = t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{faux}\}$ et $t''_2 = t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{vrai}\}$ satisfait la formule $(\!|(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u}_2)|\!)$.

Pour résumer les raisonnement faits jusque présent, l'arbre $t_{\mathcal{G}}$ satisfait la formule $(\!|\mathcal{G}|\!)$ si et seulement s'il existe une valeur booléenne b telle que chacun des deux arbres $t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{faux}\}$ et $t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{vrai}\}$ satisfait la formule $(\!|\mathcal{G}|\!)$. Ceci correspond bien au fait de tester s'il existe une valeur booléenne b telle que les deux valuations $u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{faux}$ et $u_1 \mapsto$

$\mathsf{b}, u_2 \mapsto$ faux satisfont $(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u_2})$. Pour terminer, rappelons que $(\!|(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u_2})|\!)$ est la formule

$$(\mathsf{u}_1[\mathsf{vv}[\mathbf{0}]] \mid \top \wedge \mathsf{u}_2[\mathsf{vv}[\mathbf{0}]] \mid \top) \ \vee \ (\mathsf{u}_1[\mathsf{vv}[\mathbf{0}]] \mid \top \wedge \mathsf{u}_2[\mathsf{ff}[\mathbf{0}]] \mid \top).$$

et les arbres $t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{faux}\}$ et $t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{vrai}\}$ sont

$$t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{faux}\} = \{\!|\mathsf{u}_1[\{\!|\mathsf{a}[\{\!|\}\!]|\}\!], \mathsf{u}_2[\{\!|\mathsf{ff}[\{\!|\}\!]|\}\!]|\}\!$$
$$t_{\mathcal{G}}\{u_1 \mapsto \mathsf{b}, u_2 \mapsto \mathsf{vrai}\} = \{\!|\mathsf{u}_1[\{\!|\mathsf{a}[\{\!|\}\!]|\}\!], \mathsf{u}_2[\{\!|\mathsf{vv}[\{\!|\}\!]|\}\!]|\}\!$$

On peut voir qu'en prenant $\mathsf{b} = \mathsf{vrai}$, l'étiquette a est égale à vv et chacun des ces deux arbres satisfait la formule $(\!|(u_1 \wedge u_2) \vee (u_1 \wedge \overline{u_2})|\!)$.

Montrons à présent formellement la correction de l'encodage.

**Lemme 9.9** *Pour toute formule booléenne quantifiée $\mathcal{G}$, $t_{\mathcal{G}} \models (\!|\mathcal{G}|\!)$ si et seulement si la formule $\mathcal{G}$ est valide.*

*Preuve* Soit $\mathcal{G}$ la formule $Q_1 u_1 . \cdots . Q_n u_n . G$. Montrons la propriété plus générale que pour tout $m$ dans $0..n$ et pour toute valuation $\gamma$ de domaine $\{u_1, \ldots, u_m\}$ et pour $\mathcal{G}'$ étant la formule $Q_{m+1} u_{m+1} . \cdots . Q_n u_n . G$, on a $\gamma \models \mathcal{G}'$ si et seulement si $t_{\mathcal{G}}\{\gamma\} \models (\!|\mathcal{G}'|\!)$. L'énoncé du lemme correspond à $m = 0$. La preuve se fait par récurrence sur $n - m$.

Cas de base: Pour $m = n$, par définition on a $\mathcal{G}' = G$. Soit $G = \theta_1 \vee \cdots \vee \theta_k$. Alors, par définition de l'encodage $(\!|G|\!)$ et par propriété de la satisfiabilité de LS, il est facile de voir que $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\mathcal{G}'|\!)$ si et seulement si pour un certain $I$ dans $1..k$, $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\theta_I|\!)$. Soit un tel $I$ et soit $\theta_I = l_1 \wedge \cdots \wedge l_r$, alors par les arguments que précédemment ceci est équivalent à ce que pour chaque $j$ dans $1..r$, $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|l_j|\!) \mid \top$. Pour chaque $j$ dans $1..r$, soit $h_j$ l'indice tel que $l_j = u_{h_j}$ ou $l_j = \overline{u_{h_j}}$. Rappelons que $(\!|l_j|\!)$ est égal à $\mathsf{u}_{h_j}[\mathsf{vv}[\mathbf{0}]] \mid \top$ si $l_j = u_{h_j}$ et $(\!|l_j|\!)$ est égal à $\mathsf{u}_{h_j}[\mathsf{ff}[\mathbf{0}]] \mid \top$ si $l_j = \overline{u_{h_j}}$. Il est maintenant facile de voir par définition de satisfiabilité des formules LS que pour chaque $j$ dans $1..r$, $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|l_j|\!) \mid \top$ si et seulement si $t_{\mathcal{G}}\{\gamma\}$ contient un élément d'arbre de la forme $\mathsf{u}_{h_j}[s]$ où l'arbre $s$ satisfait la formule $\mathsf{vv}[\mathbf{0}]$ si $l_j = u_{h_j}$ et l'arbre $s$ satisfait la formule $\mathsf{ff}[\mathbf{0}]$ si $l_j = \overline{u_{h_j}}$. Maintenant rappelant que, par définition, $t_{\mathcal{G}}\{\gamma\}$ est de la forme $\{\!|\mathsf{u}_1[s_1], \ldots, \mathsf{u}_n[s_n]|\!\}$ et pour chaque $i$ dans $1..n$, $s_i$ est l'un des arbres $\{\!|\mathsf{vv}[\{\!|\}\!]|\!\}$ ou $\{\!|\mathsf{ff}[\{\!|\}\!]|\!\}$, il est facile de voir par définition de la satisfiabilité pour LS que $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|l_1 \wedge \cdots \wedge l_r|\!)$ si et seulement si pour tout $j$ dans $1..r$, $\gamma(u_{h_j}) = \mathsf{vrai}$ si $l_j = u_{h_j}$ et $\gamma(u_{h_j}) = \mathsf{vrai}$ si $l_j = \overline{u_{h_j}}$, ce qui est vrai si et seulement si $\gamma$ satisfait $l_1 \wedge \cdots \wedge l_r$, c'est à dire $\gamma$ satisfait $\theta_I$. En rappelant la quantification existentielle sur $I$, c'est équivalent à $\gamma$ satisfait $G$.

Cas d'induction: Pour $m < n$, par définition $\mathcal{G}' = Q_{m+1} u_{m+1} . \cdots . Q_n u_n . G$. Soit $\mathcal{G}''$ la formule $Q_{m+2} u_{m+2} . \cdots . Q_n u_n . G$.[2] Nous savons par hypothèse de récurrence que pour toute valuation $\gamma'$ de domaine $\{u_1, \ldots, u_{m+1}\}$, on a $t_{\mathcal{G}}\{\gamma'\} \models (\!|\mathcal{G}''|\!)$ si et seulement si $\gamma' \models \mathcal{G}''$.

Remarquons d'abord que, par définition de l'arbre $t_{\mathcal{G}}\{\gamma\}$, on a

$$\begin{aligned} & t_{\mathcal{G}}\{\gamma\} = \{\!|\mathsf{u}_{m+1}[\{\!|\mathsf{vv}[\{\!|\}\!]|\}\!]|\}\!\} \uplus t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{faux}\} \\ \text{et} \qquad & \\ & t_{\mathcal{G}}\{\gamma\} = \{\!|\mathsf{u}_{m+1}[\{\!|\mathsf{ff}[\{\!|\}\!]|\}\!]|\}\!\} \uplus t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{vrai}\} \end{aligned} \tag{9.1}$$

[2] Si $n = m + 1$, nous admettons que $Q_{m+2} u_{m+2} . \cdots . Q_n u_n . G = G$.

De plus, pour tout arbre $t'$, si $t_{\mathcal{G}}\{\gamma\} = \{\!|\mathsf{u}_{m+1}[s]|\!\} \uplus t'$, alors $s$ est égal à $\{\!|\mathsf{vv}[\{\!|\}\!]|\!\}$ ou à $\{\!|\mathsf{ff}[\{\!|\}\!]|\!\}$. Autrement dit, ce sont les deux uniques décompositions de $t_{\mathcal{G}}\{\gamma\}$ sous la forme $\{\!|\mathsf{u}_{m+1}[s]|\!\} \uplus t'$.

Pour la preuve du cas d'induction nous distinguons deux cas suivant le quantificateur $Q_{m+1}$.

Si $Q_{m+1}$ est égal à $\exists$ alors, par définition de l'encodage, $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\mathcal{G}'|\!)$ si et seulement si $t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $\mathsf{u}_{m+1}[\top] \mid (\!|\mathcal{G}''|\!)$. Par propriétés de la satisfiabilité pour LS, ceci est équivalent à l'existence de deux arbres $s$ et $t'$ tels que $t_{\mathcal{G}}\{\gamma\} = \{\!|\mathsf{u}_{m+1}[s]|\!\} \uplus t'$ et $t' \models (\!|\mathcal{G}''|\!)$. En utilisant (9.1) et la remarque qui la suit, ceci est vrai si et seulement si $s = \{\!|\mathsf{vv}[\{\!|\}\!]|\!\}$ et $t' = t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{faux}\}$ ou bien $s = \{\!|\mathsf{ff}[\{\!|\}\!]|\!\}$ et $t' = t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{vrai}\}$. En appliquant l'hypothèse de récurrence sur $t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{vrai}\}$ et $t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{faux}\}$ nous déduisons que $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\mathcal{G}'|\!)$ si et seulement si $\gamma, u_{m+1} \mapsto \mathsf{vrai} \models \mathcal{G}''$ ou bien $\gamma, u_{m+1} \mapsto \mathsf{faux} \models \mathcal{G}''$. Maintenant, en utilisant la définition de la satisfiabilité pour les formules propositionnelles et de $\mathcal{G}'$, il est facile de voir que ceci est équivalent à $\gamma \models \exists u_{m+1}\mathcal{G}''$, ce qu'il fallait démontrer.

Si $Q_{m+1}$ est égal à $\forall$ alors, par définition de l'encodage, $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\mathcal{G}'|\!)$ si et seulement si $t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $\mathsf{u}_{m+1}[\top] \Rrightarrow (\!|\mathcal{G}''|\!)$. Par les propriétés de la satisfiabilité, ceci est vrai si et seulement si pour tous arbres $t'$ et $t''$ tels que $t = t' \uplus t''$, si $t' \models \mathsf{u}_{m+1}[\top]$, alors $t'' \models (\!|\mathcal{G}''|\!)$. La seule possibilité pour les arbres $t'$ et $t''$ qui garantit $t' \models \mathsf{u}_{m+1}[\top]$ est $t' = \{\!|\mathsf{u}_{m+1}[s]|\!\}$ pour un certain arbre $s$. Donc, $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\mathcal{G}'|\!)$ si et seulement si $t' = \{\!|\mathsf{u}_{m+1}[s]|\!\}$ pour un certain arbre $s$ et $t'' \models (\!|\mathcal{G}''|\!)$. Maintenant, par (9.1) nous savons que $t' = \{\!|\mathsf{u}_{m+1}[s]|\!\}$ si et seulement si $s = \{\!|\mathsf{vv}[\{\!|\}\!]|\!\}$ et $t'' = t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{faux}\}$ ou bien $s = \{\!|\mathsf{ff}[\{\!|\}\!]|\!\}$ et $t'' = t_{\mathcal{G}}\{\gamma, u_{m+1} \mapsto \mathsf{vrai}\}$. En appliquant l'hypothèse de récurrence sur les valuations $\gamma, u_{m+1} \mapsto \mathsf{faux}$ et $\gamma, u_{m+1} \mapsto \mathsf{vrai}$, nous déduisons que $t_{\mathcal{G}}\{\gamma\}$ satisfait $(\!|\mathcal{G}'|\!)$ si et seulement si $\gamma, u_{m+1} \mapsto \mathsf{vrai}$ satisfait $\mathcal{G}''$ et $\gamma, u_{m+1} \mapsto \mathsf{faux}$ satisfait $\mathcal{G}''$. Par définition de la satisfiabilité pour les formules propositionnelles ceci est équivalent à $\gamma$ satisfait $\forall u_{m+1}.\mathcal{G}''$, ce qu'il fallait démontrer.

□

Comme conséquence immédiate du précédent lemme, du théorème 9.1 établissant que le problème de validité des FBQ est PSPACE difficile et de la proposition 8.18 établissant que la complexité du model checking de la logique LS est polynomial en espace,

**Théorème 9.10** *Le problème de model checking de la logique mini-LS est PSPACE complet pour sa complexité combinée.*

## 9.3 Bornes inférieures pour la complexité de données

Dans la présente section nous étudions la complexité de données du problème de model checking pour la logique LS sans restriction et ses fragments sans quantification $\mathrm{LS}_{|\exists}$ et sans récursion $\mathrm{LS}_{|\mu}$.

En ce qui concerne le fragment sans quantification, les résultats sur la complexité de données sont obtenus à partir de l'équivalence entre ce fragment et une classe d'automates à contraintes numériques, équivalence établie dans la deuxième partie de cette thèse. Nous donnons ces résultats dans la section 9.3.1.

Pour les deux autres logiques considérées, LS et $\mathrm{LS}_{|\mu}$, nous établissons que la complexité de données est respectivement PSPACE-difficile et complet pour tous les niveaux de la hiérarchie polynomiale. Ces deux résultats sont obtens par des encodages du problème de validité des FBQ.

Plus précisément, pour chacun des deux fragments de la logique, une FBQ $\mathcal{G}$ est représentée par un arbre $t_\mathcal{G}$. Ensuite, nous construisons une formule $\phi$ constante[3] et nous montrons que $t_\mathcal{G}$ satisfait la formule $\phi$ si et seulement si la FBQ $\mathcal{G}$ est valide. L'arbre $t_\mathcal{G}$ et la formule $\phi$ diffèrent pour les deux fragments de la logique spatiale considérés, LS et $\text{LS}_{|\mu}$. Néanmoins, il y a certains points communs entre les deux. Nous commençons par exposer ces points communs dans la section 9.3.2, où nous proposons un arbre $t_\mathcal{G}$ « générique » et la façon dont il permet d'encoder la FBQ $\mathcal{G}$. Dans les sections 9.3.3 et 9.3.4 nous présentons comment l'arbre $t_\mathcal{G}$ peut être adapté aux cas LS et au cas $\text{LS}_{|\mu}$ respectivement, nous définissons la formule $\phi$ (différente pour les deux cas) et nous montrons les résultats sur les bornes inférieures de la complexité de données.

### 9.3.1 La logique sans quantification

Dans les deux premières parties de la thèse nous avons établi certaines équivalences entre des fragments de la logique spatiale sans quantification et les logiques MSO et PMSO, ainsi que l'équivalence entre $\text{LS}_{|\exists}$ et une classe d'automates à contraintes numériques. Nous nous contentons de rappeler ici ces résultats et d'en déduire des bornes inférieures pour la complexité de données des fragments concernés.

Rappelons d'abord que les fragments LSrd et LSrpd sont des fragments syntaxiques de la logique $\text{LS}_{|\exists}$, obtenus par des restrictions du mécanisme de récursion. En particulier, dans LSrd, seule la récursion descendante est autorisées et dans LSrpd, la récursion descendante et une forme restreinte de récursion en largeur sont autorisés (pour une définition précise de ces deux fragments, voir la section 7.1).

Nous avons établi dans les deux premières parties de la thèse que :
- la logique LSrd est équivalente à la logique MSO. De plus, pour toute formule de LSrd, une formule MSO équivalente peut effectivement être construite ;
- la logique LSrpd est équivalente à la logique PMSO. De plus, pour toute formule de LSrpd, une formule PMSO équivalente peut effectivement être construite ;
- pour toute formule $\text{LS}_{|\exists}$, un automate à contraintes numériques reconnaissant exactement les arbres satisfaisant la formule peut être construit.

Nous savons également que la complexité de données du model checking de MSO sur les graphes à largeur d'arbre bornée est linéaire [Courcelle, 1990a].[4] et que la complexité de données de la logique PMSO est linéaire [Seidl et al., 2003]. Maintenant, étant donné que nous nous intéressons à la complexité de données, pour une formule $\phi$ dans LSrd (resp. dans LSrpd), la construction de la formule équivalente $\psi$ dans MSO (resp. PMSO) se fait en temps constant. De plus, comme nous l'avons vu dans le chapitre 1, pour tout arbre $t$, sa représentation sous la forme de structure logique $S^t$ se fait en temps linéaire dans la taille de $t$. Nous en déduisons que :

**Proposition 9.11** *Le model checking des logiques LSrd et LSrpd est linéaire pour sa complexité de données.*

[3] Pour être plus précis, dans le cas de $\text{LS}_{|\mu}$ nous construisons une famille de formules $\phi$, une formule par niveau de la hiérarchie polynomiale.

[4] Sans donner la définition de la largeur d'arbre d'un graphe, notons simplement que les arbres sont des graphes de largeur d'arbre un.

Remarquons que le même résultat peut être établi en utilisant l'équivalence des fragments LSrd et LSrpd avec les automates à contraintes numériques sans étoile et les automates à contraintes numériques semilinéaires établie dans le chapitre 7.

En ce qui concerne la logique $\mathrm{LS}_{|\exists}$, en utilisant son équivalence avec les automates à contraintes numériques nous avons établi (corollaire 6.20 pour un arbre $t$ et une formule $\mathrm{LS}_{|\exists}$ close $\phi$ donnés, tester si $t$ satisfait $\phi$ peut se faire en temps polynomial dans la taille de l'arbre $t$, mais le degré du polynome est en $\mathcal{O}(3^{|\phi|})$. Nous en avons conclu que pour un problème de model checking $t \models \phi$ donné, la complexité de données de ce problème est polynomiale dans la taille de l'arbre $t$, mais que pour la classe de problèmes de model checking $t \models \phi$ pour $\phi$ étant une formule $\mathrm{LS}_{|\exists}$, la complexité de données n'est pas polynomiale, puisqu'il n'existe pas de polynome de degré fixe qui borne cette complexité.

### 9.3.2 Premier encodage des formules propositionnelles

Dans les trois sections qui suivent, nous allons étudier les bornes de la complexité de données de la logique spatiale LS ainsi que de son fragment sans point fixe $\mathrm{LS}_{|\mu}$. Nous montrons que la complexité de données du model checking est PSPACE difficile pour la logique LS et difficile pour tous les niveaux de la hiérarchie polynomiale pour la logique $\mathrm{LS}_{|\mu}$. Pour cela, nous utilisons un encodage de la validité des FBQ dans le problème de model checking de ces deux logiques. Dans les deux cas, étant donné une FBQ $\mathcal{G}$, nous construisons un arbre $t_\mathcal{G}$ et une formule $\phi$ fixe, et nous montrons que $t_\mathcal{G}$ satisfait $\phi$ si et seulement si la FBQ $\mathcal{G}$ est valide. Si pour les deux fragments que nous considérons, l'arbre $t_\mathcal{G}$ et la formule $\phi$ diffèrent, les arbres $t_\mathcal{G}$ considérés dans les deux cas ont certains points communs, et une façon commune d'encoder la formule $\mathcal{G}$. Dans la présente section, nous présentons une forme générale de l'arbre $t_\mathcal{G}$ et comment cet arbre est utilisé pour encoder la FBQ $\mathcal{G}$.

Dans toute cette section, nous considérons la FBQ $\mathcal{G} = Q_1 u_1 . \cdots . Q_n u_n . G$ avec $G$ étant la formule propositionnelle en forme normale disjonctive $\theta_i \vee \cdots \vee \theta_k$ construite sur l'ensemble de variables $\{u_1, \ldots, u_n\}$. Sans perte de généralité, nous supposons qu'aucun des $\theta_j$ n'est faux de manière triviale, càd aucun des $\theta_j$ ne contient à la fois la variable $u$ et sa négation $\overline{u}$, ceci pour toute variable $u$.

Pour tout $i$ dans $1..n$, soient $C_i^{\mathsf{vrai}} \subseteq \{1, \ldots, k\}$ et $C_i^{\mathsf{faux}} \subseteq \{1, \ldots, k\}$ les ensembles définis par :

$$C_i^{\mathsf{vrai}} = \{j \mid \overline{u_i} \text{ n'apparaît pas dans } \theta_j\}$$
$$C_i^{\mathsf{faux}} = \{j \mid u_i \text{ n'apparaît pas dans } \theta_j\}.$$

Intuitivement, $C_i^{\mathsf{vrai}}$ contient les $j$ tels que $\Theta_j$ n'est pas rendu si la variable $u_i$ est valuée à vrai (càd, les $\Theta_j$ qui « ont une change » de valoir vrai si la variable $u_i$ vaut vrai) et $C_i^{\mathsf{faux}}$ contient les $j$ tels que $\Theta_j$ n'est pas rendu faux si la variable $u_i$ est valuée à faux.

L'exemple 9.2 montre ces deux ensembles pour une formule particulière. Alors l'ensemble des ensembles $C_i^{\mathsf{vrai}}$ et $C_i^{\mathsf{faux}}$ pour chaque $i$ dans $1..n$ encodent la satisfiabilité de la formule $G$ de la façon suivante :

**Lemme 9.12** *Pour toute valuation $\gamma$ de domaine incluant $\{u_1, \ldots, u_n\}$, $\gamma$ satisfait $G$ si et seulement s'il existe un naturel $j$ dans $\{1, \ldots, k\}$ tel que pour tout $i$ dans $1..n$, $j$ appartient à $C_i^{\gamma(u_i)}$.*

*Preuve* Par définition des ensembles $C_i^{\mathsf{vrai}}$ et $C_i^{\mathsf{faux}}$ il suffit de montrer que $\gamma$ satisfait $G$ si et seulement s'il existe $j$ dans $1..k$ tel que pour tout $i$ dans $1..n$,

(i) si $\gamma(u_i) = \mathsf{vrai}$ alors $\overline{u_i}$ n'apparaît pas dans $\theta_j$ et si $\gamma(u_i) = \mathsf{faux}$ alors $u_i$ n'apparaît pas dans $\theta_j$.

Supposons d'abord que $\gamma$ satisfait $G$, donc par définition de la satisfiabilité des FBQ, il existe $j$ dans $1..k$ tel que $\gamma$ satisfait $\theta_j$. Maintenant pour toute variable $u_i$, si $\gamma(u_i) = \mathsf{vrai}$, alors $\gamma$ ne satisfait pas $\overline{u_i}$ et donc $\overline{u_i}$ n'apparaît pas dans $\theta_j$. De même, si $\gamma(u_i) = \mathsf{faux}$, alors $u_i$ n'apparaît pas dans $\theta_j$. Donc, $\gamma$ satisfait $G$ implique (i). Supposons maintenant que (i) est vérifié et montrons que $\gamma$ satisfait $\theta_j$, ce qui implique que $\gamma$ satisfait $G$. Pour toute variable $u_i$ apparaissant dans $\theta_j$, par (i) nécessairement $\gamma(u_i) = \mathsf{vrai}$ et donc $\gamma$ satisfait $u_i$. De même, pour toute négation de variable $\overline{u_i}$ apparaissant dans $\theta_j$, par (i) nécessairement $\gamma(u_i) = \mathsf{faux}$ et donc $\gamma$ satisfait $\overline{u_i}$. Il s'en suit par définition de la satisfiabilité que $\gamma$ satisfait $\theta_j$.

□

Soient maintenant $\mathsf{sat}$ et $\mathsf{c}_1, \ldots, \mathsf{c}_k$ des étiquettes de $\Sigma$ et soient, pour tout $u_i$ variable dans la formule $\mathcal{G}$, les arbres $s_i^{\mathsf{vrai}}$ et $s_i^{\mathsf{faux}}$ définis par :

$$s_i^{\mathsf{vrai}} = \biguplus_{j \in C_i^{\mathsf{vrai}}} \{\!|\mathsf{c}_j[\{\!|\}]|\!\} \qquad s_i^{\mathsf{faux}} = \biguplus_{j \in C_i^{\mathsf{faux}}} \{\!|\mathsf{c}_j[\{\!|\}]|\!\}.$$

Considérons l'arbre $t_{\mathcal{G}}$

$$t_{\mathcal{G}} = \{\!|e_1^{\mathsf{vrai}}, e_1^{\mathsf{faux}}, \ldots, e_n^{\mathsf{vrai}}, e_n^{\mathsf{faux}}|\!\}$$

pour les éléments d'arbres $e_i^{\mathsf{vrai}}$ et $e_i^{\mathsf{faux}}$ définis par

$$e_i^{\mathsf{vrai}} = \mathsf{u}_i[t_i^{\mathsf{vrai}} \uplus \{\!|\mathsf{vv}[\{\!|\}], \mathsf{sat}[s_i^{\mathsf{vrai}}]|\!\}] \qquad e_i^{\mathsf{faux}} = \mathsf{u}_i[t_i^{\mathsf{faux}} \uplus \{\!|\mathsf{ff}[\{\!|\}], \mathsf{sat}[s_i^{\mathsf{faux}}]|\!\}]$$

où les arbres $t_i^{\mathsf{vrai}}$ et $t_i^{\mathsf{faux}}$ sont tels qu'ils ne contiennent pas l'étiquette $\mathsf{sat}$ ni les étiquettes $\mathsf{vv}$ ou $\mathsf{ff}$ (voir l'exemple 9.2). La formule $Partout(x_{\mathsf{c}})$ de variable d'étiquette libre $x_{\mathsf{c}}$ est définie par :

$$Partout(x_{\mathsf{c}}) = \forall x_{\mathsf{u}}.(x_{\mathsf{u}}[\top] \,|\, \top \rightarrow x_{\mathsf{u}}[\mathsf{sat}[x_{\mathsf{c}}[\mathbf{0}] \,|\, \top] \,|\, \top] \,|\, \top)$$

Avec ces conventions, nous avons

**Lemme 9.13** *Pour toute valuation $\gamma$ de domaine $\{u_1, \ldots, u_n\}$, l'arbre $t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $\exists x_{\mathsf{c}}.Partout(x_{\mathsf{c}})$ si et seulement si $\gamma$ satisfait $G$.*

*Preuve* Nous allons montrer que $t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $\exists x_{\mathsf{c}}.Partout(x_{\mathsf{c}})$ si et seulement s'il existe $j$ dans $1..k$ tel que l'élément d'arbre $\mathsf{c}_j[\{\!|\}]$ est élément de $s_i^{\gamma(u_i)}$ pour tout $i$ dans $1..n$. Alors la conclusion découle de la définition des arbres $s_i^{\mathsf{vrai}}$ et $s_i^{\mathsf{faux}}$ et du lemme 9.12.

Soit $\phi(x_{\mathsf{c}}, x_{\mathsf{u}})$ la formule $x_{\mathsf{u}}[\top] \,|\, \top \rightarrow x_{\mathsf{u}}[\mathsf{sat}[x_{\mathsf{c}}[\mathbf{0}] \,|\, \top] \,|\, \top] \,|\, \top$. Par propriétés de satisfiabilité des formules de la logique, il est facile de voir que $t_{\mathcal{G}}\{\gamma\}$ satisfait $\exists x_{\mathsf{c}}.Partout(x_{\mathsf{c}})$ si et seulement si

(i) il existe une étiquette $\mathsf{c}$ dans $\Sigma$ telle que pour toute étiquette $\mathsf{u}$ dans $\Sigma$, $t_{\mathcal{G}}\{\gamma\}$ satisfait $\phi(x_{\mathsf{c}}, x_{\mathsf{u}})$ sous la valuation $\rho = [x_{\mathsf{c}} \rightarrow \mathsf{c}][x_{\mathsf{u}} \rightarrow \mathsf{u}]$.

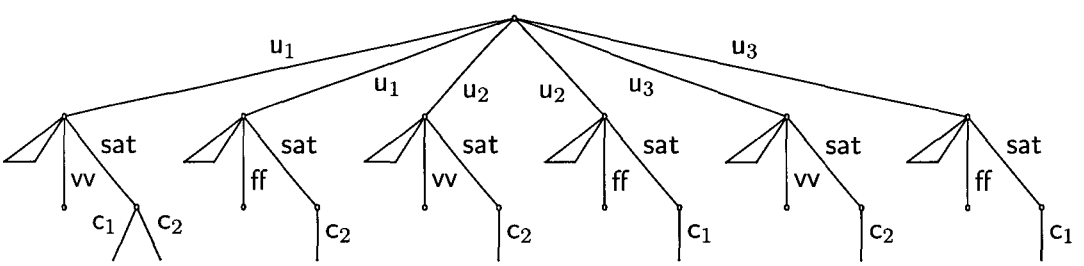
Remarquons maintenant que si u est une étiquette qui n'appartient pas à l'ensemble $A = \bigcup_{u_i \in 1..n} \{\mathsf{u}_i\}$, alors $t_{\mathcal{G}}\{\gamma\}, \rho \not\models x_{\mathsf{u}}[\top] \mid \top$ et donc $t_{\mathcal{G}}\{\gamma\}, \rho \models \phi(x_{\mathsf{c}}, x_{\mathsf{u}})$. Si u est dans l'ensemble $A$, alors $t_{\mathcal{G}}\{\gamma\}, \rho \models x_{\mathsf{u}}[\top] \mid \top$. Donc (i) est vérifié si et seulement s'il existe une étiquette c dans $\Sigma$ telle que pour toute étiquette u dans $A$, $t_{\mathcal{G}}\{\gamma\}$ satisfait $x_{\mathsf{u}}[\mathsf{sat}[x_{\mathsf{c}}[\mathbf{0}] \mid \top] \mid \top] \mid \top$ sous la valuation $[x_{\mathsf{c}} \to \mathsf{c}][x_{\mathsf{u}} \to \mathsf{u}]$. Par propriétés de la satisfiabilité de la logique, cette dernière assertion est vérifiée si et seulement si l'arbre $t_{\mathcal{G}}$ peut s'écrire comme $t' \uplus t''$ où pour tout u dans $A$, $t', [x_{\mathsf{c}} \to \mathsf{c}][x_{\mathsf{u}} \to \mathsf{u}] \models x_{\mathsf{u}}[\mathsf{sat}[x_{\mathsf{c}}[\mathbf{0}] \mid \top] \mid \top]$ et $t''$ est un arbre quelconque. En rappelant la définition de $t_{\mathcal{G}}\{\gamma\}$ et en utilisant les propriétés de la satisfiabilité, il est facile de voir que ceci est vrai si et seulement si pour toute variable $i$ dans $1..n$, l'arbre $\{\!|e_i^{\gamma(u_i)}|\!\}$ satisfait la formule $x_{\mathsf{u}}[\mathsf{sat}[x_{\mathsf{c}}[\mathbf{0}] \mid \top] \mid \top]$ sous la valuation $[x_{\mathsf{c}} \to \mathsf{c}][x_{\mathsf{u}} \to \mathsf{u}_i]$. En utilisant la définition de de l'élément d'arbre $e_i^{\gamma(u_i)}$ et les propriétés de la satisfiabilité, on en déduit facilement que $\mathsf{c}_j[\{\!|\}\!]$ est élément de $s_i^{\gamma(u_i)}$ pour tout $i$ dans $1..n$.

□

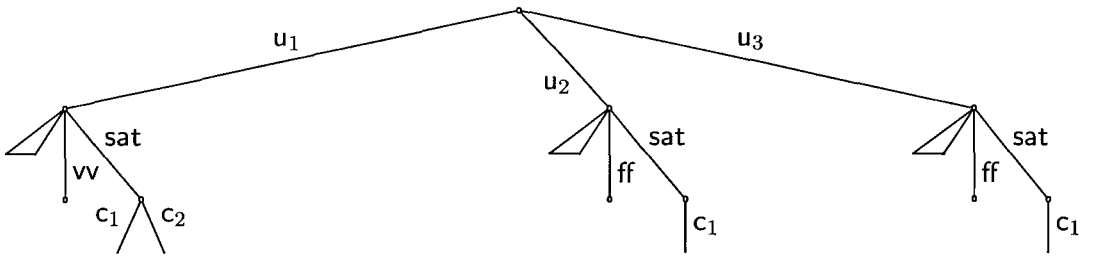
**Exemple 9.2** Soit la FBQ $\mathcal{G} = Q_1 u_1 . Q_2 u_2 . Q_3 u_3 . (u_1 \wedge \overline{u_2} \wedge \overline{u_3}) \vee (u_2 \wedge u_3)$ construite sur l'ensemble de variables $\{u_1, u_2, u_3\}$, et soient $\theta_1$ étant la conjonction $u_1 \wedge \overline{u_2} \wedge \overline{u_3}$ et $\theta_2$ la conjonction $u_2 \wedge u_3$. Les ensembles $C_i^{\mathsf{vrai}}$ et $C_i^{\mathsf{faux}}$, pour $i$ dans $\{1, 2, 3\}$ sont alors :

$$C_1^{\mathsf{vrai}} = \{1, 2\} \qquad C_1^{\mathsf{faux}} = \{2\}$$
$$C_2^{\mathsf{vrai}} = \{2\} \qquad C_2^{\mathsf{faux}} = \{1\}$$
$$C_3^{\mathsf{vrai}} = \{2\} \qquad C_3^{\mathsf{faux}} = \{1\}$$

L'arbre $t_{\mathcal{G}}$ tel que défini ci-dessus est alors de la forme




Si $\gamma$ est la valuation $u_1 \mapsto$ vrai, $u_2 \mapsto$ faux, $u_3 \mapsto$ faux, alors l'arbre $t_{\mathcal{G}}\{\gamma\}$ est l'arbre suivant




et la formule $u_1 \wedge \overline{u_2} \wedge \overline{u_3}) \vee (u_2 \wedge u_3$ est valide sous la valuation $\gamma$, ce qu'on voit par la présence d'une étiquette, en l'occurrence $\mathsf{c}_1$, que l'on retrouve sous toutes les arêtes sat dans l'arbre $t_{\mathcal{G}}$.

Les lemmes précédents nous montrent comment, étant donné la formule propositionnelle $G$ et la valuation $\gamma$ des variables de $G$, on peut encoder dans l'arbre $t_{\mathcal{G}}\{\gamma\}$ la valuation $\gamma$ et la formule $G$ et comment on peut décider si $\gamma$ satisfait $G$ répondant au problème de model checking $t_{\mathcal{G}}\{\gamma\} \models \exists x_c.Partout(x_c)$, où $Partout(x_c)$ est une formule fixe de la logique $\mathrm{LS}_{|\mu}$. Nous allons utiliser ce mécanisme dans les deux sections qui suivent.

### 9.3.3 La logique LS

Soit la FBQ $\mathcal{G} = Q_{u_1}u_1.\cdots.Q_{u_n}u_n.G$, où la formule $G$ est de la forme $\theta_1 \vee \cdots \vee \theta_k$. Nous allons définir un arbre $t_{\mathcal{G}}$ et une formule $\phi$ de la logique LS qui ne dépend pas de $\mathcal{G}$ et tels que $t_{\mathcal{G}} \models \phi$ si et seulement si la formule $\mathcal{G}$ est valide. L'arbre $t_{\mathcal{G}}$ est une adaptation de la forme générale qui a été présentée dans la section précédente.

Dans ce qui suit, $\mathsf{u}_i$ pour tout $i$ dans $1..n$, $\mathsf{c}_{\theta_j}$ pour tout $j$ dans $1..k$, $\mathsf{q}_\exists$, $\mathsf{q}_\forall$, vv, ff prec, quant, sat et $\mathsf{u}_0$ sont des étiquettes dans $\Sigma$.

Soit $t_{\mathcal{G}}$ l'arbre :

$$t_{\mathcal{G}} = \{\!|\mathsf{u}_0[\{\!|\,|\!\}], e_1^{\mathsf{vrai}}, e_1^{\mathsf{faux}}, \ldots, e_n^{\mathsf{vrai}}, e_n^{\mathsf{faux}}|\!\}$$

où, pour tout $i$ dans $1..n$, les éléments d'arbre $e_i^{\mathsf{vrai}}$ et $e_i^{\mathsf{faux}}$ sont donnés par :

$$e_i^{\mathsf{vrai}} = \mathsf{u}_i[\{\!|\mathsf{vv}[\{\!|\,|\!\}], \mathsf{quant}[\{\!|\mathsf{q}_{Q_i}[\{\!|\,|\!\}]|\!\}], \mathsf{prec}[\{\!|\mathsf{u}_{i-1}[\{\!|\,|\!\}]|\!\}], \mathsf{sat}[s_i^{\mathsf{vrai}}]|\!\}]$$
$$e_i^{\mathsf{faux}} = \mathsf{u}_i[\{\!|\mathsf{ff}[\{\!|\,|\!\}], \mathsf{quant}[\{\!|\mathsf{q}_{Q_i}[\{\!|\,|\!\}]|\!\}], \mathsf{prec}[\{\!|\mathsf{u}_{i-1}[\{\!|\,|\!\}]|\!\}], \mathsf{sat}[s_i^{\mathsf{faux}}]|\!\}]$$

Intuitivement, pour chaque variable $u_i$, l'arbre $t_{\mathcal{G}}$ encode le type de quantification de cette variable (existentielle ou universelle) dans l'élément d'arbre $\mathsf{quant}[\{\!|\mathsf{q}_{Q_i}[\{\!|\,|\!\}]|\!\}]$ et sa position dans la suite de quantifications, en précisant la variable qui précède dans l'élément d'arbre $\mathsf{prec}[\{\!|\mathsf{u}_{i-1}[\{\!|\,|\!\}]|\!\}]$. Les éléments d'arbres $\mathsf{sat}[s_i^{\mathsf{vrai}}]$ et $\mathsf{sat}[s_i^{\mathsf{faux}}]$ permettront de vérifier pour chaque valuation des variables $\gamma$ si $\gamma$ satisfait $G$ (voir le lemme 9.12 et la définition des arbres $s_i^{\mathsf{vrai}}$ et $s_i^{\mathsf{faux}}$).

La formule $\phi$ est définie par :

$$\phi = \mu\xi.ConstrVal \vee (EstValuée \wedge Valide).$$

La sous formule $ConstrVal$ est la partie récursive de $\phi$. Intuitivement, $ConstrVal$ « construit » toutes les valuations $\gamma$ de domaine $\{u_1, \ldots, u_n\}$, càd cette formule permet de considérer l'arbre $t_{\mathcal{G}}$ comme la composition de deux arbres $t'$ et $t''$ tels que $t'' = t_{\mathcal{G}}\{\gamma\}$ pour une certaine valuation $\gamma$ de domaine $\{u_1, \ldots, u_n\}$. La sous formule $EstValuée \wedge Valide$ permet de s'assurer que la valuation $\gamma$ est de domaine $\{u_1, \ldots, u_n\}$ et de vérifier si $\gamma$ satisfait $G$. Regardons maintenant les définitions complètes des différentes sous formules de $\phi$.

$$ConstrVal = \exists x_{\mathsf{u}}.PremierNonVal(x_{\mathsf{u}}) \wedge \left( \begin{array}{c} QuantUniv(x_{\mathsf{u}}) \wedge x_{\mathsf{u}}[\top] \Rrightarrow \xi \\ \vee \\ QuantExist(x_{\mathsf{u}}) \wedge x_{\mathsf{u}}[\top] \mid \xi \end{array} \right)$$

$$PremierNonVal(x_{\mathsf{u}}) = \exists x'_{\mathsf{u}}. \left( \begin{array}{c} \neg(x'_{\mathsf{u}}[\top] \mid x'_{\mathsf{u}}[\top] \mid \top) \\ \wedge \\ x_{\mathsf{u}}[\top] \mid x_{\mathsf{u}}[\top] \mid \top \\ \wedge \\ x_{\mathsf{u}}[\mathsf{prec}[x'_{\mathsf{u}}[\top]] \mid \top] \mid \top \end{array} \right)$$

$$\begin{aligned} QuantUniv(x_{\mathsf{u}}) &= x_{\mathsf{u}}[\mathsf{quant}[\mathsf{q}_\forall[\mathbf{0}]] \mid \top] \mid \top \\ QuantExist(x_{\mathsf{u}}) &= x_{\mathsf{u}}[\mathsf{quant}[\mathsf{q}_\exists[\mathbf{0}]] \mid \top] \mid \top \end{aligned}$$

$$\begin{aligned} EstValuée &= \neg(\exists x_{\mathsf{u}}.x_{\mathsf{u}}[\top] \mid x_{\mathsf{u}}[\top] \mid \top) \\ Valide &= \mathsf{u}_0[\top] \mid \exists x_{\mathsf{c}}.Partout(x_{\mathsf{c}}) \end{aligned}$$

L'encodage est correct dans le sens où :

**Lemme 9.14** *La formule booléenne quantifiée $\mathcal{G}$ est valide si et seulement si $t_{\mathcal{G}} \models \phi$.*

*Preuve* Soit $\mathcal{G}$ la formule $Q_1 u_1.\cdots.Q_n u_n.G$. Nous montrons que pour tout $m$ dans $0..n$ et pour toute valuation $\gamma$ de domaine $\{u_1, \ldots, u_n\}$ et pour $\mathcal{G}'$ étant la formule $Q_{m+1}u_{m+1}.\cdots.Q_n u_v$, l'arbre $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $\phi$ si et seulement si $\gamma$ satisfait $\mathcal{G}'$. La preuve du lemme correspond à $m = 0$.

Nous montrons d'abord que :

(i) $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $EstValuée$ si et seulement si $m = 0$ ;

(ii) $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $PremierNonVal(x_{\mathsf{u}})$ sous la valuation $\rho = [x_{\mathsf{u}} \rightarrow \mathsf{u}]$ si et seulement si $m < n$ et $\mathsf{u} = \mathsf{u}_{m+1}$ ;

(iii) $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $QuantUniv(x_{\mathsf{u}})$ (respectivement $QuantExist(x_{\mathsf{u}})$) sous la valuation $[x_{\mathsf{u}} \rightarrow \mathsf{u}]$ si et seulement si $\mathsf{u} = \mathsf{u}_i$ pour un certain $i$ dans $1..n$ et $Q_i = \forall$ (respectivement $Q_i = \exists$).

Pour la preuve du point (ii), supposons d'abord que $m = 0$ et que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ ne satisfait pas la formule $EstValuée$, c'est à dire, par propriétés de la satisfiabilité des formules et de $EstValuée$, $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $\exists x_{\mathsf{u}}.x_{\mathsf{u}}[\top] \mid x_{\mathsf{u}}[\top] \mid \top$. Ceci implique qu'il existe une étiquette $\mathsf{u}$ dans $\Sigma$ telle que l'arbre $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ peut s'écrire comme la composition $t_1 \uplus t_2 \uplus t_3$ où l'arbre $t_3$ est quelconque et les arbres $t_1$ et $t_2$ satisfont la formule $x_{\mathsf{u}}[\top]$ sous la valuation $[x_{\mathsf{u}} \rightarrow \mathsf{u}]$. C'est à dire, $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ contient deux élément d'arbres qui sont de la forme $\mathsf{u}[s]$ et $\mathsf{u}[s']$ pour deux arbres quelconques $s$ et $s'$. Il est facile de voir par définition de $t_{\mathcal{G}}\{\gamma\}$ que ce n'est pas le cas, et donc $m = 0$ implique que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait la formule

*EstValuée*. Avec un raisonnement similaire nous arrivons à une contradiction en supposant que $m > 0$ et $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait la formule *EstValuée*, ce qui permet de conclure la preuve du point (ii).

Pour la preuve du point (i), par un raisonnement similaire à celui utilisé pour la preuve du point (ii) nous savons que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $\mathit{PremierNonVal}(x_\mathsf{u})$ sous la valuation $\rho = [x_\mathsf{u} \to \mathsf{u}]$ si et seulement s'il existe une étiquette $\mathsf{u}'$ dans $\Sigma$ telle que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ contient deux éléments d'arbres $\mathsf{u}[s_1]$ et $\mathsf{u}[s_2]$ tels que l'arbre $s_1$ est de la forme $\mathsf{prec}[\mathsf{u}'[s_3]] \,|\, s_4$ et $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ ne contient pas deux éléments d'arbres de la forme $\mathsf{u}'[s_5]$ et $\mathsf{u}'[s_6]$ (où les arbres $s_2, s_3, s_4, s_5$ et $s_6$ sont quelconques). Il est facile à voir que ceci correspond à la définition de l'arbre $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ si et seulement si $m$ est inférieur strictement à $n$, $\mathsf{u}' = \mathsf{u}_m$ et $\mathsf{u} = \mathsf{u}_{m+1}$.

Le point (iii) est conséquence des propriétés de la satisfiabilité et des définition de l'arbre $t_{\mathcal{G}}\{\gamma\}$ et des formules *QuantUniv* et *QuantExist*.

La preuve du lemme se fait par récurrence sur $n - m$. Rappelons d'abord que, par définition du point fixe de la logique LS et par définition de la formule $\phi$, la formule $\phi$ est équivalente à la formule $\mathit{ConstrVal}\langle \xi \to \phi \rangle \vee (\mathit{EstValuée} \wedge \mathit{Valide})$.

Cas de base: Si $n = m$, alors en rappelant la définition de la formule *ConstrVal* et en utilisant les points (i) et (ii) ci-dessus, il est facile de voir que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ ne satisfait pas $\mathit{ConstrVal}\langle \xi \to \phi \rangle$ et $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait *EstValuée*. Donc il nous faut montrer que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait *Valide* si et seulement si $\gamma$ satisfait $G$. Ceci est conséquence de la définition de la formule *Valide*, du lemme 9.14 et de la satisfiabilité des formules de la logique LS.

Cas d'induction: Si $0 < m \leq n$, alors par le point (i) ci-dessus, $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ ne satisfait pas *EstValuée*. Donc il nous faut montrer que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $\mathit{ConstrVal}\langle \xi \to \phi \rangle$ si et seulement si $\gamma$ satisfait $\mathcal{G}'$. D'autre part, par le point (ii) nous savons que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $\mathit{PremierNonVal}(x_\mathsf{u})$ sous la valuation $\rho = [x_\mathsf{u} \to \mathsf{u}]$ si et seulement si $m < n$ et $\mathsf{u} = \mathsf{u}_{m+1}$. En rappelant la définition de $\mathit{ConstrVal}\langle \xi \to \phi \rangle$ nous en déduisons que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $\mathit{ConstrVal}\langle \xi \to \phi \rangle$ si et seulement si $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait la formule $(\mathit{QuantUniv}(x_\mathsf{u}) \wedge \neg(x_\mathsf{u}[\top] \,|\, \neg\phi)) \vee (\mathit{QuantExist}(x_\mathsf{u}) \wedge x_\mathsf{u}[\top] \,|\, \phi)$.

Supposons d'abord que $Q_{m+1} = \exists$. Alors par le point (iii) nous savons que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ ne satisfait pas $\mathit{QuantUniv}(x_\mathsf{u}) \wedge \neg(x_\mathsf{u}[\top] \,|\, \neg\phi)$. Donc nous devons montrer que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $x_\mathsf{u}[\top] \,|\, \phi$ sous la valuation $[x_\mathsf{u} \to \mathsf{u}_{m+1}]$ si et seulement si $\gamma$ satisfait $\exists u_{m+1}.\mathcal{G}''$, où $\mathcal{G}''$ est la formule $Q_{m+2}u_{m+2}.\cdots.Q_n u_n$. Maintenant, par propriétés de la satisfiabilité, $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $x_\mathsf{u}[\top] \,|\, \phi$ sous la valuation $[x_\mathsf{u} \to \mathsf{u}_{m+1}]$ si et seulement si $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ s'écrit comme la composition de deux arbres $s' \uplus s''$ où $s'$ est de la forme $\{\!|\mathsf{u}_{m+1}[s]|\!\}$ pour un arbre $s$ quelconque et $s''$ satisfait $\phi$. Par définition de $t_{\mathcal{G}}$ et par définition de la satisfiabilité, il est facile de voir que les deux seules possibilités pour $s'$ et $s''$ sont

$$\begin{array}{ll} s' = \{\!|e^{\mathsf{faux}}_{m+1}|\!\} & \text{et } s'' = t_{\mathcal{G}}\{\gamma, u_{m+1} \to \mathsf{vrai}\} \\ \text{ou} & \\ s' = \{\!|e^{\mathsf{vrai}}_{m+1}|\!\} & \text{et } s'' = t_{\mathcal{G}}\{\gamma, u_{m+1} \to \mathsf{faux}\}. \end{array} \tag{9.2}$$

La conclusion est alors facile en utilisant l'hypothèse de récurrence sur les arbres $t_{\mathcal{G}}\{\gamma, u_{m+1} \to \mathsf{vrai}\}$ et $t_{\mathcal{G}}\{\gamma, u_{m+1} \to \mathsf{faux}\}$.

Supposons maintenant que $Q_{m+1} = \forall$. Alors par le point (ii) nous savons que $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ ne satisfait pas $\mathit{QuantExist}(x_\mathsf{u}) \wedge x_\mathsf{u}[\top] \,|\, \phi$. Donc nous devons montrer que $\mathsf{u}_0[\{\!|\}] \uplus$

$t_{\mathcal{G}}\{\gamma\}$ satisfait $x_{\mathsf{u}}[\top] \Rrightarrow \xi$ sous la valuation $[x_{\mathsf{u}} \to \mathsf{u}_{m+1}]$ si et seulement si $\gamma$ satisfait $\forall u_{m+1}.\mathcal{G}''$. Maintenant, par propriétés de la satisfiabilité, $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ satisfait $\neg(x_{\mathsf{u}}[\top] \mid \neg\xi)$ sous la valuation $[x_{\mathsf{u}} \to \mathsf{u}_{m+1}]$ si et seulement si pour toute décomposition de $\mathsf{u}_0[\{\!|\}] \uplus t_{\mathcal{G}}\{\gamma\}$ sous la forme de $s' \uplus s''$, si $s'$ est de la forme $\{\!|\mathsf{u}_{m+1}[s]|\!\}$, alors $s''$ satisfait $\phi$. La seule possibilité est pour $s'$ étant de la forme $\{\!|\mathsf{u}_{m+1}[s]|\!\}$ est $s'$ étant $\{\!|e^{\mathsf{faux}}_{m+1}|\!\}$ ou $\{\!|e^{\mathsf{vrai}}_{m+1}|\!\}$ et donc, en utilisant (9.2) nous savons qu'alors $s''$ est égal à $t_{\mathcal{G}}\{\gamma, u_{m+1} \to \mathsf{vrai}\}$ et $t_{\mathcal{G}}\{\gamma, u_{m+1} \to \mathsf{faux}\}$ respectivement. La conclusion n'est alors pas très difficile en utilisant les propriétés de la satisfiabilité et l'hypothèse de récurrence.

□

Comme conséquence immédiate du précédent lemme, du théorème 9.1 statuant que le problème de validité d'une FBQ est PSPACE difficile, et de la proposition 8.18 établissant qu'il existe un algorithme de model checking pour la logique LS polynomial en espace, nous avons

**Théorème 9.15** *Le problème de model checking de la logique LS est PSPACE complet pour sa complexité de données.*

### 9.3.4 Le fragment sans récursion ($\mathrm{LS}_{|\mu}$)

Nous montrons ici que la complexité de données du model checking de la logique $\mathrm{LS}_{|\mu}$ est dur pour tous les niveaux de la hiérarchie polynomiale. Plus précisément, nous montrons que pour tout niveau existentiel $\Sigma^P_n$ de la hiérarchie polynomiale, ils existent une formule **fixe** $\phi^{\exists}_n$ de la logique $\mathrm{LS}_{|\mu}$ et un arbre $t$ tels que le problème de model checking $t \models \phi^{\exists}_n$ est $\Sigma_n$ difficile. De la même façon, pour tout niveau universel $\Pi_n$ de la hiérarchie polynomiale, ils existent une formule **fixe** $\phi^{\forall}_n$ de la logique $\mathrm{LS}_{|\mu}$ et un arbre $t$ tels que le problème de model checking $t \models \phi^{\exists}_n$ est $\Pi^P_n$ difficile. L'arbre $t$ est une adaptation de l'arbre $t_{\mathcal{G}}$ présenté dans la section 9.3.2.

Soient $n, r_1, \ldots, r_n$ des entiers naturels positifs et soit $I \subseteq \mathbb{N} \times \mathbb{N}$ l'ensemble d'indices

$$I = \{(1,1), \ldots, (1,r_1), \ldots, (n,1), \ldots, (n,r_n)\}.$$

Nous considérons la formule booléenne quantifiée

$$\mathcal{G} = Q_1 u_{(1,1)}, \ldots, u_{(1,r_1)}.Q_2 u_{(2,1)}, \ldots, u_{(2,r_2)}.\cdots.Q_n u_{(n,1)}, \ldots, u_{(n,r_n)}.G$$

où la formule propositionnelle $G$ est construite sur l'ensemble de variables $\bigcup_{i\in I}\{u_i\}$. De plus, nous supposons que pour tout $h$ dans $1..n-1$, les quantificateurs $Q_h$ et $Q_{h+1}$ sont différents, càd le degré d'alternance de la formule $\mathcal{G}$ est $n$.

Soient $\mathsf{quant}$, $\mathsf{sat}$, $\mathsf{vv}$, $\mathsf{ff}$, $\mathsf{u}_i$ pour tout $i$ dans $I$ et $\mathsf{q}_h$ pour tout $h$ dans $1..n$ sont des étiquettes dans $\Sigma$ et soit l'arbre

$$t_{\mathcal{G}} = \{\!|e^{\mathsf{vrai}}_{(1,1)}, e^{\mathsf{faux}}_{(1,1)}, \ldots, e^{\mathsf{vrai}}_{(1,r_1)}, e^{\mathsf{faux}}_{(1,r_1)}, \ldots, e^{\mathsf{vrai}}_{(n,1)}, e^{\mathsf{faux}}_{(n,1)}, \ldots, e^{\mathsf{vrai}}_{(n,r_n)}, e^{\mathsf{faux}}_{(n,r_n)}|\!\}$$

où les éléments d'arbres $e^{\mathsf{vrai}}_i$ et $e^{\mathsf{faux}}_i$ pour $(h,j)$ dans $I$ sont définis par :

$$\begin{aligned} e^{\mathsf{vrai}}_{(h,j)} &= \mathsf{u}_{(h,j)}[\{\!|\mathsf{vv}[\{\!|\}], \mathsf{quant}[\{\!|\mathsf{q}_h[\{\!|\}]|\!\}], \mathsf{sat}[s^{\mathsf{vrai}}_{(h,j)}]|\!\}] \\ e^{\mathsf{faux}}_{(h,j)} &= \mathsf{u}_{(h,j)}[\{\!|\mathsf{ff}[\{\!|\}], \mathsf{quant}[\{\!|\mathsf{q}_h[\{\!|\}]|\!\}], \mathsf{sat}[s^{\mathsf{faux}}_{(h,j)}]|\!\}]. \end{aligned}$$

Pour chaque entier $h$ dans $1..n$, soient les formules $\phi_n^{\exists}(h)$ et $\phi_n^{\forall}(h)$ définies récursivement comme :

$$\phi_n^{\exists}(h) = Enlever(h) \,|\, (EstValuée(h) \wedge \phi_n^{\forall}(h+1))$$
$$\phi_n^{\forall}(h) = \neg(Enlever(h) \,|\, (EstValuée(h) \wedge \neg\phi_n^{\exists}(h+1)))$$

où $\phi_n^{\exists}(n+1) = \phi_n^{\forall}(n+1) = Valide'$. Les formules $Enlever(h)$, $EstValuée(h)$ (pour $h$ dans $1..n$) et $Valide'$ sont définies comme suit :

$$\begin{aligned} Enlever(h) &= \forall x_{\mathsf{u}}.(x_{\mathsf{u}}[\top] \,|\, \top \rightarrow x_{\mathsf{u}}[\mathsf{quant}[\mathsf{q}_h[\mathbf{0}]] \,|\, \top] \,|\, \top) \wedge EstValuée(h) \\ EstValuée(h) &= \neg\exists x_{\mathsf{u}}.(x_{\mathsf{u}}[\mathsf{quant}[\mathsf{q}_h[\mathbf{0}]] \,|\, \top] \,|\, x_{\mathsf{u}}[\mathsf{quant}[\mathsf{q}_h[\mathbf{0}]] \,|\, \top] \,|\, \top) \\ Valide' &= \exists x_{\mathsf{c}}.Partout(x_c). \end{aligned}$$

Cet encodage est correct dans le sens où

**Lemme 9.16** *Soit $\mathcal{G}$ une formule booléenne quantifiée de degré alternance $n$ dont le quantificateur le plus externe est $\exists$ (respectivement $\forall$). Alors la formule $\mathcal{G}$ est valide si et seulement si $t_{\mathcal{G}}$ satisfait $\phi_n^{\exists}$ (respectivement $t_{\mathcal{G}}$ satisfait $\phi_n^{\forall}$).*

*Preuve* La preuve de ce lemme est moins détaillée que les preuves des lemmes 9.9 et 9.14. Notamment, nous passons vite sur tous les points qui utilisent des raisonnements similaires à ceux utilisés dans ces deux lemmes et qui ont déjà été présentés en détail.

Soit $\mathcal{G}$ la formule $Q_1 u_{(1,1)\ldots u_{1,r_1}} \cdot \cdots \cdot Q_n u_{(n,1)\ldots u_{n,r_n}} . G$. Nous montrons que pour tout $m$ dans $0..n$ et pour toute valuation $\gamma$ de domaine $\{u_{(1,1)}, \ldots, u_{(1,r_1)}, \ldots, u_{(m,1)}, \ldots, u_{(m,r_m)}\}$ et pour $\mathcal{G}'$ étant la formule $Q_{m+1} u_{(m+1,1)}, \ldots, u_{(m+1,r_{m+1})} \cdot \cdots \cdot Q_n u_{(n,1)}, \ldots, u_{(n,r_n)} . G$, il est vrai que $t_{\mathcal{G}}\{\gamma\}$ satisfait $\phi_n^{Q_{m+1}}(m+1)$ si et seulement si $\gamma$ satisfait $\mathcal{G}'$.

La preuve se fait par récurrence sur $n - m$.

Cas de base: Si $m = n$, alors nous devons montrer que $t_{\mathcal{G}}\{\gamma\}$ satisfait $Valide'$ si et seulement si $\gamma$ satisfait $G$. En rappelant que $Valide' = \exists x_{\mathsf{c}}.Partout(x_{\mathsf{c}})$, ceci est vérifié par le lemme 9.13.

Cas d'induction: On a $m < n$. Montrons d'abord les propriétés suivantes (pour deux valuations $\gamma, \gamma'$ de domaines respectifs $U, U'$, nous notons $\gamma\gamma'$ la valuation de domaine $U \cup U'$ et telle que pour tout $u$ dans $U'$, $\gamma\gamma'(u) = \gamma'(u)$ et pour tout $u$ dans $U \setminus U'$, $\gamma\gamma'(u) = \gamma(u)$) :

(i) pour tous arbres $t', t''$ tels que $t_{\mathcal{G}}\{\gamma\} = t' \uplus t''$, si l'arbre $t'$ satisfait $Enlever(m+1)$ et l'arbre $t''$ satisfait $EstValuée(m+1)$, alors il existe une valuation $\gamma'$ de domaine $\{u_{(m+1,1)}, \ldots, u_{(m+1,r_{m+1})}\}$ telle que $t'' = t_{\mathcal{G}}\{\gamma\gamma'\}$ ;

(ii) pour toute valuation $\gamma'$ de domaine $\{u_{(m+1,1)}, \ldots, u_{(m+1,r_{m+1})}\}$, $t_{\mathcal{G}}\{\gamma\gamma'\}$ satisfait $EstValuée(m+1)$ et si $t'$ est l'arbre tel que $t_{\mathcal{G}}\{\gamma\} = t' \uplus t_{\mathcal{G}}\{\gamma\gamma'\}$, alors $t'$ satisfait $Enlever(m+1)$.

Remarquons d'abord que, par définition de l'arbre $t_{\mathcal{G}}\{\gamma\}$, chacun des arbres $t'$ et $t''$ contient uniquement des éléments d'arbre de la forme $e^{\mathsf{b}}_{(h,j)}$ pour b étant une des valeurs booléennes $\{\mathsf{vrai}, \mathsf{faux}\}$ et $(h,j)$ appartenant à $I$. De plus, pour chaque $j$ dans $1..r_{m+1}$, les éléments d'arbres $e^{\mathsf{vrai}}_{(m+1,j)}$ et $e^{\mathsf{faux}}_{(m+1,j)}$ appartiennent à l'arbre $t'$ ou à l'arbre $t''$. Notons également que, par un raisonnement long mais relativement simple et utilisant uniquement la définition de la satisfiabilité de LS, on peut constater qu'un arbre $t$ composé d'éléments d'arbres de la forme $e^{\mathsf{b}}_i$ satisfait la formule $EstValuée(m+1)$ si et seulement si pour tout $j$ dans $1..r_{m+1}$, les éléments d'arbres $e^{\mathsf{vrai}}_{(m+1,j)}$ et $e^{\mathsf{faux}}_{(m+1,j)}$ ne sont pas tous les deux éléments de $t$. Par un raisonnement semblable, il est facile de se convaincre également qu'un tel arbre $t$ satisfait la formule $Enlever(m+1)$ si

et seulement si $t$ contient uniquement des éléments d'arbres $e^{\mathsf{b}}_{(m+1,j)}$ pour $j$ étant dans $1..r_{m+1}$ et b étant une valeur booléenne. Les propriétés (i) et (ii) sont conséquence facile de ces deux remarques, en utilisant que par définition de $Enlever(m+1)$, si $t'$ satisfait $Enlever(m+1)$ alors $t'$ satisfait $EstValuée(m+1)$ également.

Supposons maintenant que $Q_{m+1}$ est égal à $\exists$. Nous devons montrer que $t_{\mathcal{G}}\{\gamma\}$ satisfait $\phi^{\exists}_n(m+1)$ si et seulement si $\gamma$ satisfait $\exists u_{(m+1,1)}, \ldots, u_{(m+1,r_{m+1})}.\mathcal{G}''$ où $\mathcal{G}''$ est la formule

$$Q_{m+2}u_{(m+2,1)}, \ldots, u_{(m+2,r_{m+2})}. \cdots .Q_n u_{(n,1)}, \ldots, u_{(n,r_n)}.G.$$

Par définition de la satisfiabilité, $t_{\mathcal{G}}\{\gamma\}$ satisfait $\phi^{\exists}_n(m+1)$ si et seulement s'il existe deux arbres $t'$ et $t''$ tels que $t'$ satisfait $Enlever(h)$ et $t''$ satisfait $(EstValuée(h) \wedge \phi^{\forall}_n(m+2))$. En utilisant (i) et (ii), nous pouvons voir que ceci est vrai si et seulement si $t''$ est de la forme $t_{\mathcal{G}}\{\gamma\gamma'\}$ pour une certaine valuation $\gamma'$ de domaine $\{u_{(m+1,1)}, \ldots, u_{(m+1,r_{m+1})}\}$ et $t''$ satisfait $\phi^{\forall}_n(m+2)$. Par hypothèse de récurrence, ceci est vrai si et seulement si $\gamma\gamma'$ satisfait $\mathcal{G}''$. Par définition de la satisfiabilité des formules propositionnelles et par définition de $\mathcal{G}'$, il est facile de voir que ceci est équivalent à $\gamma$ satisfait la formule $\mathcal{G}'$.

Supposons maintenant que $Q_{m+1}$ est égal à $\forall$. Nous devons montrer que pour $t_{\mathcal{G}}\{\gamma\}$ satisfait $\phi^{\exists}_n(m+1)$ si et seulement si $\gamma$ satisfait $\forall u_{(m+1,1)}, \ldots, u_{(m+1),r_{m+1}}.\mathcal{G}''$ où $\mathcal{G}''$ est la formule

$$Q_{m+2}u_{(m+2,1)}, \ldots, u_{(m+2,r_{m+2})}. \cdots .Q_n u_{(n,1)}, \ldots, u_{(n,r_n)}.G.$$

Par définition de la satisfiabilité, $t_{\mathcal{G}}\{\gamma\}$ satisfait $\phi^{\exists}_n(m+1)$ si et seulement si pour toute décomposition de $t_{\mathcal{G}}\{\gamma\}$ sous la forme $t' \uplus t''$, l'une de ces trois conditions est vérifiée : $t'$ ne satisfait pas $Enlever(m+1)$ ou $t''$ ne satisfait pas $EstValuée(m+1)$ ou $t''$ satisfait $\phi^{\exists}_n(m+2)$. Par un raisonnement utilisant la définition de la satisfiabilité des formules et (i) et (ii), ceci est équivalent à ce que pour toute valuation $\gamma'$ de domaine $\{u_{(m+1,1)}, \ldots, u_{(m+1,r_{m+1})}\}$, $t_{\mathcal{G}}\{\gamma\gamma'\}$ satisfait la formule $\phi^{\exists}_n(m+2)$. Par hypothèse de récurrence, ceci est vrai si et seulement si $\gamma\gamma'$ satisfait $\mathcal{G}''$. La conclusion est alors faite en utilisant les définitions de la satisfiabilité des formules LS et des formules propositionnelles.

□

Comme conséquence immédiate du précédent lemme, du théorème 9.2 statuant que le problème de validité pour une FBQ $\mathcal{G}$ de degré d'alternance $n$ est complet pour le $n^{\text{ème}}$ niveau de hierarchie polynomiale existentielle (resp. universelle) si $\mathcal{G}$ est de la forme $\exists u\mathcal{G}'$ (resp. $\forall u\mathcal{G}'$), et de la proposition 8.18 qui établit qu'il existe un algorithme de model checking pour la logique LS polynomial en espace, nous avons

**Théorème 9.17** *Le problème de model checking de la logique $LS_{|\mu}$ est difficile pour tous les niveaux de la hiérarchie polynomiale pour sa complexité de données.*

## 9.4 Comparaison avec une logique spatiale de graphes

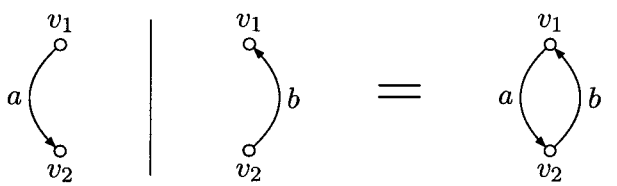
Dans [Cardelli et al., 2002] et [Dawar et al., 2004] est introduite et étudiée une logique spatiale de graphes, inspirée de la logique des ambients, tout comme la logique spatiale d'arbres que nous étudions ici. Dans [Dawar et al., 2004], les auteurs étudient l'expressivité et la comlplexité du model checking et obtiennent des résultats très semblables à ceux exposés dans ce chapitre.

Dans la présente section, nous introduisons brièvement cette logique de graphes et citons les résultats sur la complexité du model checking.

Un graphe est une structure $(V, E, \Sigma, \mathsf{arete})$ où $V$ est un ensemble de nœuds, $E$ est un ensemble d'arêtes, $A$ un ensemble d'étiquettes et arete est une fonction qui à toute arête dans $E$ associe une étiquette de $A$, un nœud source et un nœud destination de $V$ : $\mathsf{arete} : E \rightarrow \Sigma \times V \times V$. On considère également que l'ensemble $V$ des nœuds d'un graphe est un sous ensemble de l'ensemble fixe dénombrable de noms $\mathcal{V}$. Ainsi, deux graphes différents peuvent avoir des nœuds (dont les noms sont) en commun.

L'ensemble des graphes est muni d'une opération de composition, notée $|$, qui correspond intuitivement à l'union disjointe de deux graphes. Si $G_1 = (V_1, E_1, \Sigma_1, \mathsf{arete}_1)$ et $G_2 = (V_2, E_2, \Sigma_2, \mathsf{arete}_2)$ sont deux graphes, alors le graphe $G = G_1 \mid G_2$ est : soit $G = (V, E, \Sigma, \mathsf{arete})$, avec $V = V_1 \cup V_2$, $E = E_1 \uplus E_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$ et $\mathsf{arete} = \mathsf{arete}_1 \uplus \mathsf{arete}_2$, où $\uplus$ désigne l'union disjointe. L'opération de composition sur les arbres peut alors être vue comme cette opération de composition entre graphes ayant une structure d'arbre et possédant un unique nœud en commun : le nœud racine.

**Exemple 9.3** Prenons comme exemple les graphes $G_1 = (\{v_1, v_2\}, \{e\}, \{a\}, \mathsf{arete}_1(e) = (a, v_1, v_2))$ et $G_2 = (\{v_2, v_1\}, \{e'\}, \{b\}, \mathsf{arete}_1(e') = (b, v_1, v_2))$. C'est à dire, le graphe $G_1$ a deux nœuds (de noms) $v_1, v_2$ et une seule arête allant de $v_1$ à $v_2$ étiquetée par $a$. Le graphe $G_2$ a deux nœuds (de noms) $v_1, v_2$ et une arête allant de $v_2$ à $v_1$ étiquetée par $b$. La composition de ces deux graphes et le graphe à deux nœuds (de noms) $v_1, v_2$ et deux arêtes, l'une allant de $v_1$ à $v_2$ étiquetée par $a$, et l'autre allant de $v_2$ à $v_1$ étiquetée par $b$.




Pour la logique de graphes, on considère un ensemble dénombrable de variables d'étiquettes $\mathcal{X}_\Sigma$ et un ensemble dénombrable de variables de nœuds (ou de noms) $\mathcal{X}_V$. Les éléments de $\mathcal{X}_\Sigma$ sont notés $x$, les éléments de $\mathcal{X}_V$ sont notés $w$. Une formule $\psi$ de la logique de graphes est

définie récursivement par :

$$
\begin{array}{lll}
\psi ::= & \mathbf{0} & \\
& \top & \\
& \eta(\theta, \theta') & \text{pour } \eta \in \Sigma \cup \mathcal{X}_\Sigma \text{ et } \theta, \theta' \in \mathcal{V} \cup \mathcal{X}_V \\
& \theta = \theta' & \text{pour } \theta, \theta' \in \mathcal{V} \cup \mathcal{X}_V \\
& \eta = \eta' & \text{pour } \eta, \eta' \in \Sigma \cup \mathcal{X}_\Sigma \\
& \psi \mid \psi' & \\
& \psi \vee \psi' & \\
& \neg\psi & \\
& \exists w.\psi & \\
& \exists x.\psi &
\end{array}
$$

Nous donnons une définition informelle de l'interprétation des formules. La formule $\mathbf{0}$ est satisfaite par les graphes sans arêtes, la formule $\top$ est satisfaite par tout graphe ; les formules $\theta = \theta'$ et $\eta = \eta'$ permettent de tester l'égalité de deux nœuds (noms) et de deux étiquettes respectivement, les formules à quantification existentielle $\exists w.\psi$ et $\exists x.\psi$ et les opérateurs booléens $\psi \vee \psi'$ et $\neg\psi$ sont interprétés comme habituellement. La formule de composition $\psi \mid \psi'$ est satisfaite par les graphes $G \mid G'$ avec $G$ satisfaisant $\psi$ et $G'$ satisfaisant $\psi'$. Finalement, la formule $\eta(\theta, \theta')$ est satisfaite par les graphes possédant une seule arête allant du nœud $\theta$ au nœud $\theta'$ étiquetée par $\eta$. Par exemple, la formule $\exists x.\exists w.x(w, w)$ et satisfaite par les graphes ayant comme unique arête une boucle sur un de leurs nœuds (les graphes satisfaisant cette formule peuvent avoir plus d'un nœud).

Les résultats suivants d'expressivité et de complexité du model checking de la logique de graphes sont établis dans [Dawar et al., 2004] :

- la complexité combinée du model checking est PSPACE complet ;
- la complexité de donnée est dure pour tous les niveaux de la hiérarchie polynomiale, càd pour tout niveau $k$ existentiel (resp. universel) de PH, il existe une formule $\psi_k$ tel que le model checking de cette formule est difficile pour le $k^{\text{ème}}$ niveau existentiel (resp. universel) de la hiérarchie polynomiale.

La logique de graphes peut être enrichie avec un opérateur de plus petit point fixe :

$$\mu\xi.\psi$$

est un nouveau constructeur de formule. L'interprétation des formules à point fixe est similaire que l'interprétation du plus petit point fixe pour la logique spatiale d'arbres, càd on considère $\psi$ comme une fonction de l'ensemble des ensembles de graphes dans lui même, et $\mu\xi.\psi$ est le plus petit point fixe de cette fonction. En ce qui concerne cette extension de la logique de graphes :

- la complexité combinée du model checking est PSPACE complet ;
- la complexité de données du model checking est PSPACE complet.

## Conclusion

Nous avons établi différentes bornes de complexité pour la logique spatiale et ses fragments. Tout d'abord, nous nous sommes intéressés à la complexité combinée et nous avons constaté que

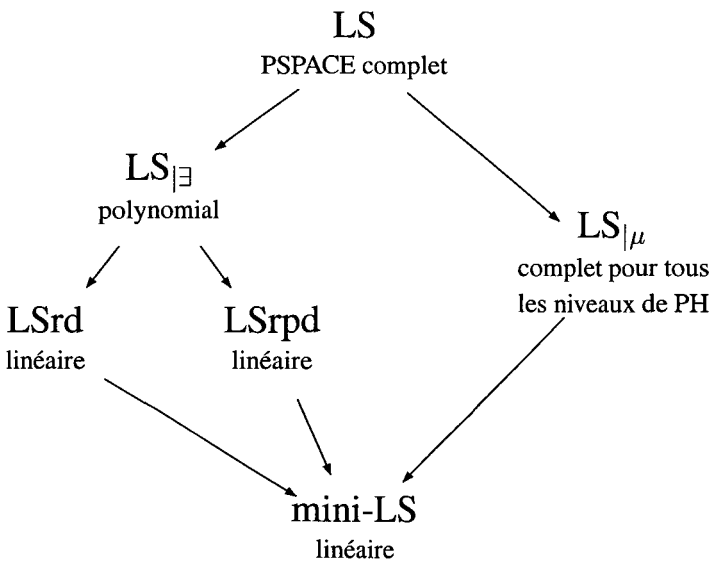



FIG. 1 – Bornes inférieures de la complexité de données pour les fragments de la logique spatiale. Les flèches indiquent la relation d'inclusion pour les fragments : par exemple, la flèche entre les fragments LS et $LS_{|\mu}$ indique que le fragment $LS_{|\mu}$ est inclus dans LS. La complexité combinée est PSPACE complet pour tous ces fragments.

même pour le plus petit fragment contenant les opérateurs spatiaux – mini-LS – la complexité combinée du model checking est PSPACE-difficile. Ceci combiné avec les résultats du chapitre précédent nous a permis de conclure que le model checking de la logique spatiale est un problème PSPACE-complet (pour sa complexité combinée). Ce résultat nous paraît très important : d'une part, il montre que l'algorithme de model checking proposé dans le chapitre précédent a la complexité optimale que l'on peut espérer[5]. D'autre part ce résultat montre que la complexité du model checking est inhérente à l'opérateur de composition et avoir des complexités théoriques élevées est inévitable. Cependant, ce résultat n'est pas entièrement décourageant pour la possibilité d'utiliser la logique spatiale en pratique. En effet, pour montrer cette borne de complexité, nous avons dû utiliser l'imbrication de composition et de négation (dans l'opérateur $\models\!\!\Rightarrow$), ce qui donne facilement des formules inintelligibles. Nous pouvons supposer sans prendre beaucoup de risques que dans les exemples utiles en pratique, cette imbrication sera rare et d'une profondeur bornée.

Nous avons ensuite étudié la complexité de données du model checking. Nous avons établi que celle-ci est PSPACE-complet pour la logique spatiale sans restrictions et complet pour tous les niveaux de la hiérarchie polynomiale pour la logique sans récursion. Pour différents fragments de la logique sans quantification, la complexité de données est linéaire (pour LSrd et LSrpd), ou polynomiale ($LS_{|\exists}$) pour un problème de model checking fixé. Ces différentes bornes inférieures sont résumées sur la figure 1. Les résultats sur la complexité de données sont plus significatifs que les résultats sur la complexité combinée en ce qui concerne la praticabilité du model checking, puisqu'il s'agit là d'une complexité dans la taille de l'arbre, qui peut être très grande.

[5]Il s'agit de la complexité asymptotique. En particulier, l'algorithme tel qu'il est donné peut éventuellement être amélioré, restant cependant dans la même classe de complexité au pire des cas.

# Conclusion

L'objectif de cette thèse était de faire une étude systématique de l'expressivité de la logique spatiale et de ses fragments, en s'intéressant en particulier au problème de satisfiabilité et au problème de model checking. L'étude de ces deux problèmes a été motivée par la possibilité d'utiliser la logique spatiale pour la modélisation ou l'interrogation de documents semi structurés.

En ce qui concerne la satisfiabilité, il était déjà connu que ce problème est indécidable en présence de quantification. Nous avons établi que c'est aussi le cas pour la logique sans quantification. Nous avons également identifié deux fragments de la logique, LSrd et LSrpd, pour lesquels le problème de satisfiabilité est décidable. Ces fragments ont été obtenus par une restriction du mécanisme de récursion de la logique, et restent très expressifs. De plus, ces deux fragments sont équivalents à deux logiques d'arbres non ordonnés : la logique MSO et son extension PMSO.

L'étude du problème de model checking nous a permis de proposer un algorithme pour le model checking s'exécutant en espace polynomial. Nous avons également établi des bornes inférieures pour la complexité du problème de model checking de différents fragments de la logique spatiale, en s'intéressant à la complexité de données et à la complexité combinée. Pour la logique sans restriction, la complexité du model checking est PSPACE difficile, y compris pour la complexité de données. Par contre, en ce qui concerne la logique sans quantification, la complexité de donnée du model checking est polynomiale.

Comme résultat annexe à l'étude de l'expressivité de la logique spatiale, nous avons introduit les automates à contraintes numériques qui capturent son fragment quantification.

*Perspectives* Cette thèse représente essentiellement un travail théorique, même si certaines des motivations venaient d'une possibilité d'appliquer la logique spatiale à la modélisation et l'interrogation de documents semi structurés. Une première perspective serait d'étudier l'applicabilité de ces résultats en pratique.

A première vue, les résultats de complexité ne sont pas très encourageants pour une utilisation de la logique spatiale comme un langage de requêtes pour données semi structurées : dans la pratique, même un algorithme polynomial de degré deux pourrait s'avérer impraticable, puisque les données qu'on considère sont parfois de très grande taille. D'autre part, ces résultats ne sont pas vraiment surprenants ; utiliser des arbres non ordonnés est dans tous les cas plus coûteux qu'utiliser des arbres ordonnés. Néanmoins, ignorer l'ordre dans les arbres peut être une solution simple de plusieurs problèmes qui existent en pratique, comme par exemple la prise en compte des attributs dans un document XML, ou bien pour exprimer des propriétés de comp-

tage. De ce fait, nous pensons que si la logique spatiale devait être utilisée en pratique, ce serait comme outil pour traiter certains cas particuliers et non comme langage de requêtes en soi. Des solutions mixtes proposant la possibilité d'utiliser ou non l'ordre des fils d'un arbre ont été proposées dans [Seidl et al., 2004] et [Dal Zilio and Lugiez, 2003], le second de ces travaux utilise la logique spatiale lorsqu'il s'agit d'ignorer l'ordre. L'étude systématique de l'expressivité de la logique spatiale que nous avons faite ici peut alors permettre de faire le bon compromis entre expressivité et complexité.

Certaines questions théoriques peuvent également être étudiée. Par exemple, il serait intéressant de connaître l'expressivité de la logique spatiale si celle-ci est interprétées sur des arbres ordonnés, où l'opérateur | serait un opérateur associatif mais non commutatif.[6] Le lien entre la logique spatiale et d'autres formalismes peut également être étendu. Nous avons vu que les logiques MSO et PMSO correspondent à des fragments syntaxiques de la logiques spatiale et à des classes d'automates à contraintes numériques. Par ailleurs, nous La logique MSO à comptage (CMSO) [Courcelle, 1990b] est également une extension de MSO par des formules exprimant contraintes de comptage du type $C^k_{\mathrm{mod}n}(U)$, satisfaite lorsque le cardinal de l'ensemble $U$ est congru à $k$ modulo $n$. Nous savons que la logique CMSO correspond à une classe d'automates à contraintes numériques [Boneva and Talbot, 2005] et que son expressivité est capturée par PMSO, mais nous ne savons pas si elle peut être caractérisée par un fragment de la logique spatiale. D'autres formalismes, telle la logique propositionnelle dynamique, sont également capturés par la logique spatiale et les liens exacts pourraient être étudiés.

[6] Dans ce cas, il ne s'agirait plus d'une logique spatiale.

# Bibliographie


[Arnold and Niwinski, 2001] Arnold, A. and Niwinski, D. (2001). *Rudiments of $\mu$-Calculus*. North-Holland.

[Autebert et al., 1997] Autebert, J.-M., Berstel, J., and Boasson, L. (1997). *Context-Free Languages and Push-Down Automata*, volume 1 of *Handbook of Formal Languages*, chapter 3, pages 111–174. Springer.

[Barcelo and Libkin, 2005] Barcelo, P. and Libkin, L. (2005). Temporal Logics over Unranked Trees. In *Proceedings 20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 31–40. Elsevier.

[Boneva and Talbot, 2004] Boneva, I. and Talbot, J.-M. (2004). On Complexity of Model-Checking for the TQL Logic. In *3rd IFIP International Conference on Theoretical Computer Science (TCS2004)*, pages 381–394.

[Boneva and Talbot, 2005] Boneva, I. and Talbot, J. M. (2005). Automata and Logics for Unranked and Unordered Trees. In *16th International Conference on Rewriting Techniques and Applications*, volume 3467 of *LNCS*, pages 500–515. Springer.

[Boneva et al., 2005] Boneva, I., Talbot, J. M., and Tison, S. (2005). Expressiveness of a spatial logic for trees. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005)*. IEEE Comp. Soc. Press.

[Boudet and Comon, 1996] Boudet, A. and Comon, H. (1996). Diophantine equations, Presburger arithmetic and finite automata. In *Proc. Coll. on Trees in Algebra and Programming (CAAP'96)*, volume 1059 of *LNCS*. Springer Verlag.

[Büchi, 1960] Büchi, J. R. (1960). Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundl. Math.*, 6 :66–92.

[Calcagno et al., 2003] Calcagno, C., Cardelli, L., and Gordon, A. D. (2003). Deciding Validity in Spatial Logic for Trees. In *Proceedings of the 2003 ACM SIGPLAN international workshop on Types in languages design and implementation*, pages 62–73.

[Cardelli et al., 2002] Cardelli, L., Gardner, P., and Ghelli, G. (2002). A spatial logic for querying graphs. In *29th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *LNCS*, pages 597–610. Springer.

[Cardelli and Ghelli, 2001] Cardelli, L. and Ghelli, G. (2001). A Query Language Based on the Ambient Logic. In *European Symposium on Programming (ESOP'01)*, volume 2028 of *LNCS*, pages 1–22. Springer.

[Cardelli and Ghelli, 2004] Cardelli, L. and Ghelli, G. (2004). TQL : A Query Language for Semistructured Data Based on the Ambient Logic. *Mathematical Structures in Computer Science*, 14 :285–327.

[Cardelli and Gordon, 2000a] Cardelli, L. and Gordon, A. D. (2000a). Anytime, Anywhere : Modal Logics for Mobile Ambients. In *27th ACM Symposium on Principles of Programming Languages (POPL'00)*, pages 365–377. Springer.

[Cardelli and Gordon, 2000b] Cardelli, L. and Gordon, A. D. (2000b). Mobile Ambients. *TCS*, 240 :177–213.

[Charatonik et al., 2001] Charatonik, W., Dal Zilio, S., Gordon, A. D., Mukhopadhyay, S., and Talbot, J.-M. (2001). The Complexity of Model Checking Mobile Ambients. In *Foundations of Software Science and Computation Structures (FoSSaCS'01)*, volume 2030 of *LNCS*, pages 152–167. Springer.

[Charatonik et al., 2003] Charatonik, W., Dal Zilio, S., Gordon, A. D., Mukhopadhyayd, S., and Talbot, J.-M. (2003). Model checking mobile ambients. *TCS*, 308(3) :277–331.

[Charatonik and Talbot, 2001] Charatonik, W. and Talbot, J.-M. (2001). The Decidability of Model Checking Mobile Ambients. In *Computer Science Logic (CSL'01)*, volume 2142 of *LNCS*, pages 339–354. Springer.

[Colcombet, 2002] Colcombet, T. (2002). Rewriting in the partial algebra of typed terms modulo AC. In *Electronic Notes in Theoretical Computer Science*, volume 68. Elsevier Science Publishers.

[Comon et al., 1997] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. (1997). Tree automata techniques and applications. Available on : `http://www.grappa.univ-lille3.fr/tata`. release October, 1rst 2002.

[Conforti et al., 2002a] Conforti, G., Ferrara, O., and Ghelli, G. (2002a). Tql algebra and its implementation (extended abstract). In *Proc. of IFIP International Conference on Theoretical Computer Science (IFIP TCS)*. Kluwer.

[Conforti et al., 2002b] Conforti, G., Ghelli, G., Albano, A., Colazzo, D., Manghi, P., and Sartiani, C. (2002b). he query language TQL. In *Proc. of the 5th International Workshop on the Web and Databases (WebDB)*.

[Courcelle, 1990a] Courcelle, B. (1990a). Graph Rewriting : An Algebraic and Logic Approach. In *Handbook of Theoretical Computer Science*, volume B, chapter 5, pages 193–242. Elsevier.

[Courcelle, 1990b] Courcelle, B. (1990b). The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *IC*, 85(1) :12–75.

[Dal Zilio, 2001] Dal Zilio, S. (2001). Fixed points in the ambient logic. In *Proc. of FICS 2001 – 3rd Workshop on Fixed Points in Computer Science*.

[Dal Zilio and Lugiez, 2002] Dal Zilio, S. and Lugiez, D. (2002). XML Schema, Tree Logic and Sheaves Automata. Research Report 4631, INRIA.

[Dal Zilio and Lugiez, 2003] Dal Zilio, S. and Lugiez, D. (2003). XML Schema, Tree Logic and Sheaves Automata. In *Rewriting Techniques and Applications, 14th International Conference, RTA 2003*, volume 2706 of *LNCS*, pages 246–263. Springer.

[Dal Zilio et al., 2004] Dal Zilio, S., Lugiez, D., and Meyssonnier, C. (2004). A logic you can count on. In *Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 135–146. ACM.

[Dawar et al., 2004] Dawar, A., Gardner, P., and Ghelli, G. (2004). Expressiveness and complexity of graph logic. Technical report, Imperial College.

[Doner, 1970] Doner, J. (1970). Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4 :406–451.

[Elgot, 1961] Elgot, C. C. (1961). Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98 :21–52.

[Gaubert and Giua, 1999] Gaubert, S. and Giua, A. (1999). Petri net languages and infinite subsets of $\mathbb{N}^m$. *Journal of Computer and System Sciences*, 3 :373–391.

[Goubault-Larrecq and Verma, 2002] Goubault-Larrecq, J. and Verma, K. N. (2002). Alternating Two-way AC-Tree Automata. Research Report LSV, Laboratoire Spécification et Vérification.

[Klaedtke, 2004] Klaedtke, F. (2004). On the automata size for Presburger arithmetic. In *Logic in Computer Science (LICS 2004)*, pages 110–119. IEEE.

[Libkin, 2005] Libkin, L. (2005). Logics over unranked trees : an overview. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'05)*, number 3580 in LNCS, pages 35–50.

[Lugiez, 2003] Lugiez, D. (2003). Counting and equality corstraints for multitree automata. In *Foundations of Software Science and Computational Structures : 6th International Conference, FOSSACS 2003*, volume 2620 of *LNCS*, pages 328 – 342. Springer.

[Lugiez and Dal Zilio, 2002] Lugiez, D. and Dal Zilio, S. (2002). Multitree Automata, Presburger's Constraints and Tree Logics. Technical Report 08-2002, Labortoire d'Informactique Fondamentale de Marseille.

[Lugiez and Moysset, 1994] Lugiez, D. and Moysset, J. L. (1994). Tree Automata Help One To Solve Equational Formulae In AC-Theories. *Journal of Symbolic Computation*, 18(4) :297–318.

[Mader, 1997] Mader, A. (1997). *"Verification of Modal Properties Using Boolean Equation Systems"*. Bertz Verlag.

[Minsky, 1961] Minsky, M. L. (1961). Recursive insolvability of Post's problem of "tag" ard other topics in the theory of turing machines. In *Annals of Mathematics, Second Series*, volume 74, pages 437–455.

[Neven and Schwentick, 2002] Neven, F. and Schwentick, T. (2002). Query automata over finite trees. *Theoretical Computer Science*, 275 :633–674.

[Niehren and Podelski, 1993] Niehren, J. and Podelski, A. (1993). Feature automata and recognizable sets of feature trees. In *TAPSOFT*, pages 356–375.

[Ohsaki, 2001] Ohsaki, H. (2001). Beyond Regularity : Equational Tree Automata for Associative and Commutative Theories. In *15th International Conference of the European Association for Computer Science Logic (CSL2001)*, number 2142 in LNCS, pages 539–553. Springer Verlag.

[Ohsaki and Takai, 2002] Ohsaki, H. and Takai, T. (2002). Decidability and Closure Properties of Equational Tree Languages. In *13th International Conference on Rewriting Techniques and Applications (RTA2002)*, number 2378 in LNCS, pages 114–128. Springer Verlag.

[Papadimitriou, 1995] Papadimitriou, C. H. (1995). *Computational complexity*. Addison Wesley Longman.

[Seidl et al., 2003] Seidl, H., Schwentick, T., and Muscholl, A. (2003). Numerical Document Queries. In *Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 155–166. ACM.

[Seidl et al., 2004] Seidl, H., Schwentick, T., Muscholl, A., and Habermehl, P. (2004). Counting in Trees for Free. In *Automata, Languages and Programming : 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *LNCS*, pages 1136–1149. Springer Verlag.

[Stirling and Walker, 1991] Stirling, C. and Walker, D. (1991). Local model checking in the modal mu-calculus. *TCS*, 89 :161–177.

[Stockmeyer, 1976] Stockmeyer, L. J. (1976). The Polynomial-time Hierarchy. *TCS*, 3(1) :1–22.

[Thatcher and Wright, 1968] Thatcher, J. W. and Wright, J. B. (1968). Generalized finite automata with an application to a decision problem of second-order logic. *Mathematical System Theory*, 2 :57–82.

[Trakhtenbrot, 1950] Trakhtenbrot, B. A. (1950). The impossibility of an algorithm for the decision problem for finite models. *Doklad Akademii Nauk*, SSR(70) :569–572.

[Verma, 2003] Verma, K. N. (2003). *Automates d'arbres bidirectionnels modulo théories équationnelles (Two-Way Equational Tree Automata).* PhD thesis, ENS Cachan.

[Winskel, 1991] Winskel, G. (1991). A note on model checking the modal $\nu$-calculus. *TCS*, 83 :157–167.

[Wolper and Boigelot, 2000] Wolper, P. and Boigelot, B. (2000). On the Construction of Automata from Linear Arithmetic Constraints. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS 2000)*, volume 1785 of *LNCS*, pages 1–19. Springer Verlag.

# Annexe A

# Preuves omises dans le document

## A.1 Preuve du théorème 3.1

Nous devons montrer que pour tout automate à contraintes numériques sans étoile $A$ il existe une formule MSO close $\psi_A$ telle que $\mathcal{L}(A)$ est l'ensemble des arbres satisfaisant $\psi_A$ et inversement, pour toute formule de MSO close $\psi$, il existe un automate à contraintes numériques $A_\psi$ tel que $\mathcal{L}(A)$ est l'ensemble des arbres satisfaisant $\psi$. Comme nous l'avons déjà mentionné, la preuve est classique et suit le mêmes idées utilisées pour montrer l'équivalence de la logique MSO sur les mots et les automates finis de mots, par exemple (voir par exemple [Comon et al., 1997] pour plus de détails).

Commençons par rappeler ou fixer quelques notations et faits connus. Pour un arbre $t$, $S^t$ est la $\sigma$-structure associée à cet arbre. Nous noterons $E_t$ l'ensemble des arêtes de l'arbre $t$. L'union et l'intersection d'ensembles sont MSO-définissables, ainsi que le test qu'un ensemble est vide. De ce fait, nous utilisons les notations standard pour exprimer ces propriétés ; par exemple, les notations $U = \varnothing$, $U = U_1 \cap U_2$, etc. sont autorisées dans une formule MSO.

Montrons d'abord comment, pour un automate $A = (\Sigma, Q, \Delta, F)$ donné, on construit la formule $\psi_A$. Soit $Q = \{q_1, \ldots, q_k\}$. Nous construisons la formule $\psi'_A(U_{q_1}, \ldots, U_{q_k})$ de variables libres $U_{q_1}, \ldots, U_{q_k}$ qui sera satisfaite par la valuation $\gamma$ et l'arbre $t$ si les ensembles $\gamma(U_{q_1}), \ldots, \gamma(U_{q_k})$ sont les ensembles d'arêtes de l'arbre $t$ étiquetées $q_1, \ldots, q_k$ par une exécution acceptante $r$ de l'automate $A$ sur l'arbre $t$. Formellement, la formule $\psi'_A(U_{q_1}, \ldots, U_{q_k})$ est telle que que pour tout arbre $t$ et pour toute valuation $\gamma$ des variables $U_{q_1}, \ldots, U_{q_k}$, on a $S^t, \gamma \models \psi'_A(U_{q_1}, \ldots, U_{q_k})$ si et seulement si

(a) $\gamma(U_{q_1}), \ldots, \gamma(U_{q_k})$ est un partitionnement de l'ensemble des arêtes de l'arbre $t$ et

(b) $r_\gamma$ est une exécution acceptante de l'automate $A$ sur l'arbre $t$,

où $r_\gamma$ est l'application de $E_t$ dans $q$ définie par :

pour toute arête $e$ de $t$, $r_\gamma(e) = q$ où $q$ est l'état tel que $e$ appartient à l'ensemble $\gamma(U_q)$

Nous commençons par définir différentes relations qui nous servirons pour définir la formule $\psi_A$.

- La formule $\mathsf{partition}(U_1, \ldots, U_n)$ exprime le fait que le ensembles $U_1, \ldots, U_n$ forment une partition de l'ensemble des arêtes de l'arbre

$$\mathsf{partition}(U_1, \ldots, U_k) := \forall u. u \in (U_1 \cup \cdots \cup U_k) \wedge \bigwedge_{i \in 1..k, j \in i+1..k} U_i \cap U_j = \varnothing$$

- Pour tout entier naturel $n$, la formule $\mathsf{card}_{=n}(U)$ exprime le fait que le cardinal de l'ensemble $U$ est $n$

$$\begin{aligned}\mathsf{card}_{=n}(U) := \exists u_1, \ldots, u_n. & \bigwedge_{i \in 1..n} u_i \in U \\ \wedge & \bigwedge_{i \in 1..n, j \in i+1..n} u_i \neq u_j \\ \wedge \forall u. ( & \bigwedge_{i \in 1..n} u \neq u_i) \rightarrow u \notin U\end{aligned}$$

si $n$ est strictement positif et

$$\mathsf{card}_{=0}(U) := U = \varnothing$$

- Pour tout entier naturel $n$, la formule $\mathsf{card}_{\geq n}(U)$ exprime le fait que la cardinal de l'ensemble $U$ est supérieure ou égale à $n$.

$$\begin{aligned}\mathsf{card}_{\geq n}(U) = \exists u_1, \ldots, u_n. & \bigwedge_{i \in 1..n} u_i \in U \\ \wedge & \bigwedge_{i \in 1..n, j \in i+1..n} u_i \neq u_j\end{aligned}$$

- Pour tout ensemble linéaire $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_j)$ inclus dans $\mathbb{N}^Q$, la formule $\mathsf{contr}_{\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_j)}(U_1, \ldots, U_k)$ exprime le fait que le multiensemble $(q_1 \rightarrow |U_1|, \ldots, q_k \rightarrow |U_k|)$ appartient à l'ensemble $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_j)$ (ici $|U_i|$ désigne la cardinal de l'ensemble $U_i$)

$$\mathsf{contr}_{\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_j)}(U_1, \ldots, U_k) := \bigwedge_{i \in 1..k} \begin{cases} \mathsf{card}_{=\mathbf{b}(i)}(U_i) & \text{si } 1_{q_i} \text{ n'est pas parmi } \mathbf{p}_1, \ldots, \mathbf{p}_j \\ \mathsf{card}_{\geq \mathbf{b}(i)}(U_i) & \text{si } 1_{q_i} \text{ est parmi } \mathbf{p}_1, \ldots, \mathbf{p}_j \end{cases}$$

- Pour tout ensemble de multiensembles sans étoile $D \subseteq \mathbb{N}^Q$, la formule $\mathsf{contr}_D(U_1, \ldots, U_k)$ exprime le fait que le multiensemble $(q_1 \rightarrow |U_1|, \ldots, q_k \rightarrow |U_k|)$ appartient à l'ensemble semilinéaire $D$, représenté comme une union d'ensembles linéaires (ici $|U_i|$ désigne la cardinal de l'ensemble $U_i$).

$$\mathsf{contr}_D(U_1, \ldots, U_k) := \bigwedge_{L \text{ composante linéaire de } D} \mathsf{contr}_L(U_1, \ldots, U_k)$$

- Pour tout ensemble fini ou co-fini d'étiquettes $\alpha$, la formule $\mathsf{étiquette}_\alpha(u)$ exprime le fait que l'arête $u$ est étiquetée par $a \in \alpha$.

$$\mathsf{étiquette}_\alpha(u) := \begin{cases} \bigvee_{a \in \alpha} \mathsf{etq}_a(u) & \text{si } \alpha \text{ fini} \\ \bigwedge_{a \notin \alpha} \neg \mathsf{etq}_a(u) \end{cases}$$

- La formule successeurs$(u, U)$ est satisfaite si $U$ est l'ensemble des arêtes successeurs de l'arête $u$.

$$\mathsf{successeurs}(u, U) = \forall u'.u' \in U \leftrightarrow u < u'$$

- Pour tout ensemble de multiensembles sans étoile $D \subseteq \mathbb{N}^Q$, la formule $\mathsf{contr\_succ}_D(u, U_1, \ldots, U_k)$ exprime le fait que les ensembles $U_1, \ldots, U_k$ limités aux successeurs de l'arête $u$ satisfont la contrainte $D$.

$$\mathsf{contr\_succ}_D(u, U_1, \ldots, U_k) := \exists U.\mathsf{successeurs}(u, U) \wedge \mathsf{contr}_D(U_1 \cap U, \ldots, U_k \cap U)$$

- La formule racines$(U)$ exprime le fait que l'ensemble d'arêtes $U$ est l'ensemble des arêtes successeurs de la racine de l'arbre

$$\mathsf{racines}(U) := u \in U \leftrightarrow \neg\exists u'.u' < u$$

Soit maintenant la formule $\psi'_A$ définie par

$$\begin{aligned}
&\psi'_A(U_{q_1}, \ldots, U_{q_k}) := \\
&\quad \mathsf{partition}(U_{q_1}, \ldots, U_{q_k}) \wedge \\
&\quad \forall u. \bigwedge_{i \in 1..k} \left( (u \in U_{q_i}) \rightarrow \bigvee_{(q_i, \alpha, D) \in \Delta} \mathsf{étiquette}_\alpha(u) \wedge \mathsf{contr_succ}_D(u, U_{q_1}, \ldots, U_{q_n}) \right) \wedge \\
&\quad \exists U.\mathsf{racines}(U) \wedge \mathsf{contr}_F(U_{q_1} \cap U, \ldots, U_{q_k} \cap U)
\end{aligned}$$

La formule $\psi'_A$ telle que pour tout arbre $t$ et pour toute valuation des variables $U_{q_1}, \ldots, U_{q_k}$ $\gamma$, $S^t$ satisfait $\psi'_A(U_{q_1}, \ldots, U_{q_k})$ sous la valuation $\gamma$ si et seulement si :

- $q_1, \ldots, q_k$ sont des états de l'automate $A$,
- les ensembles $\gamma(U_{q_1}), \ldots, \gamma(U_{q_k})$ forment une partition de l'ensemble des arêtes de $t$ ;
- soit $r_\gamma$ l'application qui à tout $e$ arête de $t$ associe l'état $q_i$ tel que $e \in \gamma(U_{q_i})$, alors $r_\gamma$ est une exécution acceptante de $A$ sur $t$.

Alors, la formule $\psi_A$ doit juste vérifier qu'une telle valuation $\gamma$ $\gamma$ existe (càd l'exécution acceptante $r_\gamma$ existe)

$$\psi_A := \exists U_{q_1}, \ldots, U_{q_k}.\psi'_A(U_{q_1}, \ldots, U_{q_k}).$$

Nous devons montrer à présent que pour toute formule close $\psi$ de la logique MSO, il existe un ACN sans étoile $A_\psi$ tel que le langage de $A_\psi$ est exactement l'ensemble des arbres modèles de la formule $\psi$.

Considérons la logique $\mathrm{MSO}_0$, équivalente à MSO, qui n'utilise que des variables du second ordre. Une formule $\mathrm{MSO}_0$ est définie par la syntaxe

$$\psi ::= \mathsf{etq}_a(U) \text{ (pour } a \in \Sigma\text{)} \mid U < U' \mid U \subseteq U' \mid \mathsf{singleton}(U) \mid \psi \vee \psi' \mid \neg\psi \mid \exists U.\psi$$

Etant donné une structure $S$ et une valuation $\gamma$ des variables du second ordre, $S$ satisfait la formule $\psi$ sous la valuation $\gamma$ si

- $\psi$ est $\mathsf{etq}_a(U)$ et pour toute arête $e$ dans $\gamma(U)$, $\mathsf{etq}_a(e)$ est vrai dans $S$ ;
- $\psi$ est $U < U'$, l'ensemble $\gamma(U)$ est à un unique élément, soit $\{e\}$, l'ensemble $\gamma(U')$ est non vide et pour toute arête $e'$ dans $\gamma(U')$, $e < e'$ est vrai dans $S$ ;

- $\psi$ est $U \subseteq U'$ et $\gamma(U) \subseteq \gamma(U')$ ;
- $\psi$ est $\mathsf{singleton}(U)$ et $\gamma(U)$ est un ensemble à un unique élément ;
- $\psi$ est de la forme $\psi \vee \psi'$ ou $\neg\psi$ ou $\exists U.\psi$, dans quel cas l'interprétation est comme pour MSO.

Pour toute formule MSO $\psi'$, il existe une formule $\mathrm{MSO}_0$ $\psi''$ équivalente ; de plus la transformation de $\psi'$ en $\psi''$ est syntaxique. Donc, il suffit de faire la preuve pour $\psi$ étant une formule $\mathrm{MSO}_0$.

Nous allons construire l'automate $A_\psi$ par induction sur la structure de la formule $\psi$ (de la logique $\mathrm{MSO}_0$). Dans ce cas, nous devons considérer le cas où $\psi$ est une formule non close. Pour prendre en compte la formule $\psi(U_1, \ldots, U_k)$ à $k$ variables libres, nous allons définir la satisfiabilité pour une telle formule, par des arbres étiquetés par des éléments de $\Sigma \times \{0,1\}^k$. Soit $t'$ un arbre sur $\Sigma \times \{0,1\}^k$ ; alors $S^{t'}$ satisfait la formule $\psi(U_1, \ldots, U_k)$ si $S^t$ satisfait la formule $\psi(U_1, \ldots, U_k)$ pour la valuation $\gamma$, où :

- $t$ est l'arbre $h(t')$, où $h$ est l'homomrphisme d'étiquettes défini par l'application $h_{\Sigma\times\{0,1\}^k}$ : $\Sigma \times \{0,1\}^k \to \Sigma$ qui à tout $(a, b_1, \ldots, b_k)$ associe $a$ (càd, $t$ est obtenu par renommage des arêtes de $t'$ en projetant sur la première composante) et
- $\gamma$ est la valuation qui à $U_i$ associe l'ensemble des arêtes de l'arbre $t$ (et donc de l'arbre $t'$, puisqu'il y a une bijection entre les arêtes de $t$ et $t'$)

$$\{e \mid \mathsf{etq}(e) = (a, b_1, \ldots, b_k) \text{ dans } t' \text{ et } b_i = 1\}.$$

L'idée utilisée habituellement dans ce type de preuves est de construire d'abord l'automate $A_\psi$ pour $\psi$ étant une formule atomique, et d'utiliser la clôture des ACN par les opérations booléennes et par projection pour l'induction. Cette idée ne peut pas être appliquée telle quelle dans notre cas : il s'avère que l'ensemble d'arbres modèles de la formule $\mathsf{etq}_a(U)$ n'est pas définissable par un ACN sans étoile. En effet, cette formule est satisfaite par tout arbre dans $\mathcal{A}_{\Sigma\times\{0,1\}}$ tel qu'aucune des arêtes de l'arbre n'est étiquetée par $(a', 1)$ pour $a' \neq a$ (càd si une arête appartient à $U$, alors son étiquette doit être $a$). Donc, l'automate $A_\phi$ doit pouvoir distinguer une arête dont l'étiquette est dans l'ensemble $\omega = \{(a', 1) \mid a' \neq a\}$ d'une arête dont l'étiquette est dans $\Sigma \times \{0,1\} \setminus \omega$. Or, l'ensemble $\omega$ n'est ni fini, ni cofini. Donc, aucun ACN sans étoile ne pourrait faire la distinction entre ces deux arêtes. Pour résoudre cette difficulté, nous allons considérer une extension des ACN sans étoile.

Considérons l'ensemble produit $\Sigma \times \{0,1\}^n$, pour un certain entier naturel $n$. Soit $\pm_n$ l'ensemble des sous ensembles de $\Sigma \times \{0,1\}^n$ qui obtenus par des les opérations ensemblistes de base – union, intersection et complémentaire – à partir d'ensembles de la forme $\alpha \times B_1 \times \cdots \times B_n$ inclus dans $\Sigma \times \{0,1\}^n$ et tels que $\alpha$ est fini ou cofini. Formellement,

- pour tout $\alpha \subseteq \Sigma$ fini ou cofini, et pour tous $B_1, \ldots, B_n \subseteq \{0,1\}$, $\alpha \times B_1 \times \cdots \times B_n$ appartient à $\pm_n$ ;
- si $\omega, \omega'$ appartiennent à $\pm_n$, alors $\omega \cap \omega'$, $\omega \cup \omega'$ et $\pm_n \setminus \omega$ appartiennent à $\pm_a$.

Pour tout entier naturel $n$, un *$n$-automate à contraintes numériques étendu sans étoile ($n$-ACNe sans étoile)*, est un quadruplet $(\pm_n, Q, \Delta, F)$, où $Q$ est un ensemble fini d'états, $\Delta \subseteq Q \times \pm_n \times \wp(\mathbb{N}^Q)$ est la relation de transition et $F \subseteq \mathbb{N}^Q$ est la condition d'acceptation. De plus, pour toute transition $(q, \omega, D)$ dans $\Delta$, $D$ est un ensemble de multiensembles sans étoile.

Notons que si les 0-ACNe sans étoile sont les ACN sans étoile.

Nous allons montrer que pour toute formule $\mathrm{MSO}_0$ $\psi(U_1, \ldots, U_n)$ à $n$ variables libres, il existe un $n$-ACNe sans étoile $A_\psi$ qui reconnaît exactement les arbres modèles de la formule $\psi$. Dans ce cas, si $\psi$ est une formule close, l'automate $A_\psi$ est un ACN sans étoile.

La preuve se fait par induction sur la structure de la formule $\psi$. D'abord, le cas de base, pour $\psi$ étant une formule atomique. Si $\psi$ est la formule $\mathrm{etq}_a(U)$, alors $A_\psi = (\pm_1, Q, \Delta, F)$, où $Q = \{q_{\mathrm{ok}}, q_{\mathrm{err}}\}$, la relation de transition $\Delta$ est donnée par

$$\Delta = \left\{ \begin{array}{lll} (q_{\mathrm{ok}}, & \{(a,1)\}, & \{(q_{\mathrm{ok}} \mapsto m, q_{\mathrm{err}} \mapsto 0) \mid m \in \mathbb{N}\}), \\ (q_{\mathrm{ok}}, & \Sigma \times \{0\}, & \{(q_{\mathrm{ok}} \mapsto m, q_{\mathrm{err}} \mapsto 0) \mid m \in \mathbb{N}\}), \\ (q_{\mathrm{err}}, & (\Sigma \setminus \{a\}) \times \{1\}, & \mathbb{N}^Q), \\ (q_{\mathrm{err}}, & \Sigma \times \{0,1\}, & \{(q_{\mathrm{ok}} \mapsto m, q_{\mathrm{err}} \mapsto m') \mid m \in \mathbb{N}, m' \geq 1\}) \ \} \end{array} \right\}$$

Finalement, la condition d'acceptation est $\{(q_{\mathrm{ok}} \mapsto m, 0) \mid m \in \mathbb{N}\}$. Intuitivement, l'état $q_{\mathrm{ok}}$ définit les éléments d'arbres dans lesquels toutes les arêtes satisfont la condition donnée par la formule $\psi$, et $q_{\mathrm{err}}$ définit les éléments d'arbres qui contiennent au moins une étiquette qui contredit la condition donnée par $\psi$.

Si $\psi$ est la formule atomique $U < U'$, rappelons que $\psi$ est satisfaite seulement si $U$ est un élément à un unique élément, soit $e$, et que les arêtes dans $U'$ sont toutes des arêtes successeurs de l'arête $e$. L'automate $A_\psi$ est alors $A_\psi = (\pm_2, Q, \Delta, F)$, où $Q = \{q_-, q', q, q^-\}$. Intuitivement, $q_-$ est l'état qui sera associé aux arêtes qui ne sont ni dans $U$ ni dans $U'$ et qui se trouvent ou bien « au dessous » de $e$, ou bien sur un chemin de l'arbre qui ne passe pas par $e$ ; $q^-$ est l'état associé aux arêtes se trouvant « au dessus » de $e$ ; $q'$ est l'état associé aux arêtes dans $U'$ et $q$ est l'état associé à l'arête $e$. L'ensemble de transitions $\Delta$ est alors défini par :

$$\Delta = \left\{ \begin{array}{lll} (q_-, & \Sigma \times \{0\} \times \{0\}, & \{(q_- \mapsto m, q' \mapsto 0, q \mapsto 0, q^- \mapsto 0) \mid m \in \mathbb{N}\}), \\ (q', & \Sigma \times \{0\} \times \{1\}, & \{(q_- \mapsto m, q' \mapsto 0, q \mapsto 0, q^- \mapsto 0) \mid m \in \mathbb{N}\}), \\ (q, & \Sigma \times \{1\} \times \{0\}, & \{(q_- \mapsto m, q' \mapsto m', q \mapsto 0, q^- \mapsto 0) \mid m \in \mathbb{N}, m' \geq 1\}), \\ (q^-, & \Sigma \times \{0\} \times \{0\}, & \{(q_- \mapsto m, q' \mapsto 0, q \mapsto m_1, q^- \mapsto m_2) \mid m \in \mathbb{N}, m_1 + m_2 = 1\}) \end{array} \right\}$$

Finalement, la condition d'acceptation est $F = \{(q_- \mapsto m, q' \mapsto 0, q \mapsto m_1, q^- \mapsto m_2) \mid m \in \mathbb{N}, m_1 + m_2 = 1\}$.

Si $\psi$ est la formule atomique $U \subseteq U'$, il faut vérifier que toute arête dans $U$ est aussi dans $U'$. L'automate $A_\psi$ est $A_\psi = (\pm_2, Q, \Delta, F)$, avec $Q = \{q, q', q_{\mathrm{err}}\}$. Intuitivement, dans une exécution ascendante, $q$ est associé aux arêtes appartenant à $U$ et à $U'$, $q'$ est associé aux arêtes qui n'appartiennent pas à $U$ et $q_{\mathrm{err}}$ est associé aux arêtes qui appartiennent à $U$ mais pas à $U'$ et est propagé vers le haut. Alors, la relation de transition $\Delta$ est définie par

$$\Delta = \left\{ \begin{array}{lll} (q, & \Sigma \times \{1\} \times \{1\}, & \{(q \mapsto m, q' \mapsto m', q_{\mathrm{err}} \mapsto 0) \mid m, m' \in \mathbb{N}\}), \\ (q', & \Sigma \times \{0\} \times \{0,1\}, & \{(q \mapsto m, q' \mapsto m', q_{\mathrm{err}} \mapsto 0) \mid m, m' \in \mathbb{N}\}), \\ (q_{\mathrm{err}}, & \Sigma \times \{1\} \times \{0\}, & \{(q \mapsto m, q' \mapsto m', q_{\mathrm{err}} \mapsto 0) \mid m, m' \in \mathbb{N}\}), \\ (q_{\mathrm{err}}, & \Sigma \times \{0,1\} \times \{0,1\}, & \{(q \mapsto m, q' \mapsto m', q_{\mathrm{err}} \mapsto m'') \mid m, m' \in \mathbb{N}, m'' \geq 1\}) \end{array} \right\}$$

La condition d'acceptation est l'ensemble $F = \{(q \mapsto m, q' \mapsto m', q_{\mathrm{err}} \mapsto 0) \mid m, m' \in \mathbb{N}\}$.

Si $\psi$ est la formule atomique $\mathrm{singleton}(U)$, il faut vérifier qu'une seule arête de l'arbre est dans l'ensemble $U$. L'automate $A_\psi$ est $A_\psi = (\pm_1, Q, \Delta, F)$, avec $Q = \{q, q', q_{\mathrm{ok}}, q_{\mathrm{err}}\}$.

Intuitivement, dans une exécution ascendante de l'automate, l'état $q$ est associé à une première arête appartenant à $U$, $q'$ est l'état associé aux arêtes n'appartenant pas à $U$ et si aucune arête dans $U$ n'a encore été rencontrée, $q_{\text{ok}}$ est associé aux arêtes définissant un élément d'arbre contenant une et une seule arête de $U$ et $q_{\text{err}}$ est associé aux arêtes définissant un élément d'arbre dans lequel il y a au moins deux arêtes appartenant à $U$. La relation de transition est alors

$$\Delta = \left\{ \begin{array}{lll} (q', & \Sigma \times \{0\}, & \{(q \mapsto 0, q' \mapsto m', q_{\text{ok}} \mapsto 0, q_{\text{err}} \mapsto 0) \mid m' \in \mathbb{N}\}), \\ (q, & \Sigma \times \{1\}, & \{(q \mapsto 0, q' \mapsto m', q_{\text{ok}} \mapsto 0, q_{\text{err}} \mapsto 0) \mid m' \in \mathbb{N}\}), \\ (q_{\text{ok}}, & \Sigma \times \{0\}, & \{(q \mapsto m, q' \mapsto m', q_{\text{ok}} \mapsto m_{\text{ok}}, q_{\text{err}} \mapsto 0) \mid m' \in \mathbb{N}, m + m_{\text{ok}} = 1\}), \\ (q_{\text{err}}, & \Sigma \times \{1\}, & \{(q \mapsto m, q' \mapsto m', q_{\text{ok}} \mapsto m_{\text{ok}}, q_{\text{err}} \mapsto 0) \mid m' \in \mathbb{N}, m + m_{\text{ok}} \neq 0\}), \\ (q_{\text{err}}, & \Sigma \times \{0,1\}, & \{(q \mapsto m, q' \mapsto m', q_{\text{ok}} \mapsto m_{\text{ok}}, q_{\text{err}} \mapsto m_{\text{err}}) \mid m', m, m_{\text{ok}} \in \mathbb{N}, m_{\text{err}} \neq 0\}) \end{array} \right\}$$

La condition d'acceptation est l'ensemble $F = \{(q \mapsto m, q' \mapsto m', q_{\text{ok}} \mapsto m_{\text{ok}}, q_{\text{err}} \mapsto 0) \mid m' \in \mathbb{N}, m + m_{\text{ok}} = 1\}$.

Pour chacun des automates $A_\psi$ définis jusque présent, avec $\psi$ étant une formule atomique, il est facile de voir que les ensembles de multiensembles apparaissant dans les transitions sont des ensembles sans étoile.

Pour le cas d'induction, il nous faut construire l'automate $A_\psi$ pour $\psi$ étant une formule non atomique. Si $\psi$ est la formule $\psi' \vee \psi''$ (respectivement $\neg\psi'$), alors l'automate $A_\psi$ est l'automate union des automates $A_{\psi'}$ et $A_{\psi''}$ (respectivement l'automate complémentaire de $A_{\psi'}$). Finalement, si $\psi$ est la formule $\exists U.\psi'$, supposons que la formule $\psi'$ est une formule à $n$ variables libres, dont $U$. Alors $\psi$ est une formule à $n-1$ variables libres. Soit $A_{\psi'}$ l'automate $A_{\psi'} = (\pm_n, Q, \Delta', F)$. Alors $A_\psi$ est l'automate $A_\psi = (\pm_{n-1}, Q, \Delta, F)$, où la relation de transition $\Delta$ est obtenue à partir de $\Delta'$ en projetant chacun des ensembles d'étiquettes $\omega \subseteq \Sigma \times \{0,1\}^n$ sur les $n-1$ composantes qui ne correspondent pas à $U$ :

$$\Delta' = \{(q, \mathsf{prj}_U(\omega), D) \mid (q, \omega, D) \in \Delta\}.$$

## A.2 Preuves omises dans le chapitre 4

### Preuve du lemme 4.21

Nous devons montrer que la relation de réécriture $\rightarrow_s$ termine. De plus, si $\mathcal{S}_t$ est une forme normale de $\mathcal{S}_i$ pour $\rightarrow_s$, alors

(i) $\mathcal{S}_t$ est simplifié ;

(ii) $\mathcal{S}_t$ est équivalent à $\mathcal{S}_i$ sur $\{\mathsf{Vars}(\mathcal{S}_i)\}$ ;

Montrons d'abord que le système de réécriture termine. Pour chaque terme fonctionnel $f$, soit $p(f)$ l'entier naturel défini par :

- si $f$ est un terme simple, alors $p(f) = 0$ ;
- sinon, récursivement sur la structure de $f$ : $p(\alpha[f']) = 1 + p(f')$ et $p(f' \circ f'') = 1 + p(f') + p(f'')$ pour étant $\circ$ un des opérateurs $|, \|, \vee, \wedge$.

Pour un système d'équations $\mathcal{S}$, soit

$$p(\mathcal{S}) = \sum_{\xi \stackrel{\kappa}{=} f \text{ équation dans } \mathcal{S}} p(f).$$

Alors $\mathcal{S}$ est un système simplifié si et seulement si $p(\mathcal{S}) = 0$. Pour montrer que le système de réécriture termine, il suffit de montrer que si $\mathcal{S}_l \rightarrow_s \mathcal{S}_r$, alors $p(\mathcal{S}_l) > p(\mathcal{S}_r)$. En effet, dans la réduction $\mathcal{S}, l, \mathcal{S}' \rightarrow_s \mathcal{S}, r, \mathcal{S}'$, les quantités $p(\mathcal{S})$ et $p(\mathcal{S}')$ ne changent pas, et pour chacune des règles de réduction $l \rightarrow_s r$, le fait que $p(l) > p(r)$ suit immédiatement des définitions.

Montrons maintenant que $\mathcal{S}_t$ est simplifié (point (i)). Il suffit de montrer que pour tout système d'équations $\mathcal{S}$, si $p(\mathcal{S}) > 0$, alors au moins une règle de réduction peut être appliquée à $\mathcal{S}$. Soit $\xi \stackrel{\kappa}{=} f$ une équation dans $\mathcal{S}$ telle que $p(f) > 0$. En fonction de la forme de $f$ : si $f$ est $\alpha[f']$ et $f'$, alors $f'$ n'est pas une variable et la règle (Red $\alpha$) s'applique ; si $f$ est $f' \circ f''$ pour $\circ$ étant un opérateur binaire, alors au moins un parmi $f'$ et $f''$ n'est pas une variable et une des règles (Red $\circ$ $i$), pour $i$ étant 1,2 ou 3, s'applique.

La preuve du point (ii) se fait par récurrence sur le nombre de pas de réduction utilisés pour obtenir $\mathcal{S}_t$ à partir de $\mathcal{S}_i$. Soit $\mathcal{S}_l \rightarrow_s \mathcal{S}_r$ avec $\mathcal{S}_l = \mathcal{S}, l, \mathcal{S}'$ et $\mathcal{S}_r = \mathcal{S}, r, \mathcal{S}'$, et soit $l$ le système $\xi \stackrel{\kappa}{=} f$. Notons d'abord que $\{\mathsf{Vars}(\mathcal{S}_l)\} \subseteq \{\mathsf{Vars}(\mathcal{S}_r)\}$. Alors il suffit de montrer que $\mathcal{S}_l$ et $\mathcal{S}_r$ sont équivalents sur l'ensemble de variables $\mathsf{Vars}(\mathcal{S}_l)$, et donc, d'après la forme de $\mathcal{S}_l$ et $\mathcal{S}_r$, de montrer que la solution pour la variable $\xi$ est la même dans les systèmes $\mathcal{S}_l$ et $\mathcal{S}_r$. La preuve se fait suivant la règle de réduction appliquée. Le principe étant similaire pour tous les cas, nous considérons ici uniquement la règle (Red | 3). C'est à dire,

$$\mathcal{S}_l = \mathcal{S}, \xi \stackrel{\kappa}{=} f' \,|\, f'', \mathcal{S}'$$
$$\mathcal{S}_r = \mathcal{S}, \xi' \stackrel{\kappa}{=} f',\ \xi'' \stackrel{\kappa}{=} f'',\ \xi \stackrel{\kappa}{=} \xi' \,|\, \xi'', \mathcal{S}'$$

et il faut montrer $Sol_{\mathcal{T}}(\mathcal{S}_l, \xi) = Sol_{\mathcal{T}}(\mathcal{S}_r, \xi)$. Par la proposition 4.5, nous savons que le système d'équations

$$\mathcal{S},\ \xi' \stackrel{\kappa}{=} f',\ \xi'' \stackrel{\kappa}{=} f'',\ \xi \stackrel{\kappa}{=} f' \,|\, f'',\ \mathcal{S}' \tag{A.1}$$

a la même solution que $\mathcal{S}_r$. Par définition des règles de réécriture, dans le système A.1 les variables $\xi'$ et $\xi''$ n'apparaissent nulle part en partie droite d'équation et donc les équations correspondantes peuvent être éliminées sans modifier la solution pour les variables restantes. Or, l'élimination de ces deux équations nous donne exactement $\mathcal{S}_l$.

## Preuve du lemme 4.28

Nous devons montrer que pour tous ensembles d'arbres $T, S$, les égalités ci-dessous sont satisfaites.

$$T|^{\mathcal{T}}S = \begin{cases} T\int^{\mathcal{T}}S & \text{si } \{\!|\}\!\} \notin T, \{\!|\}\!\} \notin S \\ T\int^{\mathcal{T}}S \vee (S \wedge \overline{\mathbf{0}}) & \text{si } \{\!|\}\!\} \in T, \{\!|\}\!\} \notin S \\ T\int^{\mathcal{T}}S \vee (T \wedge \overline{\mathbf{0}}) & \text{si } \{\!|\}\!\} \notin T, \{\!|\}\!\} \in S \\ T\int^{\mathcal{T}}S \vee (S \wedge \overline{\mathbf{0}}) \vee (T \wedge \overline{\mathbf{0}}) \vee \mathbf{0} & \text{si } \{\!|\}\!\} \in T, \{\!|\}\!\} \in S \end{cases}$$

$$T\|^{\mathcal{T}}S = \begin{cases} T\rfloor\!\lceil^{\mathcal{T}}S \wedge (T \vee \mathbf{0}) \wedge (S \vee \mathbf{0}) \wedge \overline{\mathbf{0}} & \text{si } \{\!|\}\!\} \notin T, \{\!|\}\!\} \notin S \\ T\rfloor\!\lceil^{\mathcal{T}}S \wedge (T \vee \mathbf{0}) & \text{si } \{\!|\}\!\} \in T, \{\!|\}\!\} \notin S \\ T\rfloor\!\lceil^{\mathcal{T}}S \wedge (S \vee \mathbf{0}) & \text{si } \{\!|\}\!\} \notin T, \{\!|\}\!\} \in S \\ T\rfloor\!\lceil^{\mathcal{T}}S & \text{si } \{\!|\}\!\} \in T, \{\!|\}\!\} \in S \end{cases}$$

Montrons d'abord que

(i) pour tous ensembles d'arbres $T, S$, on a $T|^{\mathcal{T}}(S \vee \mathbf{0}) = T|^{\mathcal{T}}S \vee T$.

La preuve est facile à partir de la définition de $|^{\mathcal{T}}$. En effet,

$$\begin{aligned} & T|^{\mathcal{T}}(S \vee \mathbf{0}) && \text{(par définition)} \\ &= \{t \uplus s \mid t \in T \wedge s \in (S \cup \{\{\!|\}\!\})\} \\ &= \{t \uplus s \mid (t \in T \wedge s \in S) \vee (t \in T \wedge s = \{\!|\}\!\})\} \\ &= \{t \uplus s \mid t \in T \wedge s \in S\} \cup \{t \uplus \{\!|\}\!\} \mid t \in T\} && \text{(par définition)} \\ &= T|^{\mathcal{T}}S \vee T. \end{aligned}$$

Par dualité, il nous avons également que

(ii) pour tous ensembles d'arbres $T$ et $S$, $T\|^{\mathcal{T}}(S \wedge \overline{\mathbf{0}}) = T\|^{\mathcal{T}}S \wedge T$.

En effet, il est facile de voir que (ici $\complement$ désigne le complémentaire dans l'ensemble $\mathcal{A}_\Sigma$) $T\|^{\mathcal{T}}(S \wedge \overline{\mathbf{0}}) = \complement(\complement T|^{\mathcal{T}}\complement(S \wedge \overline{\mathbf{0}}))$, ce qui est égal à $\complement(\complement T|^{\mathcal{T}}(\complement S \vee \mathbf{0}))$ et, par la propriété montrée précédemment, est égal à $\complement(\complement T|^{\mathcal{T}}\complement S \vee \complement T)$ et donc à $T\|^{\mathcal{T}}S \wedge T$.

Évidemment, par commutativité de l'opérateur $\uplus$, nous avons également $(T \vee \mathbf{0})|^{\mathcal{T}}S = T|^{\mathcal{T}}S \vee S$ et $(T \wedge \overline{\mathbf{0}})\|^{\mathcal{T}}S = T\|^{\mathcal{T}}S \wedge S$, ceci pour tous ensembles d'arbres $T$ et $S$.

Maintenant les égalités de la figure 4 sont faciles à prouver. Tout d'abord, d'après la remarque 4.22, $T|^{\mathcal{T}}S = T\int^{\mathcal{T}}S$ lorsque $\{\!|\}\!\} \notin T$ et $\{\!|\}\!\} \notin S$. De la même façon, $T\|^{\mathcal{T}}S = T\rfloor\!\lceil^{\mathcal{T}}S$ lorsque $\{\!|\}\!\} \in T$ et $\{\!|\}\!\} \in S$. Pour les autres égalités, nous utilisons (i) et (ii). Nous faisons la preuve pour deux cas, les autres étant similaires. Soit $\{\!|\}\!\} \in T$ et $\{\!|\}\!\} \in S$. Alors $T|^{\mathcal{T}}S = ((T \wedge \overline{\mathbf{0}}) \vee \mathbf{0})|^{\mathcal{T}}((S \wedge \overline{\mathbf{0}}) \vee \mathbf{0})$, ce qui par (i) est égal à $((T \wedge \overline{\mathbf{0}}) \vee \mathbf{0})|^{\mathcal{T}}(S \wedge \overline{\mathbf{0}}) \vee ((T \wedge \overline{\mathbf{0}}) \vee \mathbf{0})$, qui par (i) est égal à $(T \wedge \overline{\mathbf{0}})|^{\mathcal{T}}(S \wedge \overline{\mathbf{0}}) \vee (T \wedge \overline{\mathbf{0}}) \vee (S \wedge \overline{\mathbf{0}}) \vee \mathbf{0}$ et la conclusion suit par la remarque 4.22. Soit maintenant $\{\!|\}\!\} \in T$ et $\{\!|\}\!\} \notin S$. Alors $T\|^{\mathcal{T}}S = (T \vee \mathbf{0})\|^{\mathcal{T}}((S \vee \mathbf{0}) \wedge \overline{\mathbf{0}})$, ce qui par (ii) est égal à $T\|^{\mathcal{T}}S = (T \vee \mathbf{0})\|^{\mathcal{T}}(S \vee \mathbf{0}) \wedge (T \vee \mathbf{0})$, qui est égal à $(T \vee \mathbf{0})\|^{\mathcal{T}}(S \vee \mathbf{0}) \wedge (T \vee \mathbf{0})$ et encore une fois la conclusion suit de la remarque 4.22.

## Preuve du lemme 4.29

Nous devons montrer que si $\mathcal{S}$ un système d'équations sur $fonct(\tau', \mathcal{X}_r)$, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, on peut décider si $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$.

Soit Bool le treillis complet $(\{0,1\}, \vee, \wedge)$ muni de la relation d'ordre $0 < 1$, appelé également algèbre de Boole, et soit $\pi$ est la signature $\{\top, \bot, \vee, \wedge\}$. L'interprétation $\mathcal{B}$ de $\pi$ sur le treillis Bool est l'interprétation usuelle.

Pour tout système d'équations $\mathcal{S}$ sur $fixpt(\tau', \mathcal{X}_r)$, soit $\mathcal{S}^b$ le système d'équations sur $fixpt(\pi, \mathcal{X}_r)$ obtenu en replaçant dans $\mathcal{S}$ toute équation $\xi \overset{\kappa}{=} f$ par l'équation $\xi \overset{\kappa}{=} f^b$, où pour tout terme $f$ dans $fixpt(\tau', \mathcal{X}_r)$, le terme $f^b$ est défini récursivement sur la structure de $f$ par :

- pour toute variable $\xi$, $\xi^b = \xi$,
- $\top^b = \mathbf{0}^b = \top$,
- pour tout terme $f_1$, $\bot^b = \overline{\mathbf{0}}^b = \alpha[f_1]^b = \bot$,
- $(f_1 \vee f_2)^b = (f_1 \,\|\, f_2)^b = f_1^b \vee f_2^b$,
- $(f_1 \wedge f_2)^b = (f_1 \,|\, f_2)^b = f_1^b \wedge f_2^b$,
- pour tous termes $f_1, f_2$, $(f_1 \lceil f_2)^b = \bot$ et $(f_1 \rfloor\!\lceil f_2)^b = \top$ et
- $(\kappa\xi.f)^b = \kappa\xi.f^b$.

Nous allons montrer que pour tout système d'équations $\mathcal{S}$ sur $fixpt(\tau, \mathcal{X}_r)$ et pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ si et seulement si $Sol_{\mathcal{B}}(\mathcal{S}^b, \xi) = 1$. Il est bien connu que la solution de tout système d'équations sur $fixpt(\pi, \mathcal{X}_r)$ est effectivement calculable, donc c'est suffisant pour la preuve du lemme.

Commençons par les résultats intermédiaires énoncés dans les faits A.1 et A.2 ci-dessous.

**Fait A.1** Pour tout terme $b$ dans $fixpt(\pi, \mathcal{X}_r)$ et toute valuation $\gamma$ des variables libres de $b$,

(i) $\bigwedge\{h \in \{0,1\} \mid [\![b]\!]_{\gamma[\xi \to h]} \leq h\} = 1$ si et seulement si $[\![b]\!]_{\gamma[\xi \to 0]} = 1$ et

(ii) $\bigvee\{h \in \{0,1\} \mid h \leq [\![b]\!]_{\gamma[\xi \to h]}\} = 1$ si et seulement si $[\![b]\!]_{\gamma[\xi \to 1]} = 1$.

La preuve est presque immédiate des définitions. Pour (i), $\bigwedge\{h \in \{0,1\} \mid [\![b]\!]_{\gamma[\xi \to h]} \leq h\} = 1$ si et seulement si $[\![b]\!]_{\gamma[\xi \to 0]} \not\leq 0$, si et seulement si $[\![b]\!]_{\gamma[\xi \to 0]} = 1$. Pour (ii), $\bigvee\{h \in \{0,1\} \mid h \leq [\![b]\!]_{\gamma[\xi \to h]}\} = 1$ si et seulement si $1 \leq [\![b]\!]_{\gamma[\xi \to 1]}$, si et seulement si $[\![b]\!]_{\delta[\xi \to 1]} = 1$. Ceci termine la preuve du fait A.1.

**Fait A.2** Soit $f$ un terme dans $fixpt(\tau', \mathcal{X}_r)$ et soit $\delta$ une valuation de domaine comprenant au moins les variables libres de $f$, soit $\gamma^b$ la valuation de même domaine que $\delta$ et qui à toute variable $\xi \in dom(\delta)$ associe 1 si $\{\!|\}$ $\in \delta(\xi)$ et 0 sinon. Alors $\{\!|\}$ $\in [\![f]\!]_\delta$ si et seulement si $[\![f^b]\!]_{\gamma^b} = 1$.

La preuve de cette propriété se fait par récurrence sur la structure de $f$. Si $f$ est une variable ou un terme parmi $\top, \bot, \mathbf{0}, \overline{\mathbf{0}}$, la preuve est immédiate de la définition de $f^b$. Si $f = \alpha[f_1]$, alors par définition pour toute valuation $\delta$, $\{\!|\}$ $\notin [\![f]\!]_\delta$, et par définition $f^b = 0$, donc $[\![f^b]\!]_{\gamma^b} = 0$. Si $f$ est le terme $f_1 \vee f_2$, alors par définition $\{\!|\}$ $\in [\![f]\!]_\delta$ si et seulement si $\{\!|\}$ $\in [\![f_1]\!]_\delta$ ou $\{\!|\}$ $\in [\![f_2]\!]_\delta$, si et seulement si (par hypothèse de récurrence) $[\![f_1^b]\!]_{\gamma^b} = 1$ ou $[\![f_2^b]\!]_{\gamma^b} = 1$, si et seulement si (par définition de l'interprétation de $\pi$) $1 = [\![f_1^b \vee f_2^b]\!]_\delta = [\![(f_1 \vee f_2)^b]\!]_\delta$. Si $f$ est le terme $f_1 \wedge f_2$, la preuve est similaire au cas précédent. Si $f$ est le terme $f_1 \,|\, f_2$, il suffit de remarquer que $\{\!|\}$ $\in [\![f_1 \,|\, f_2]\!]_\delta$ si et seulement si $\{\!|\}$ $\in [\![f_1]\!]_\delta$ et $\{\!|\}$ $\in [\![f_2]\!]_\delta$, ensuite on utilise les mêmes arguments que pour les cas précédents. Si $f$ est le terme $f_1 \,\|\, f_2$, il suffit de remarquer que $\{\!|\}$ $\in [\![f_1 \,\|\, f_2]\!]_\delta$ si et seulement si $\{\!|\}$ $\in [\![f_1]\!]_\delta$ ou $\{\!|\}$ $\in [\![f_2]\!]_\delta$. Ensuite la preuve est similaire

au cas précédents. Pour $f$ étant le terme $f_1 \int f_2$, respectivement le terme $f_1 \rceil\!\int f_2$, il suffit de remarquer pour tous ensembles d'arbres $S_1, S_2$, $\{\!|\}$ n'appartient pas à $S_1 \int S_2$, respectivement $\{\!|\}$ appartient à $S_1 \rceil\!\int S_2$. Finalement, montrons la propriété si $f$ est le terme $\kappa\xi.f_1$ pour $\kappa = \mu$ ou $\kappa = \nu$. D'abord pour $\kappa = \mu$. Par définition, $[\![\mu\xi.f_1]\!]_\delta = \bigcap\{S \subseteq \mathcal{A}_\Sigma \mid [\![f_1]\!]_{\delta[\xi \to S]} \subseteq S\}$ et $[\![\mu\xi.f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}} = \bigwedge\{h \in \{0,1\} \mid [\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to h]} \leq h\}$. Par le fait A.1, il suffit de montrer que

(a) $\{\!|\} \in \bigcap\{S \subseteq \mathcal{A}_\Sigma \mid S \supseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$ si et seulement si $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 0]} = 1$.

Supposons d'abord $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 0]} = 1$. Alors, par hypothèse de récurrence du fait A.2 nous savons que pour tout ensemble d'arbres $S$, si $\{\!|\} \notin S$, alors $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 0]} = 1$ si et seulement si $\{\!|\} \in [\![f_1]\!]_{\delta[\xi \to S]}$ ; donc d'après l'hypothèse, si $\{\!|\} \notin S$, alors $\{\!|\} \in [\![f_1]\!]_{\delta[\xi \to S]}$. Par monotonie de la fonction $S \mapsto [\![f_1]\!]_{\delta[\xi \to S]}$ nous déduisons que pour tout ensemble $S$, $\{\!|\} \in [\![f_1]\!]_{\delta[\xi \to S]}$, et donc naturellement $\{\!|\} \in \bigcap\{S \subseteq \mathcal{A}_\Sigma \mid S \supseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$. Supposons maintenant que $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 0]} = 0$. Alors, par hypothèse de récurrence du fait A.2 nous savons que pour tout ensemble d'arbres $S$, si $\{\!|\} \notin S$, alors $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 0]} = 0$ si et seulement si $\{\!|\} \notin [\![f_1]\!]_{\delta[\xi \to S]}$ ; donc d'après l'hypothèse,

($\star$) si $\{\!|\} \notin S$, alors $\{\!|\} \notin [\![f_1]\!]_{\delta[\xi \to S]}$.

Soit maintenant $S$ un ensemble d'arbres tel que $[\![f_1]\!]_{\delta[\xi \to S]} \subseteq S$. Alors, par monotonie de la fonction $S \mapsto [\![f_1]\!]_{\delta[\xi \to S]}$ nous avons que $[\![f_1]\!]_{\delta[\xi \to S \setminus \{\{\!|\}\!\}]} \subseteq S$. Or, par ($\star$) nous avons que $\{\!|\} \notin [\![f_1]\!]_{\delta[\xi \to S \setminus \{\{\!|\}\!\}]}$, et donc $[\![f_1]\!]_{\delta[\xi \to S \setminus \{\{\!|\}\!\}]} \subseteq S \setminus \{\{\!|\}\!\}$. Ceci implique naturellement que $\{\!|\} \notin \bigcap\{S \subseteq \mathcal{A}_\Sigma \mid S \supseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$ et termine la preuve de (a).

Maintenant, pour $\kappa = \nu$, par définition $[\![\nu\xi.f_1]\!]_\delta = \bigcup\{S \subseteq \mathcal{A}_\Sigma \mid S \subseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$ et $[\![\nu\xi.f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}} = \bigvee\{h \in \{0,1\} \mid h \leq [\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to h]}\}$. Par le fait A.1, il suffit de montrer que

(b) $\{\!|\} \in \bigcup\{S \subseteq \mathcal{A}_\Sigma \mid S \subseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$ si et seulement si $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 1]} = 1$.

Supposons d'abord $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 1]} = 1$. Alors, par hypothèse de récurrence du fait A.2 nous savons que pour tout ensemble d'arbres $S$, si $\{\!|\} \in S$, alors $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 1]} = 1$ si et seulement si $\{\!|\} \in [\![f_1]\!]_{\delta[\xi \to S]}$, donc de l'hypothèse nous déduisons que si $\{\!|\} \in S$, alors $\{\!|\} \in [\![f_1]\!]_{\delta[\xi \to S]}$. En particulier, $\{\{\!|\}\} \subseteq [\![f_1]\!]_{\delta[\xi \to \{\{\!|\}\!\}]}$, et donc $\{\!|\} \in \bigcup\{S \subseteq \mathcal{A}_\Sigma \mid S \subseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$. Supposons maintenant que $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 1]} = 0$. Alors, par hypothèse de récurrence du fait A.2 nous savons que pour tout ensemble d'arbres $S$, si $\{\!|\} \in S$, alors $[\![f_1^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}[\xi \to 1]} = 0$ si et seulement si $\{\!|\} \notin [\![f_1]\!]_{\delta[\xi \to S]}$ ; donc de l'hypothèse nous déduisons que

($\star$) si $\{\!|\} \in S$, alors $\{\!|\} \notin [\![f_1]\!]_{\delta[\xi \to S]}$.

En particulier, si $S \subseteq [\![f_1]\!]_{\delta[\xi \to S]}$, alors nécessairement $\{\!|\} \notin S$, donc $\{\!|\} \notin \bigcup\{S \subseteq \mathcal{A}_\Sigma \mid S \subseteq [\![f_1]\!]_{\delta[\xi \to S]}\}$. Ceci termine la preuve de (b). Ceci termine la preuve du fait A.2.

Revenons maintenant à la preuve du lemme. Nous devons montrer que pour tout système d'équations $\mathcal{S}$ sur $\mathit{fixpt}(\tau', \mathcal{X}_\tau)$, et pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ si et seulement si $Sol_{\mathcal{B}}(\mathcal{S}^{\mathsf{b}}, \xi) = 1$. La preuve se fait par récurrence sur la nombre de variables de $\mathcal{S}$.

Pour le cas de base, $\mathcal{S}$ est le système à une équation $\xi \stackrel{\kappa}{=} f$, et par définition $Sol_{\mathcal{T}}(\mathcal{S}, \xi) = [\![\kappa\xi.f]\!]$ et $Sol_{\mathcal{B}}(\mathcal{S}^{\mathsf{b}}, \xi) = [\![\kappa\xi.f^{\mathsf{b}}]\!]$. Nous savons par le fait A.2, que $\{\!|\} \in [\![\kappa\xi.f]\!]$ si et seulement si $[\![\kappa\xi.f^{\mathsf{b}}]\!] = 1$. Pour le cas d'induction, supposons que pour tout système à $n$ équations $\mathcal{S}$ et pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S})$, $\{\!|\}$ appartient à $Sol_{\mathcal{T}}(\mathcal{S}, \xi)$ si et seulement si $Sol_{\mathcal{B}}(\mathcal{S}^{\mathsf{b}}, \xi) = 1$. Soit maintenant $\mathcal{S}'$ le système d'équations à $n+1$ équations $\xi_0 \stackrel{\kappa_0}{=} f_0$, $\xi_1 \stackrel{\kappa_1}{=} f_1$, $\cdots$, $\xi_n \stackrel{\kappa_n}{=} f_n$ et soit $g$ est le terme $\kappa_0\xi_0.f_0$. Par définition, la solution de $\mathcal{S}'$ est le

tuple $\langle g(S_1,\ldots,S_n), S_1,\ldots,S_n\rangle$ où $\langle S_1,\ldots,S_n\rangle$ est la solution du système d'équations $\xi_1 \stackrel{\kappa_1}{=} f_1\langle x_0 \to g\rangle$, $\cdots$, $\xi_n \stackrel{\kappa_n}{=} f_n\langle x_0 \to g\rangle$. De la même façon, la solution de $\mathcal{S}'^{\mathsf{b}}$ est le tuple $\langle g^{\mathsf{b}}(h_1,\ldots,h_n), h_1,\ldots,h_n\rangle$ où $\langle h_1,\ldots,h_n\rangle$ est la solution du système d'équations $\xi_1 \stackrel{\kappa_1}{=} f_1^{\mathsf{b}}\langle x_0 \to g^{\mathsf{b}}\rangle$, $\cdots$, $\xi_n \stackrel{\kappa_n}{=} f_n^{\mathsf{b}}\langle x_0 \to g^{\mathsf{b}}\rangle$. Par hypothèse de récurrence,

($\star$) pour tout $i$ dans $1..n$, $\{\!|\}\!| \in S_i$ si et seulement si $h_i = 1$.

Donc il suffit de montrer que $\{\!|\}\!| \in Sol_{\mathcal{T}}(\xi_0,\mathcal{S})$ si et seulement si $Sol_{\mathcal{B}}(\xi_0,\mathcal{S}^{\mathsf{b}}) = 1$. Soit $\delta$ la valuation de domaine $\{\xi_1,\ldots,\xi_n\}$ qui à $\xi_i$ associe $S_i$ pour tout $i$ dans $1..n$. Alors, par définition, $Sol_{\mathcal{T}}(\xi_0,\mathcal{S}') = [\![g]\!]_\delta$. Par ($\star$), $\gamma^{\mathsf{b}}$ est la valuation qui à $\xi_i$ associe $h_i$ pour tout $i$ dans $1..n$, et donc $Sol_{\mathcal{B}}(\xi_0,\mathcal{S}') = [\![g^{\mathsf{b}}]\!]_{\gamma^{\mathsf{b}}}$. Alors la conclusion suit immédiatement du fait A.2.

## Preuve du lemme 4.31

Nous devons montrer que pour tout système d'équations normalisé $\mathcal{S}$, le système d'équations $\mathsf{CG}(\mathcal{S})$ est un système d'équations à composition gardée équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$.

Il est immédiat de la définition que $\mathsf{CG}(\mathcal{S})$ est un système d'équations à composition gardée. Nous allons montrer que pour tout $i$ dans $1..n$, le système d'équations

$$\mathcal{S}_i = \mathsf{CG}'(x_i, \mathsf{CG}'(x_{i-1},\ldots,\mathsf{CG}'(x_1,\mathcal{S})))$$

est équivalent à $\mathcal{S}$ sur l'ensemble de variables $\{\mathsf{Vars}(\mathcal{S})\}$. Comme $\mathsf{CG}'(\mathcal{S}) = \mathcal{S}_n$, il en suit immédiatement que $\mathsf{CG}'(\mathcal{S})$ est équivalent à $\mathcal{S}$ sur l'ensemble de variables $\mathsf{Vars}(\mathcal{S})$.

Il est facile de voir que pour tout $i$ dans $1..n$, les $n-i$ dernières équations de $\mathcal{S}_i$ sont le mêmes que les $n-i$ dernières équations de $\mathcal{S}$. Supposons que dans $\mathcal{S}$, pour toute variable $x_i$ dans $\mathsf{Vars}(\mathcal{S})$, l'équation pour cette variable est $x_i \stackrel{\kappa_i}{=} f_i$.

La preuve se fait par récurrence sur $i$. Pour le cas de base, $i = 1$. Si la partie droite de l'équation pour $x$ dans $\mathcal{S}$ n'est pas un terme à composition non gardée, $\mathsf{CG}'(x_1,\mathcal{S}) = \mathcal{S}$. Sinon, soit $x_1 \stackrel{\kappa}{=} y \circ z$ cette équation, où $\circ$ est l'un parmi $\mid$ et $\|$. Il y a huit cas possibles suivant $\circ$ et le fait que $\{\!|\}\!|$ appartienne ou non aux ensembles $Sol_{\mathcal{T}}(\mathcal{S},y)$ et $Sol_{\mathcal{T}}(\mathcal{S},z)$. Tous ces cas se ressemblent, nous donnons uniquement la preuve pour $\circ$ étant $\mid$ et $\{\!|\}\!| \in Sol_{\mathcal{T}}(\mathcal{S},y)$ et $\{\!|\}\!| \notin Sol_{\mathcal{T}}(\mathcal{S},z)$. Dans ce cas, $\mathsf{CG}'(x_1,\mathcal{S})$ est le système $x \stackrel{\kappa}{=} y \lceil z \vee z$, $x_2 \stackrel{\kappa_2}{=} f_2$, $\ldots$, $x_n \stackrel{\kappa_n}{=} f_n$. Il suffit de montrer que les solutions des systèmes

$$\begin{array}{lll} x_1 & \stackrel{\kappa_1}{=} & (y \mid z) \quad (x_1,\ldots,x_n) \\ x_2 & \stackrel{\kappa_2}{=} & f_2 \quad\quad (x_1,\ldots,x_n) \\ \vdots & & \vdots \\ x_n & \stackrel{\kappa_n}{=} & f_n \quad\quad (x_1,\ldots,x_n) \end{array} \tag{A.2}$$

et

$$\begin{array}{lll} x_1 & \stackrel{\kappa_1}{=} & (y \lceil z \vee z) \quad (x_1,\ldots,x_n) \\ x_2 & \stackrel{\kappa_2}{=} & f_2 \quad\quad (x_1,\ldots,x_n) \\ \vdots & & \vdots \\ x_n & \stackrel{\kappa_n}{=} & f_n \quad\quad (x_1,\ldots,x_n) \end{array} \tag{A.3}$$

coïncident sur les variables $x_1, \ldots, x_n$. Au vue de la forme des deux systèmes, il suffit de montrer que les solutions coïncident pour la variable $x_1$, ce qui est conséquence immédiate de la définition de la solution d'un système d'équations et du lemme 4.28.

Pour l'induction, $i > 1$. Soit $\mathcal{S}_{i-1}$ le système $\mathcal{S}'$, $x_i \overset{\kappa}{=} f$, $\mathcal{S}''$ et donc $\mathcal{S}_i$ est le système $\mathcal{S}'$, $\mathsf{cg}(x_i \overset{\kappa}{=} f)$, $\mathcal{S}''$. Si $f$ n'est pas un terme à composition, alors $\mathcal{S}_i = \mathcal{S}_{i-1}$. Sinon, soit $f = y \circ z$, où $\circ$ est l'un parmi $|, \|$. Il y a huit cas à considérer suivant le type de composition dans $f$ et le fait que $\{\!|\}\!|$ appartienne ou non à $Sol_{\mathcal{T}}(\mathcal{S}_{i-1}, y)$ et $Sol_{\mathcal{T}}(\mathcal{S}_{i-1}, z)$. Comme précédemment, tous les cas suivent le même raisonnement et nous considérons ici seulement le cas où $f = y \,|\, z$ et $\{\!|\}\!| \in Sol_{\mathcal{T}}(\mathcal{S}_{i-1}, y)$ et $\{\!|\}\!| \notin Sol_{\mathcal{T}}(\mathcal{S}_{i-1}, z)$. Dans ce cas, $\mathsf{cg}(x_i \overset{\kappa}{=} f)$ est le système $x \overset{\kappa}{=} y \sqcap z \vee z$. Nous allons utiliser la proposition 4.4 pour nous ramener au cas de base. Suivant cette proposition, la solution du système d'équations $\mathcal{S}_{i-1}$ s'exprime en fonction des solutions des systèmes $\mathcal{S}'$ et $x_i \overset{\kappa}{=} y \,|\, z$, $\mathcal{S}''$ et la solution de $\mathcal{S}_i$ s'exprime en fonction des solutions des systèmes $\mathcal{S}'$ et $\mathsf{cg}(x_i \overset{\kappa}{=} f)$, $\mathcal{S}''$. Pour toute variable $x$ dans $\mathsf{Vars}(\mathcal{S})$, soit $h_x(x_i, \mathsf{Vars}(\mathcal{S}''))$ la solution de $\mathcal{S}'$ (rappelons que $h_x$ est une fonction sur la séquence de variables $x_i, \mathsf{Vars}(\mathcal{S}'')$ prenant ces valeurs dans $\wp(\mathcal{A}_\Sigma)$) et soit $\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}''))$ la solution de $\mathcal{S}'$ (qui est un tuple de fonctions). Comme la solution de $\mathcal{S}'$ est commune pour les systèmes $\mathcal{S}_{i-1}$, il suffit de montrer que les solutions des systèmes

$$\begin{array}{llll} x_i & \overset{\kappa}{=} & (y \,|\, z) & (\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}'')), x_i, \mathsf{Vars}(\mathcal{S}'')), \\ x_{i+1} & \overset{\kappa_{i+1}}{=} & f_{i+1} & (\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}'')), x_i, \mathsf{Vars}(\mathcal{S}'')), \\ \vdots & & \vdots & \\ x_n & \overset{\kappa_n}{=} & f_n & (\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}'')), x_i, \mathsf{Vars}(\mathcal{S}'')) \end{array}$$

et

$$\begin{array}{llll} x_i & \overset{\kappa}{=} & (y \sqcap z \vee z) & (\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}'')), x_i, \mathsf{Vars}(\mathcal{S}'')), \\ x_{i+1} & \overset{\kappa_{i+1}}{=} & f_{i+1} & (\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}'')), x_i, \mathsf{Vars}(\mathcal{S}'')), \\ \vdots & & \vdots & \vdots \\ x_n & \overset{\kappa_n}{=} & f_n & (\mathbf{h}(x_i, \mathsf{Vars}(\mathcal{S}'')), x_i, \mathsf{Vars}(\mathcal{S}'')) \end{array}$$

coïncident sur l'ensemble des variables $\{x_i\} \cup \{\mathsf{Vars}(\mathcal{S}'')\}$. Comme pour le cas de base, c'est une conséquence immédiate de la définition de la solution d'un système d'équations.

## Preuve du lemme 4.34

Nous considérons $\mathcal{S}$ un système d'équations sur les variables $\xi_1, \ldots, \xi_n$ et la suite $(\mathcal{S}_i)_{i \in 0..n}$ de systèmes d'équations suivante : $\mathcal{S}_0 = \mathcal{S}$ et pour tout $i$ dans $1..n$,
$\mathcal{S}_i = \mathsf{transforme}(\mathsf{élimine}(\mathcal{S}_{i-1}, \xi_i), \xi_i)$. Nous devons montrer que si $\mathcal{S}$ est un système d'équations semi-simplifié, alors pour tout $i$ dans $0..n$ et pour tous $j, k$ dans $1..n$ tels que $j \leq i$, si $\mathcal{B}_{\mathcal{S}_i}(\xi_j, \xi_k)$, alors $k > j$.

Notons d'abord que, par le lemme 4.33, $\mathcal{S}_i$ est un système d'équations semi-simplifié. Donc, par définition, pour tous $j, k$ dans $1..n$, $\mathcal{B}_{\mathcal{S}_i}(\xi_j, \xi_k)$ si et seulement si l'équation pour $\xi_j$ est un terme booléen et $\xi_k$ apparaît dans ce terme. Nous montrons les deux propriétés suivantes :

HR1($i$) pour tout $j > i$, si la partie droite de l'équation pour $\xi_j$ dans $\mathcal{S}_i$ est un terme booléen, alors les variables $\xi_1, \ldots, \xi_i$ n'apparaissent dans ce terme ;

HR2($i$) pour tout $j \leq i$, si la partie droite de l'équation pour $\xi_j$ dans $\mathcal{S}_i$ est un terme booléen, alors les variables $\xi_1, \ldots, \xi_j$ n'apparaissent pas dans ce terme.

Le lemme est conséquence immédiate de HR1($i$).

La preuve de ces deux propriétés se fait par récurrence sur $i$. Pour le cas de base, $i = 0$ et HR1(0) et HR2(0) sont triviaux. Pour l'induction, soit $i > 0$ et soient les propriétés HR1($l$) et HR2($l$) vérifiées pour tout $l \leq i$. Nous allons montrer que HR1($i+1$) et HR2($i+1$) sont vérifiées. Nous notons $f_l$ la partie droite de l'équation pour la variable $\xi_l$ dans $\mathcal{S}_i$, ceci pour chaque $l$ dans $1..n$.

Pour HR1($i+1$), soit $j > i+1$ et soit le terme $f_j$ booléen. Nous devons montrer que les variables $\xi_1, \ldots, \xi_{i+1}$ n'apparaissent pas dans la partie droite de l'équation pour $\xi_j$ dans $\mathcal{S}_{i+1}$. Par définition, la partie droite de l'équation pour $\xi_j$ dans $\mathcal{S}_{i+1}$ est le terme booléen

$$f_j\langle \xi_{i+1} \rightarrow f_{i+1}\langle \xi_{i+1} \rightarrow h\rangle\rangle$$

où $h$ est l'un parmi $\bot$ et $\top$. La variable $\xi_{i+1}$ n'apparaît pas dans ce terme. Donc, il suffit de montrer que (a) les variables $\xi_1, \ldots, \xi_i$ n'apparaissent pas dans $f_j$ et (b) les variables $\xi_1, \ldots, \xi_i$ n'apparaissent pas dans $f_{i+1}(\xi_1, \ldots, \xi_i, B, \xi_{i+2}, \ldots, \xi_n)$. La propriété (a) et (b) sont vraies par HR1($i$).

Pour HR2($i+1$), soit $j \leq i+1$. Nous devons montrer que les variables $\xi_1, \ldots, \xi_j$ n'apparaissent pas dans $f_j$. Si $j \leq i$, c'est vrai par HR2($i$). Si $j = i+1$, alors par définition la partie droite de l'équation pour $\xi_j$ dans $\mathcal{S}_{i+1}$ est le terme

$$f_{i+1}(\xi_1, \ldots, \xi_i, B, \xi_{i+2}, \ldots, \xi_n)$$

Il est clair que la variable $\xi_{i+1}$ n'apparaît pas dans ce terme et, par HR1($i$), les variables $\xi_1, \ldots, \xi_i$ n'apparaissent pas dans ce terme, ce qui termine la preuve.

## Preuve du lemme 4.39

Nous devons montrer que l'application $d : \mathcal{A}_\Sigma, \mathcal{A}_\Sigma \rightarrow \mathbb{R}$ définie par

$$\begin{cases} d(S,S) = 0 \\ d(S,T) = 2^{-x} \text{ où } x = \inf\{|t| \mid t \in (S \cup T) \setminus (S \cap T)\} \text{ pour } S \neq T \end{cases}$$

est une distance pour les ensembles d'arbres.

Rappelons que $d$ est une distance sur $\wp(\mathcal{A}_\Sigma)$ si c'est une application de $\wp(\mathcal{A}_\Sigma) \times \wp(\mathcal{A}_\Sigma)$ dans $\mathbb{R}$ qui satisfait les conditions suivantes :

- $d$ est positive,
- $d$ est symétrique,
- pour tous $S, T \subseteq \mathcal{A}_\Sigma$, $d(S,T) = 0$ si et seulement si $S = T$ et
- $d$ satisfait l'inégalité triangulaire : $\forall S, T, U \subseteq \mathcal{A}_\Sigma$, $d(S,T) \leq d(S,U) + d(T,U)$.

Les trois premières conditions sont conséquence immédiate de la définition de $d$. Montrons que $d$ satisfait l'inégalité triangulaire. Soient $u, s, t$ des arbres qui servent au calcul des quantités $d(S,T)$, $d(U,T)$ et $d(U,S)$ respectivement, càd $u \in (S \cup T) \setminus (S \cap T)$ tel que $d(S,T) = 2^{-|u|}$, et de même pour les arbres $s$ et $t$ respectivement aux ensembles $U, T$ et $U, S$. La preuve se fait en envisageant tous les cas d'appartenance de chacun des arbres $u, s, t$ à chacun des ensembles $U, S, T$. Tout d'abord, pour l'arbre $u$ et les ensembles $S, T$ nous avons deux possibilités : ou bien

($\star$) $u \in S, u \notin T$, ou bien

($\star\star$) $u \in T, u \notin S$.

Supposons que c'est ($\star$) qui est vérifié. Maintenant, si l'arbre $u$ appartient à $U$, alors nous déduisons que $u \notin T, u \in U$, et comme nous savons que $d(U,T) = 2^{-|s|}$ et par définition de la distance $d$, nous déduisons que $|u| \geq |s|$. Si au contraire $u$ n'appartient pas à $U$, nous pourrons déduire que $|u| \geq |t|$. D'autre part, si c'est ($\star\star$) qui est vérifié, et si $u \in U$, nous déduisons que $|u| \geq |t|$, et si $u \notin U$, nous déduisons que $|u| \geq |s|$. De façon symétrique, pour l'arbre $s$ (respectivement l'arbre $t$), et en fonction des différentes possibilités d'appartenance de $s$ aux ensembles $U, S, T$ et de $t$ aux ensembles $U, S, T$, nous pouvons déduire indépendamment ou bien que $|s| \geq |t|$ ou bien que $|s| \geq |u|$ (respectivement ou bien que $|t| \geq |s|$ ou bien que $|t| \geq |u|$). En fonction de la combinaison de ces différentes égalités, nous avons donc huit possibilités différentes :

(i) $|u| \geq |s|, |s| \geq |u|, |t| \geq |s|$, et donc $|t| \geq |s| = |u|$ ;

(ii) $|u| \geq |s|, |s| \geq |u|, |t| \geq |u|$, et donc $|t| \geq |s| = |u|$ ;

(iii) $|u| \geq |s|, |s| \geq |t|, |t| \geq |s|$, et donc $|u| \geq |s| = |t|$ ;

(iv) $|u| \geq |s|, |s| \geq |t|, |t| \geq |u|$, et donc $|u| = |s| = |t|$ ;

(v) $|u| \geq |t|, |s| \geq |u|, |t| \geq |s|$, et donc $|u| = |s| = |t|$ ;

(vi) $|u| \geq |t|, |s| \geq |u|, |t| \geq |u|$, et donc $|s| \geq |t| = |u|$ ;

(vii) $|u| \geq |t|, |s| \geq |t|, |t| \geq |s|$, et donc $|u| \geq |s| = |t|$ ;

(viii) $|u| \geq |t|, |s| \geq |t|, |t| \geq |u|$, et donc $|s| \geq |t| = |u|$.

Finalement nous distinguons deux cas de figure : le premier est pour $|u| = |s|$ ou $|u| = |t|$, il couvre tous les cas ci-dessus sauf (iii) et (vii). Il est évident que dans ce cas $2^{-|u|} \leq 2^{-|s|} + 2^{-|t|}$. Le deuxième correspond au cas (iii) et (vii) et nous avons $|u| \geq |s| = |t|$, ce qui implique $2^{-|u|} \leq 2^{-|s|}$ et donc $2^{-|u|} \leq 2^{-|s|} + 2^{-|t|}$.

## Preuve du lemme 4.41

Nous devons montrer que si $\mathcal{S}$ le système d'équations normalisé

$$x_1 \overset{\kappa_1}{=} f_1, \ldots, x_n \overset{\kappa_n}{=} f_n,$$

alors pour tout $i$ dans $1..n$ et pour tous ensembles d'arbres $S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$, la fonction $g(x_i) = f_i^{\mathcal{T}}(S_1, \ldots, S_{i-1}, x_i, S_{i+1}, \ldots, S_n)$ est strictement contractante.

Soit $i \in 1..n$, les ensembles d'arbres $S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$ et soit $g$ la fonction $x_i \mapsto f_i^{\mathcal{T}}(S_1, \ldots, S_{i-1}, x_i, S_{i+1}, \ldots, S_n)$. Nous allons montrer que $g$ est contractante de facteur de contraction $2^{-1}$. Considérons les différents cas pour la forme de $f_i$. Tout d'abord, remarquons que si la variable $x_i$ n'apparaît pas dans $f_i$, alors $g$ est une fonction constante et pour tous ensembles $S, T$, $g(S) = g(T)$, donc $d(g(S), g(T)) = 0 \leq 2^{-1} d(S,T)$. Maintenant, si $x_i$ apparaît dans $f_i$, nous savons que $f_i$ ne peut pas être un terme booléen puisque $\mathcal{S}$ est un système normalisé et donc sans cycles de dépendance booléenne. Il nous reste alors trois cas à considérer :

(i) $f_i = \alpha[x_i]$, (ii) $f_i = x_i \lceil x_j$ et (iii) $f_i = x_i \rfloor\lceil x_j$ (les cas $f_i = x_j \lceil x_i$ et $f_i = x_j \rfloor\lceil x_i$ sont symétriques à (ii) et (iii) respectivement). Dans la suite de la preuve nous dirons que *l'arbre $t$ fait la distance entre les ensembles $S$ et $T$* si $t \in (S \cup T) \setminus (S \cap T)$ et $d(S,T) = 2^{-|t|}$.

Pour (i), $f_i = \alpha[x_i]$, pour un certain $\alpha \subseteq \Sigma$ non vide. Soient $S, T \in \wp(\mathcal{A}_\Sigma)$. Alors $d(g(S), g(T)) = d(\alpha[S], \alpha[T])$. Par définition de $\alpha[S]$ et $\alpha[T]$, il est facile de voir que l'ensemble $(\alpha[S] \cup \alpha[T]) \setminus (\alpha[S] \cap \alpha[T])$ est égal à l'ensemble $\alpha[(S \cup T) \setminus (S \cap T)]$. Soit $t$ un arbre qui fait la distance entre $S$ et $T$. Alors pour tout $a \in \alpha$, l'arbre $\{\!|a[t]|\!\}$ est dans $(\alpha[S] \cup \alpha[T]) \setminus (\alpha[S] \cap \alpha[T])$. De plus, il est facile de voir qu'il n'y a pas d'arbre de taille inférieure strictement à $|\{\!|a[t]|\!\}|$ appartenant à $(\alpha[S] \cup \alpha[T]) \setminus (\alpha[S] \cap \alpha[T])$, donc $\{\!|a[t]|\!\}$ fait la distance entre $\alpha[S]$ et $\alpha[T]$. Et comme $|\{\!|a[t]|\!\}| = 1 + |t|$, nous en déduisons que $d(g(S), g(T)) = 2^{-(|t|+1)} \leq 2^{-1} d(S, T)$.

Pour (ii), $g(x_i) = x_i \int^{\mathcal{T}} S_j$. Soit $S, T \in \wp(\mathcal{A}_\Sigma)$. Alors $d(g(S), g(T)) = d(S \int^{\mathcal{T}} S_j, T \int^{\mathcal{T}} S_j)$. Soit $t$ un arbre qui fait la distance entre $S \int^{\mathcal{T}} S_j$ et $T \int^{\mathcal{T}} S_j$, et supposons que $t \in S \int^{\mathcal{T}} S_j$. Soient les arbres $s, r$ tels que $t = s \uplus r$ et $s \in S \setminus \{\{\!|\}\!\}\}$ et $r \in S_j \setminus \{\{\!|\}\!\}\}$. Alors l'arbre $s$ n'appartient pas à $T$ ; en effet, dans le cas contrarie, $s \uplus r = t$ appartiendrait à $T \int^{\mathcal{T}} S_j$, ce qui contredirait l'hypothèse que $t$ fait la distance entre $S \int^{\mathcal{T}} S_j$ et $T \int^{\mathcal{T}} S_j$. Il en suit que $|s| \geq |h|$, où $h$ est un arbre qui fait la distance entre $S$ et $T$. Donc nous avons l'inégalité $d(S \int^{\mathcal{T}} S_j, T \int^{\mathcal{T}} S_j) = 2^{-|s \uplus r|} = 2^{-|s|} 2^{-|r|} \leq 2^{-|h|} 2^{-|r|} = 2^{-|r|} d(S, T)$. Il suffit alors de remarquer que $2^{-|r|} \leq 2^{-1}$ puisque, par hypothèse, $r \neq \{\!|\}\!\}$ et donc $|r| \geq 1$. Le cas où $t \in T \int^{\mathcal{T}} S_j$ est symétrique.

Pour (iii), $g(x_i) = x_i \rceil\!\int^{\mathcal{T}} S_j$. Soient $S, T \in \wp(\mathcal{A}_\Sigma)$. Alors $d(g(S), g(T)) = d(S \rceil\!\int^{\mathcal{T}} S_j, T \rceil\!\int^{\mathcal{T}} S_j)$. Remarquons maintenant que pour tous ensembles d'arbres $S', T'$ on a $d(\mathcal{A}_\Sigma \setminus S', \mathcal{A}_\Sigma \setminus T') = d(S', T')$. Pour faciliter la lecture, notons par $\overline{R}$ l'ensemble $\mathcal{A}_\Sigma \setminus R$ (ceci pour tout ensemble d'arbres $R$). De $S \rceil\!\int^{\mathcal{T}} T = \overline{\overline{S} \int^{\mathcal{T}} \overline{T}}$, nous déduisons que $d(S \rceil\!\int^{\mathcal{T}} S_j, T \rceil\!\int^{\mathcal{T}} S_j) = d(\overline{S} \int^{\mathcal{T}} \overline{S_j}, \overline{T} \int^{\mathcal{T}} \overline{S_j})$. Par ce qui a été prouvé dans (ii) et en utilisant le fait que l'ensemble $S_j$ est arbitraire, nous déduisons que $d(\overline{S} \int^{\mathcal{T}} \overline{S_j}, \overline{T} \int^{\mathcal{T}} \overline{S_j}) \leq 2^{-1} d(\overline{S}, \overline{T})$. Par la remarque précédente, la partie droite de cette inégalité est $d(S, T)$.

## A.3 Preuves omises dans le chapitre 7

Soit $\phi$ une formule close de la logique $\mathrm{LS}_{|\exists}$. Rappelons que $\mathsf{Eq}(\phi)$ est le système d'équations sur $fonct(\tau, \mathcal{X}_\mathrm{r})$ tel que sa solution pour la dernière variable est l'ensemble d'arbres $[\![\phi]\!]$ (voir la section 4.2.4). Nous noterons $\mathcal{S}_\phi$ le système d'équations simplifié construit à partir de $\mathsf{Eq}(\phi)$ comme il a été décrit dans la section 4.3.1 et $\mathcal{S}_\phi^{\mathrm{num}}$ le système numérique correspondant, obtenu comme il a été décrit dans la section 6.2.2. Rappelons finalement que $\mathsf{pos}(\phi)$ est la formule positive construite à partir de la formule $\phi$ comme il a été décrit dans la section 4.2.4.

Nous commençons par élargir la notion de dépendance entre variables aux systèmes d'équations $\mathsf{Eq}(\phi)$ et $\mathcal{S}_\phi$.

**Définition A.1 (Chemin de dépendance et dépendance entre variables)** Un chemin est dit *chemin de dépendance* s'il contient uniquement les opérateurs $\vee, \wedge, |, \| , \xi$. Si $c$ est un chemin de dépendance dans la formule $\phi$, la formule $\phi(c)$ est dite *en position de dépendance dans* $\phi$.

Soit $\mathcal{S}$ l'un des systèmes d'équations $\mathsf{Eq}(\phi)$, $\mathcal{S}_\phi$ ou $\mathcal{S}_\phi^{\mathrm{num}}$. Nous définissons la relation binaire $\longrightarrow_\mathcal{S}$ entre variables du système $\mathcal{S}$ : pour $\xi, \xi'$ appartenant à $\mathsf{Vars}(\mathcal{S})$ et $f$ étant la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$, nous disons que $\xi$ *dépend de* $\xi'$ *dans* $\mathcal{S}$, noté $\xi \longrightarrow_\mathcal{S} \xi'$, si $\xi'$ est en position de dépendance dans $f$. Nous écrirons $\xi \xrightarrow{c}_\mathcal{S} \xi'$ pour préciser le chemin de dépendance $c$ tel que $f(c) = \xi'$.

La clôture transitive et réflexive de $\longrightarrow_\mathcal{S}$ est notée $\longrightarrow_\mathcal{S}^*$.

Remarquons que pour $\mathcal{S}$ étant le système d'équations numérique $\mathcal{S}_\phi^{\mathrm{num}}$, cette définition coïncide avec la définition de la relation de dépendance entre variables telle qu'elle a été définie dans la section 7.2.2.

L'exemple suivant illustre la dépendance entre variables.

**Exemple A.1** Soit $\xi$ une variable dans le système d'équations $\mathcal{S}$ et soit

$$\phi = (\alpha[\mathbf{0}] \vee (\beta[\xi_2] \,|\, \xi_1)) \wedge (\beta[\xi_1] \,\|\, \xi_3)$$

la partie droite de l'équation pour $\xi$ dans $\mathcal{S}$. Pour la variable $\xi_1$, $\xi_1 = \phi(c)$ si $c$ est l'un des chemins $\wedge.\vee.|.\xi_1$ ou $\wedge.\|.\beta.\xi_1$. Le premier de ces chemins est un chemin de dépendance, donc $\xi \longrightarrow_\mathcal{S} \xi_1$. Pour la variable $\xi_2$, l'unique occurrence de cette variable dans $\phi$ est atteinte via le chemin $\wedge.\vee.\beta.\xi_2$ et donc il n'est pas le cas que $\xi \longrightarrow_\mathcal{S} \xi_2$. Pour la variable $\xi_3$, nous avons $\xi_3 = \phi(\wedge.\|.\xi_3)$ et donc $\xi \longrightarrow_\mathcal{S} \xi_3$.

Remarquons que, d'après la définition du système numérique $\mathcal{S}_\phi^{\mathrm{num}}$, pour toutes variables $\xi, \xi'$ on a $\xi \longrightarrow_{\mathcal{S}_\phi} \xi'$ si et seulement si $\xi \longrightarrow_{\mathcal{S}_\phi^{\mathrm{num}}} \xi'$. Donc nous utiliserons indépendamment $\longrightarrow_{\mathcal{S}_\phi}$ et $\longrightarrow_{\mathcal{S}_\phi^{\mathrm{num}}}$.

Dans la suite, nous allons mettre en évidence les liens existant entre $\mathsf{pos}(\phi)$, $\mathsf{Eq}(\phi)$, $\mathcal{S}_\phi$ et les dépendances entre variables induites par ces deux systèmes.

**Lemme A.2** *Soit une formule* $\phi$.

*(i) Pour toute variable* $\xi$ *dans* $\mathsf{Vars}(\mathcal{S}_\phi)$, *si la classe d'équivalence* $(\xi)_{\mathcal{S}_\phi}$ *existe, alors elle contient au moins une variable de* $\mathsf{Vars}(\mathsf{Eq}(\phi))$ ;

*(ii) Si $\xi, \xi'$ sont des variables dans* $\mathsf{Vars}(\mathsf{Eq}(\phi))$ *et* $x_1, \ldots, x_n$ *sont des variables dans* $\mathsf{Vars}(\mathcal{S}_\phi)$ *mais pas dans* $\mathsf{Vars}(\mathsf{Eq}(\phi))$ *et si*

$$\xi \xrightarrow{c_0}_{\mathcal{S}_\phi} x_1 \xrightarrow{c_1}_{\mathcal{S}_\phi} \cdots \xrightarrow{c_{n-1}}_{\mathcal{S}_\phi} x_n \xrightarrow{c_n}_{\mathcal{S}_\phi} \xi'$$

*alors $\xi \xrightarrow{c}_{\mathsf{Eq}(\phi)} \xi'$ où $c$ est le chemin obtenu par la concaténation des chemins $c_0, \ldots, c_n$. De plus, si l'opérateur de point fixe associé à $\xi$ dans* $\mathsf{Eq}(\phi)$ *est $\kappa$, alors c'est aussi l'opérateur de point fixe associé à chacune des variables $x_1, \ldots, x_n$ dans $\mathcal{S}_\phi$.*

*(iii) Soient $\xi', \xi''$ deux variables dans* $\mathsf{Vars}(\mathsf{Eq}(\phi))$ *et soient $\phi', \phi''$ les sous formules de* $\mathsf{pos}(\phi)$ *telles que $\kappa'\xi'.\phi'$ et $\kappa''\xi''.\phi''$ sont les sous formules à récursion correspondant aux variables $\xi'$ et $\xi''$. Alors $\xi \xrightarrow{c}_{\mathsf{Eq}(\phi)} \xi'$ si et seulement si l'une de ces deux conditions est vérifiée : ou bien $\phi'(c) = \kappa''\xi''.\phi''$, ou bien $\phi'(c) = \xi''$. Dans ce deuxième cas, $\kappa'\xi'.\phi'$ est sous formule de $\kappa''\xi''.\phi''$.*

*Preuve* Suit facilement des définitions. Le premier point est conséquence de la méthode de construction du système $\mathcal{S}_\phi$. En effet, on peut voir que toute variable ajoutée à un système lors de l'étape de simplification ne peut dépendre que de variables déjà présentes dans le système. Donc, il est impossible de faire un cycle en n'utilisant que des variables ajoutée pour la simplification. Pour le second point, la preuve est une récurrence facile sur la structure de la formule qui est la partie droite pour la variable $\xi$ dans $\mathsf{Eq}(\phi)$. Le troisième point est conséquence facile de la définition du système d'équations $\mathsf{Eq}(\phi)$ et de la relation de dépendance entre variables.

□

Soit $(\xi)_{\mathcal{S}_\phi}$ une classe d'équivalence dans le système $\mathcal{S}_\phi$. D'après le lemme précédent, tout cycle pour la relation $\longrightarrow_{\mathcal{S}_\phi}$ d'éléments de $(\xi)_{\mathcal{S}_\phi}$ est de la forme

$$\xi_1 \longrightarrow_{\mathcal{S}_\phi} x_1^1 \longrightarrow_{\mathcal{S}_\phi} \cdots \longrightarrow_{\mathcal{S}_\phi} x_{n_1}^1 \longrightarrow_{\mathcal{S}_\phi} \xi_2 \longrightarrow_{\mathcal{S}_\phi} \cdots$$
$$\cdots \longrightarrow_{\mathcal{S}_\phi} \xi_k \longrightarrow_{\mathcal{S}_\phi} x_1^k \longrightarrow_{\mathcal{S}_\phi} \cdots \longrightarrow_{\mathcal{S}_\phi} x_{n_k}^k \longrightarrow_{\mathcal{S}_\phi} \xi_1$$

où les $\xi_i$ sont des variables de $\mathsf{Vars}(\mathsf{Eq}(\phi))$ et pour chaque $i$ dans $1..k$, les $x_j^i$ ne sont pas des variables de $\mathsf{Vars}(\mathcal{S}_\phi, \xi_i)$. Par conséquent, de tout cycle pour la relation $\longrightarrow_{\mathcal{S}_\phi}$, on peut extraire un cycle pour la relation $\longrightarrow_{\mathsf{Eq}(\phi)}$. Notamment, pour le cycle ci-dessus, le cycle pour $\longrightarrow_{\mathsf{Eq}(\phi)}$ correspondant est

$$\xi_1 \longrightarrow_{\mathsf{Eq}(\phi)} \xi_2 \longrightarrow_{\mathsf{Eq}(\phi)} \cdots \longrightarrow_{\mathsf{Eq}(\phi)} \xi_k \longrightarrow_{\mathsf{Eq}(\phi)} \xi_1.$$

Soit $\ll_{\mathsf{pos}(\phi)}$ la relation d'ordre partiel entre les variables de la formule $\mathsf{pos}(\phi)$ induite par la relation de sous formule. Pour deux variables $\xi, \xi'$, on a $\xi' \ll_{\mathsf{pos}(\phi)} \xi$ si $\kappa'\xi'.\phi'$ est sous formule de $\phi$ et le chemin dans $\phi$ qui mène à $\kappa'\xi'.\phi'$ ne contient pas d'opérateurs de point fixe ; $\phi$ et $\phi$ sont les formules telles que $\kappa\xi.\phi$ et $\kappa'\xi'.\phi'$ sont des sous formules de $\mathsf{pos}(\phi)$. On note $\ll^*_{\mathsf{pos}(\phi)}$ la clôture transitive réflexive de $\ll_{\mathsf{pos}(\phi)}$ et $\ll^+_{\mathsf{pos}(\phi)}$ sa clôture transitive. Notons que si $\xi' \ll^*_{\mathsf{pos}(\phi)} \xi$, alors (avec les notations ci-dessus) $\kappa'\xi'.\phi'$ est sous formule de $\kappa\xi.\phi$ et est atteinte via un chemin qui peut contenir des opérateurs de point fixe. Avec ces notations, le troisième point du lemme A.2 implique que si $\xi' \longrightarrow_{\mathsf{Eq}(\mathcal{S})} \xi$, alors ou bien $\xi' \ll_{\mathsf{pos}(\phi)} \xi$, ou bien $\xi \ll^*_{\mathsf{pos}(\phi)} \xi'$. L'exemple A.2 ci-dessus illustre la relation $\ll_{\mathsf{pos}(\phi)}$.

**Exemple A.2** Soit $\mathsf{pos}(\phi)$ la formule

$$\mu\xi_1.a[\nu\xi_2.b[\mathbf{0}] \,||\, (\mu\xi_3.a[\mathbf{0}] \,|\, \xi_3 \vee \mathbf{0})] \vee \mathbf{0}$$

Alors $\xi_2 \ll_{\mathsf{pos}(\phi)} \xi_1, \xi_3 \ll_{\mathsf{pos}(\phi)} \xi_2$ et donc $\xi_3 \ll^*_{\mathsf{pos}(\phi)} \xi_1$ mais il n'est pas le cas que $\xi_3 \ll_{\mathsf{pos}(\phi)} \xi_1$.

Le lemme suivant met en évidence un lien très important entre les dépendances de variables dans $\mathsf{Eq}(\phi)$ et la structure de la formule $\mathsf{pos}(\phi)$.

**Lemme A.3** *Soit $\phi$ une formule et soit $\xi_0 \longrightarrow_{\mathsf{Eq}(\phi)} \cdots \longrightarrow_{\mathsf{Eq}(\phi)} \xi_{k-1} \longrightarrow_{\mathsf{Eq}(\phi)} \xi_0$ un cycle de dépendance entre variables différentes, càd $i \neq j$ implique $\xi_i \neq \xi_j$. Alors les variables $\xi_0, \ldots, \xi_{k-1}$ sont toutes deux à deux comparables par $\ll^*_{\mathsf{pos}(\phi)}$.*

*Preuve* Dans cette preuve, toute opération arithmétique entre les indices des variables est implicitement effectuée modulo $k$. Pour raccourcir l'écriture, nous utilisons $\ll$ à la place de $\ll_{\mathsf{pos}(\phi)}$.

Notons d'abord que par la remarque précédant le lemme,

($\star$) toutes deux variables consécutives dans le cycle sont comparables,

càd $\xi_{i+1} \ll \xi_i$ ou bien $\xi_i \ll^+ \xi_{i+1}$ (on a bien $\xi_i \ll^+ \xi_{i+1}$ et non $\xi_i \ll^* \xi_{i+1}$ puisque par hypothèse $i \neq i+1$). Supposons maintenant que $i$ et $j$ sont tels que $i < j$ et toutes les variables dans la portion du cycle $\xi_i \longrightarrow_{\mathsf{Eq}(\phi)} \xi_{i+1} \longrightarrow_{\mathsf{Eq}(\phi)} \cdots \longrightarrow_{\mathsf{Eq}(\phi)} \xi_{j-1} \longrightarrow_{\mathsf{Eq}(\phi)} \xi_j$ sont deux à deux comparables. (Notons que la supposition $i < j$ est sans perte de généralité : elle peut toujours être satisfaite par un renommage des variables.) Supposons également que $j+1 \neq i$, puisque le contraire est équivalent à ce que nous voulons prouver. Nous allons montrer que la variable $\xi_{j+1}$ est nécessairement comparable avec chacune des variables $\xi_1, \ldots, \xi_j$.

Nous donnons une preuve visuelle du lemme en construisant la partie de l'arbre de la relation $\ll$ contenant les variables $\xi_0, \ldots, \xi_{k-1}$. Cette construction est basée sur les informations fournies par la dépendance entre les variables et ($\star$). Un chemin non vide dans l'arbre sera représenté par un trait simple et une arête de l'arbre par un trait double (voir la figure 1). Voici les règles de la construction : nous commençons par la variable $\xi_i$ et nous ajoutons une à une les variables dans l'ordre de leur dépendance : càd $\xi_{i+1}, \xi_{i+2}, \ldots$ jusque $\xi_{i-1}$. Si $\xi_l$ est déjà placée et nous voulons ajouter $\xi_{l+1}$, d'après ($\star$) nous avons deux possibilités (voir la figure 1) :

(i) $\xi_{l+1} \ll \xi_l$, alors on ajoute $\xi_{l+1}$ en tant que nouveau fils de $\xi_l$ et on le relie à $\xi_l$ par un lien double,

(ii) $\xi_l \ll^+ \xi_{l+1}$, alors on ajoute $\xi_{l+1}$ en tant que ancêtre de $\xi_l$.

Pour un ensemble de variables $X$ inclus dans $\mathit{varrec}(\mathsf{pos}(\phi))$, dire que les variables de $X$ sont deux à deux comparables est équivalent à dire que les variables de $X$ sont toutes alignées sur une même branche de l'arbre de la relation $\ll$. Notons également que la partie de l'arbre de $\ll$ pour les variables $\xi_0, \ldots, \xi_{k-1}$ est connectée, puisque par ($\star$) chacune d'entre elles est reliée au moins à ces deux voisines dans le cycle.

Maintenant, par hypothèse, les variables $\xi_i, \ldots, \xi_j$ sont alignées dans l'arbre de la relation $\ll$. Supposons que la partie de l'arbre de $\ll$ pour ces variables est déjà construite. Montrons d'abord que

($\star\star$) si $\xi_j \ll^+ \xi_i$, alors $\xi_j \ll \xi_{j-1} \ll \cdots \ll \xi_{i+1} \ll \xi_i$.

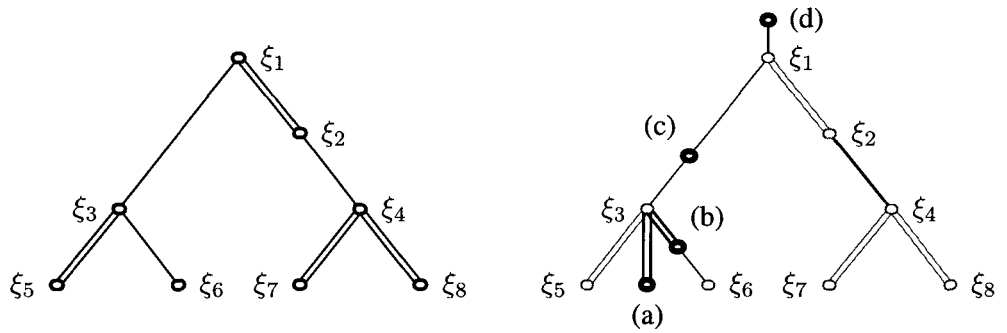



FIG. 1 – A gauche, exemple d'arbre de la relation $\ll$ pour les variables $\xi_1, \ldots, \xi_8$. Les liens doubles indiquent des arêtes de cet arbre, les liens simples indiquent des chemins de longueur un ou supérieure. A droite, les différentes possibilités d'ajouter un nouveau nœud $\xi$, sachant que $\xi_3 \longrightarrow_{\mathsf{Eq}(\phi)} \xi$. Les différents emplacements possible sont dessinés en gras, ainsi que les changements dans les liens entre les nœud. Si $\xi \ll \xi_3$, on a deux possibilités : (a) ajouter $\xi$ disjoint de tout chemin existant dans l'arbre et (b) ajouter $\xi$ sur un chemin déjà existant en coupant une ligne simple. Si $\xi_3 \ll^+ \xi$, les deux possibilités sont : (c) ajouter $\xi$ sur le chemin entre $\xi_3$ et la racine et (d) ajouter $\xi$ comme nouvelle racine.

La preuve se fait par récurrence sur la différence entre $j$ et $i$. Si $j = i + 1$, c'est vrai par $(\star)$. Supposons que $j = i + n$ pour $n > 1$ et $\xi_{j-1} \ll \cdots \ll \xi_i$. Alors, par $(\star)$ nous avons deux possibilités :

- $\xi_j \ll \xi_{j-1}$, dans quel cas on obtient ce qu'on voulait démontrer ;
- $\xi_{j-1} \ll^+ \xi_j$, càd $\xi_j$ est ancêtre de $\xi_{j-1}$. Or d'après l'hypothèse de récurrence, en remontant la branche de l'arbre de $\ll$ partant de $\xi_{j-1}$ vers la racine, nous traversons uniquement des liens doubles et les nœuds $\xi_{j-2}, \ldots, \xi_i$. Comme (par hypothèse) $\xi_j$ est différent de chacun des ces nœuds, la seule option est de placer $\xi_j$ en tant que ancêtre de $\xi_i$, ce qui contredit l'hypothèse que $\xi_j \ll^+ \xi_i$.

Nous allons montrer qu'on ne peut ajouter $\xi_{j+1}$ à la partie déjà construite que aligné avec les autres variables. Notons d'abord que si $\xi_{j+1}$ est ajouté comme ancêtre de $\xi_j$, c'est vrai. Il reste alors à considérer le cas où $\xi_{j+1}$ est ajouté en tant que fils de $\xi_j$ et relié par un lien double. Nous distinguons deux possibilités : $\xi_j \ll^+ \xi_i$ ou $\xi_i \ll^+ \xi_j$. Si $\xi_j \ll^+ \xi_i$, d'après $(\star\star)$ nous savons que $\xi_j$ est la plus petite parmi les variables $\xi_i, \ldots, \xi_j$. Donc en ajoutant $\xi_{j+1}$ en tant que fils de $\xi_j$ nous prolongeons la branche de l'arbre et les variables restent alignées. Si $\xi_i \ll^+ \xi_j$, la situation est représentée sur la figure 2. Il y a deux façons possibles d'ajouter $\xi_{j+1}$ : sur la même branche que les variables déjà présentes, dans quel cas les variables sont alignées, ou en tant que fils de $\xi_j$ non aligné avec les variables déjà présentes. Considérons cette deuxième possibilités, et montrons qu'elle contredit l'hypothèse que les variables $\xi_0, \ldots, \xi_k$ sont différentes deux à deux. En effet, nous devons pouvoir ajouter les variables $\xi_{j+1}, \xi_{j+3}, \ldots$ jusque $\xi_{i-1}$ de telle sorte que $\xi_{i-1}$ et $\xi_i$ soient directement reliées (par un lien simple ou double). L'ajout des variables se fait en utilisant une des règles (i) ou (ii). Il n'est pas difficile de se rendre compte que pour un certain $l$ parmi $j + 2, \ldots, i - 1$, la variable $\xi_l$ doit être placée en tant que ancêtre de $\xi_{l-1}$. Dans ce cas $\xi_l$ est également ancêtre de $\xi_j$, et donc de $\xi_i$. Maintenant pour atteindre $\xi_i$ à partir de $\xi_l$, nous sommes contraints à utiliser la règle (ii) pour « descendre » vers $\xi_i$, et donc nous devons traverser le nœud $\xi_j$. Ceci implique que l'une des variables $\xi_{l+2}, \ldots, \xi_{i-1}$ soit égale à $\xi_j$, ce qui contredit l'hypothèse que toutes les variables sont différentes.

□

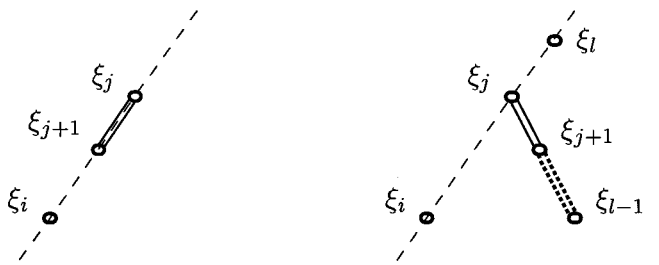



FIG. 2 – Possibilités pour placer le nœud $\xi_{j+1}$ en tant que descendant de $\xi_j$. La ligne en pointillé passant par les nœuds $\xi_i$ et $\xi_j$ représente la branche de l'arbre de $\ll$ sur laquelle ces deux nœuds se trouvent. La première possibilité (à gauche) consiste à placer $\xi_{j+1}$ sur la même branche que les variables $\xi_i$ et $\xi_j$. La deuxième possibilité place $\xi_{j+1}$ sur un branche différente. $\xi_l$ est le premier nœud dans la suite $\xi_{j+2}, \ldots, \xi_{i-1}$ qui est placé en tant qu'ancêtre de $\xi_j$. On voit sur la figure qu'on ne peut pas atteindre $\xi_i$ à partir de $\xi_l$, puisque la seule façon de descendre est en utilisant un double lien (càd règle (ii)) et on ne peut pas traverser le nœud $\xi_j$ avec ce type de lien.

Comme conséquence facile des lemmes A.2 et A.3, nous avons

**Lemme A.4** *Soit* $\xi_0 \xrightarrow{c_0}_{\mathsf{Eq}(\phi)} \cdots \xrightarrow{c_{k-1}}_{\mathsf{Eq}(\phi)} \xi_{k-2} \xrightarrow{c_{k-1}}_{\mathsf{Eq}(\phi)} \xi_0$ *un cycle de dépendance entre variables différentes, càd* $i \neq j$ *implique* $\xi_i \neq \xi_j$, *et supposons que* $\xi_0$ *est la plus grande de ses variables pour la relation* $\ll^{+}_{\mathsf{pos}(\phi)}$. *Pour tout* $i$ *dans* $1..k-1$, *soient* $\kappa_i$ *et* $\phi_i$ *tels que* $\kappa_i \xi_i . \phi_i$ *est sous formule de* $\mathsf{pos}(\phi)$. *Soit* $c$ *le chemin*

$$c = \kappa_0 \xi_0 . c_0 . \kappa_1 \xi_1 . c_1 . \kappa_2 \xi_2 . \ldots . c_{k-2} . \kappa_{k-1} \xi_{k-1} . c_{k-1} .$$

*Alors* $\kappa_0 \xi_0(c) = \xi_0$.

Nous pouvons maintenant donner les preuves des lemmes 7.9 et 7.10.

## Preuve du lemme 7.9

Nous devons montrer que si $\phi$ est une formule du fragment LSrd, alors aucune des variables ne définit de classe d'équivalence. Par définition, ceci équivaut à montrer que la relation $\longrightarrow_{\mathcal{S}^{\mathrm{num}}_{\phi}}$ est sans cycle. Par le lemme A.2, il suffit de montrer que la relation $\longrightarrow_{\mathsf{Eq}(\phi)}$ est sans cycle. En effet, supposons le contraire, et d'après le lemme A.3 soit $\xi$ la variable la plus grande dans ce cycle pour la relation d'ordre $\ll^{*}_{\mathsf{pos}(\phi)}$, et soient $\kappa$ et $\phi'$ tels que $\kappa\xi.\phi'$ est sous formule de $\mathsf{pos}(\phi)$. Soit $c$ le chemin tel que défini dans le lemme A.4. Par définition c'est un chemin de dépendance, et donc il y a une occurrence de la variable $\xi$ qui apparaît dans une position de dépendance dans $\phi'$. D'après le lemme 7.4, c'est impossible dans le fragment LSrd.

## Preuve du lemme 7.10

Nous devons montrer que si $\phi$ est une formule LSrpd, alors pour toute variable $\xi$ dans $\mathsf{Vars}(\mathcal{S}^{\mathrm{num}}_{\phi})$, le système d'équations $\mathcal{S}'$ extrait de $\mathcal{S}^{\mathrm{num}}_{\phi}$ induit par $(\!(\xi)\!)_{\mathcal{S}^{\mathrm{num}}_{\phi}}$ satisfait une de ces deux conditions :

(i) toutes les équations dans $\mathcal{S}'$ sont des équations à plus petit point fixe et toute partie droite d'équation dans $\mathcal{S}'$ utilise uniquement les opérateurs $\vee$, $|$ et les opérateurs constants $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ et des constantes $C_{\{1_q\}}$ pour un certain $q$ dans $Q_\phi$.

(ii) toutes les équations dans $\mathcal{S}'$ sont des équations à plus grand point fixe et toute partie droite d'équation dans $\mathcal{S}'$ utilise uniquement les opérateurs $\wedge$, $\|$ et les opérateurs constants $\mathbf{0}, \overline{\mathbf{0}}, \top, \bot$ et des constantes $C_{\{1_q\}}$ pour un certain $q$ dans $Q_\phi$.

Il est facile de voir par construction que pour tout constante $C_D$ apparaissant en partie droite d'équation dans $\mathcal{S}'$, l'ensemble $D$ est de la forme $\{1_q\}$, pour $q$ dans $Q_\phi$.

Dans la suite, les opérateurs $\vee, |$ sont dits positifs, et les opérateurs $\wedge, \|$ sont dits négatifs. Nous devons montrer que les équations de $\mathcal{S}'$ sont ou bien toutes à plus petit point fixe et utilisant uniquement des opérateurs positifs (en plus des constantes), ou bien toutes à plus grand point fixe et utilisant uniquement des opérateurs négatifs (en plus des constantes). Rappelons que les variables de $\mathsf{Vars}(\mathcal{S}')$ forment un cycle pour la relation $\longrightarrow_{\mathcal{S}^{\mathrm{num}}_{\phi}}$. Nous allons montrer que si $\xi_1 \longrightarrow_{\mathcal{S}^{\mathrm{num}}_{\phi}} \cdots \longrightarrow_{\mathcal{S}^{\mathrm{num}}_{\phi}} \xi_k \longrightarrow_{\mathcal{S}^{\mathrm{num}}_{\phi}} \xi_1$ est un cycle sans doublons de variables de $\mathsf{Vars}(\mathcal{S}')$, alors le point fixe associé aux variables $\xi_1, \ldots, \xi_k$ dans $\mathsf{Vars}(\mathcal{S}')$ est le même. De plus, si cet opérateur de point fixe est $\mu$, alors les parties droites des équations correspondantes contiennent uniquement des opérateurs positifs, et si l'opérateur est $\nu$, les parties droites des équations correspondantes contiennent uniquement des opérateurs négatifs. Comme chacune des variables de $\mathsf{Vars}(\mathcal{S}')$ participe au moins à un cycle sans doublons, et que chaque cycle sans doublons a au moins une variable en commun avec au moins un autre cycle sans doublons, c'est suffisant pour la preuve du lemme.

Soit donc un cycle sans doublons pour $\longrightarrow_{\mathcal{S}^{\mathrm{num}}_{\phi}}$, et soit $\xi_1 \xrightarrow{c_1}_{\mathsf{Eq}(\phi)} \cdots \xrightarrow{c_{k-1}}_{\mathsf{Eq}(\phi)} \xi_k \xrightarrow{c_k}_{\mathsf{Eq}(\phi)} \xi_1$ le cycle de variables de $\mathsf{Vars}(\mathsf{Eq}(\phi))$ extrait d'après le lemme A.2. D'après les propriétés de préservation des chemins et des opérateurs de point fixe établies dans ce même lemme, et rappelant que par construction de $\mathcal{S}_\phi$, pour chacune des variables $\xi_i$ pour $i$ dans $1..k$, l'opérateur de point fixe pour $\xi_i$ est le même dans $\mathsf{Eq}(\mathcal{S})$ et $\mathcal{S}^{\mathrm{num}}_{\phi}$, il suffit de montrer que les variables $\xi_i, \ldots, \xi_k$ ont le même point fixe dans $\mathsf{Eq}(\phi)$ et que les chemins $c_1, \ldots, c_k$ ne contiennent que des opérateurs d'un même type, positifs ou négatifs, en fonction de l'opérateur de point fixe. Pour chaque $i$ dans $1..k$, soit $\kappa_i$ l'opérateur de point fixe pour la variable $\xi_i$ dans $\mathsf{Eq}(\phi)$. Supposons sans perte de généralité que la variable $\xi_1$ est la plus grande parmi $\xi_1, \ldots, \xi_k$ pour la relation $\ll^*_{\mathsf{pos}(\phi)}$ et soit $c'$ le chemin $\kappa_1\xi_1.c_1.\ldots.\kappa_k\xi_k.c_k$. Alors, par le lemme A.4, $\kappa_1\xi_1.\phi'_1(c') = \xi_1$, où $\phi'_1$ est la formule telle que $\kappa_1\xi_1.\phi'_1$ est sous formule de $\mathsf{pos}(\phi)$. Rappelant que $c'$ est un chemin de dépendance, et donc sans opérateur $\alpha$, la conclusion suit du lemme 7.4, utilisant que $\phi$ est une formule LSrpd.

## Preuve du lemme 7.16

Nous commençons par mettre en évidence un principe de préservation des chemins de dépendance de variables entre $\mathcal{S}''_A$ et $\mathsf{formule}(\mathcal{S}''_A)$ : en utilisant la définition de $\mathsf{formule}(\mathcal{S}''_A)$ et

ce qui a été montré dans le lemme 4.12, on peut montrer que si $\mu\xi.\phi$ est sous formule de formule($\mathcal{S}''_A$) et que $c$ est un chemin tel que $\mu\xi.\phi(c) = \xi$, alors il existe une suite de variables $\xi_1, \ldots, \xi_n$ dans Vars($\mathcal{S}''_A$) deux à deux distinctes et il existe une suite de chemins $c_0, \ldots, c_n$ tels que $c = \mu\xi.c_0.\mu\xi_1.c_1. \ldots .\mu\xi_n.c_n.\xi$ satisfaisant

($\star$) pour tout $i$ dans $1..n-1$, $f_i(c_i) = x_{i+1}$ et $f(c_0) = c_1$ et $f_n(c_n) = \xi$,

où $f_1, \ldots, f_n$ sont les parties droites des équations pour $\xi_1, \ldots, \xi_n$ dans $\mathcal{S}''_A$. Donc, il suffit de montrer que dans $\mathcal{S}''_A$, pour toute suite de variables $\xi_1, \ldots, \xi_n$ deux à deux distinctes et pour toute suite de chemins $c_0, \ldots, c_n$ satisfaisant ($\star$), au moins un des $c_i$ contient un opérateur $\alpha$. C'est effectivement le cas puisque

- par définition de $\mathcal{S}''_A$, chaque partie droite d'équation dans $\mathcal{S}''_A$ est de la forme $\Gamma_{\mathcal{S}_A,\mathcal{X}_Q}(\phi')$ ;
- par définition, dans $\Gamma_{\mathcal{S}_A,\mathcal{X}_Q}$, à chaque variable $\xi_q$ est associée une formule qui est une disjonction de formules de la forme $\alpha[\xi_D]$ pour $\xi_D$ appartenant à $\mathcal{X}_{\mathcal{D}_A}$.

## A.4 Preuves des lemmes 8.6 et 8.8

**Définition A.5** Un ensemble d'arbres $T$ est dit *clos pour la relation de composant* si pour tout arbre $t$ dans $T$ et pour tout arbre $t'$ composant de $t$, $t'$ est aussi dans $T$.

Pour les démonstrations des lemmes 8.6 et 8.8, nous avons besoin du résultat intermédiaire annoncé ci-dessous.

**Lemme A.6** *Soient $\phi$ une formule et $\rho$ et $\delta$ des valuations telles que $\phi$ est close par $\rho$ et* libres$(\phi, \delta) \subseteq \{\xi\}$. *Soit $T$ un ensemble d'arbres clos pour la relation de composant. Alors :*

*(i) pour tout ensemble d'arbres $S$, $[\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap T = [\![\phi]\!]_{\rho,\delta[\xi\to S\cap T]} \cap T$ ;*

*(ii) les ensembles d'arbres $A = \bigcap\{S \mid [\![\phi]\!]_{\rho,\delta[\xi\to S]} \subseteq S\} \cap T$ et $B = \bigcap\{S \mid [\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap T \subseteq S\} \cap T$ sont égaux.*

*Preuve* Montrons d'abord que le point (i) du lemme implique le point (ii). Il est évident que si, pour un ensemble d'arbres $S$, on a $[\![\phi]\!]_{\rho,\delta[\xi\to S]} \subseteq S$, alors on a également $[\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap T \subseteq S$, et donc $B \subseteq A$. Soit maintenant $S$ un ensemble tel que $[\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap T \subseteq S$. Comme $t \in A$ implique $t \in T$, par définition de $B$ il suffit de montrer que $t$ appartient à $S$. Remarquons que $[\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap T \cap S \subseteq T \cap S$. Maintenant, d'après le point (i) du lemme, $[\![\phi]\!]_{\rho,\delta[\xi\to S]} \cap T \cap S = [\![\phi]\!]_{\rho,\delta[\xi\to S\cap T]} \cap T \cap S$ et donc, en posant $S' = S \cap T$, $[\![\phi]\!]_{\rho,\delta[\xi\to S']} \subseteq S'$. Par définition de $A$ et en utilisant que $t$ appartient à $A$, nous avons donc $t \in S' = S \cap T$, et donc $t$ appartient à S.

La preuve du point (i) se fait par induction sur la structure de la formule $\phi$.

Cas de base: Si $\phi$ est une des formules $\mathbf{0}$, $\top$, $X$, $\eta = \eta'$ ou $\xi'$ pour $\xi' \neq \xi$, alors par définition de la satisfiabilité les deux côtés de l'égalité à prouver sont égaux à $[\![\phi]\!]_{\rho,\delta} \cap T$. Si $\phi$ est la formule $\xi$, alors par définition de satisfiabilité les deux côtés sont égaux à $S \cap T$.

Cas d'induction: Si $\phi$ est la formule $\eta[\phi']$, par propriétés de la satisfiabilité, $t \in [\![\eta[\phi']]\!]_{\rho,\delta[\xi\to S]} \cap T$ si et seulement si (*) il existe un arbre $t'$ tel que $t = \{\!|\rho(\eta)[t']|\!\}$, $t' \in [\![\phi']\!]_{\rho,\delta[\xi\to S]}$ et $t \in T$. En utilisant que $T$ est clos pour la relation de composant, nous savons que $t \in T$ implique $t' \in T$, et donc dans (*) nous pouvons remplacer la condition $t' \in [\![\phi']\!]_{\rho,\delta[\xi\to S]}$ par $t' \in [\![\phi']\!]_{\rho,\delta[\xi\to S]} \cap T$. Maintenant par hypothèse de récurrence cette dernière est équivalente à $t' \in [\![\phi']\!]_{\rho,\delta[\xi\to S\cap T]} \cap T$. La conclusion est alors immédiate en utilisant les propriétés de la satisfiabilité.

Si $\phi$ est la formule $\phi' \mid \phi''$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\neg\phi'$, par hypothèse de récurrence $\complement([\![\phi']\!]_{\rho[\xi\to S]} \cap T) = \complement([\![\phi']\!]_{\rho[\xi\to S\cap T]} \cap T)$. Donc, $[\![\neg\phi']\!]_{\rho[\xi\to S]} \cup \complement T = [\![\neg\phi']\!]_{\rho[\xi\to S\cap T]} \cup \complement T$. Ceci implique que $([\![\neg\phi']\!]_{\rho[\xi\to S]} \cup \complement T) \cap T = ([\![\neg\phi']\!]_{\rho[\xi\to S\cap T]} \cup \complement T) \cap T$. Donc, $[\![\neg\phi']\!]_{\rho[\xi\to S]} \cap T = [\![\neg\phi']\!]_{\rho[\xi\to S\cap T]} \cap T$.

Si $\phi$ est la formule $\phi' \vee \phi''$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\exists x.\phi$, par définition de la satisfiabilité, il suffit de montrer que

$$\bigcup_{a\in\Sigma} [\![\phi]\!]_{\rho[x\to a],\delta[\xi\to S]} \cap T = \bigcup_{a\in\Sigma} [\![\phi]\!]_{\rho[x\to a],\delta[\xi\to S\cap T]} \cap T,$$

ce qui est immédiat en remarquant que par hypothèse de récurrence, pour toute étiquette $a$, $[\![\phi]\!]_{\rho[x\to a],\delta[\xi\to S]} \cap T = [\![\phi]\!]_{\rho[x\to a],\delta[\xi\to S\cap T]} \cap T$.

Si $\phi$ est la formule $\exists X.\phi$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\mu\xi'(M).\phi'$, on distingue deux cas. Si $\xi = \xi'$, alors la variable $\xi$ n'apparaît pas libre dans $\phi$ et donc les deux côtés de l'égalité à prouver sont égaux à $[\![\phi]\!]_{\rho,\delta} \cap T$. Si $\xi \neq \xi'$, par définition de la satisfiabilité il suffit de montrer que

$$\bigcap\{S' \mid S' \supseteq [\![\phi']\!]_{\rho,\delta[\xi\to S][\xi'\to S']} \cap \complement M\} \cap T = \bigcap\{S' \mid S' \supseteq [\![\phi']\!]_{\rho,\delta[\xi\to S\cap T][\xi'\to S']} \cap \complement M\} \cap T.$$

Comme $\xi \neq \xi'$, les valuations $\delta[\xi \to S][\xi' \to S']$ et $\delta[\xi' \to S'][\xi \to S]$ sont équivalentes et il en est de même pour les valuations $\delta[\xi \to S \cap T][\xi' \to S']$ et $\delta[\xi' \to S'][\xi \to S \cap T]$. Alors il suffit de montrer que pour chaque ensemble $S'$,

$$[\![\phi']\!]_{\rho,\delta[\xi'\to S'][\xi\to S]} = [\![\phi']\!]_{\rho,\delta[\xi'\to S'][\xi\to S\cap T]},$$

ce qui est vrai par hypothèse de récurrence appliquée à la formule $\phi'$ et les valuations $\rho$ et $\delta[\xi' \to S']$.

□

## Preuve du lemme 8.6

Pour tout arbre $t$, on note $\mathsf{Comp}(t)$ l'ensemble de ses composants. Nous montrons la propriété suivante : pour toute formule $\phi$ et pour tous arbres $s, s' \notin \mathsf{Comp}(t)$ il est vrai que $[\![\phi]\!]_{\rho[X\to s],\delta} \cap \mathsf{Comp}(t) = [\![\phi]\!]_{\rho[X\to s'],\delta} \cap \mathsf{Comp}(t)$. Il est facile de voir par propriétés de la satisfiabilité que cela implique le lemme 8.6. La démonstration se fait par récurrence sur la structure de la formule $\phi$.

Cas de base: Si $\phi$ est une formule parmi $\top$, $\eta = \eta'$, $\xi$ ou $\mathbf{0}$, on a $[\![\phi]\!]_{\rho[X\to s],\delta} = [\![\phi]\!]_{\rho,\delta} = [\![\phi]\!]_{\rho[X\to s'],\delta}$ et la propriété en suit immédiatement. Si $\phi$ est une variable $Y$, on distingue deux cas : si $X \neq Y$, alors on se ramène au cas précédent. Si $X = Y$, alors $[\![\phi]\!]_{\rho[X\to s],\delta} \cap \mathsf{Comp}(t) = \{s\} \cap \mathsf{Comp}(t)$. Comme $s \notin \mathsf{Comp}(t)$, ceci est égal à $\varnothing$. De même $[\![\phi]\!]_{\rho[X\to s'],\delta} \cap \mathsf{Comp}(t) = \{s'\} \cap \mathsf{Comp}(t) = \varnothing$.

Cas d'induction: Si $\phi$ est la formule $\eta[\phi']$, alors pour tout arbre $t_1$, $t_1 \in [\![\phi]\!]_{\rho[X\to s],\delta} \cap \mathsf{Comp}(t)$ si et seulement si $t_1 \in \mathsf{Comp}(t)$ et il existe un arbre $t_2$ tel que $t_1 = \{\!|\rho(\eta)[t_2]|\!\}$ et $t_2 \in [\![\phi']\!]_{\rho[X\to s],\delta}$. Par hypothèse de récurrence, ceci est vrai si et seulement si $t_2 \in [\![\phi']\!]_{\rho[X\to s'],\delta}$ et donc, par définition de la satisfiabilité, si et seulement si $t_1 \in [\![\phi]\!]_{\rho[X\to s'],\delta} \cap \mathsf{Comp}(t)$.

Si $\phi$ est la formule $\phi' \mid \phi''$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\neg\phi'$, par hypothèse de récurrence on sait que $\complement([\![\phi']\!]_{\rho[X\to s],\delta} \cap \mathsf{Comp}(t)) = \complement([\![\phi']\!]_{\rho[X\to s'],\delta} \cap \mathsf{Comp}(t))$. Donc, par définition de la satisfiabilité, $[\![\neg\phi']\!]_{\rho[X\to s],\delta} \cup \complement\mathsf{Comp}(t) = [\![\neg\phi']\!]_{\rho[X\to s'],\delta} \cup \complement\mathsf{Comp}(t)$. Ceci implique

$$([\![\neg\phi']\!]_{\rho[X\to s],\delta} \cup \complement\mathsf{Comp}(t)) \cap \mathsf{Comp}(t) = ([\![\neg\phi']\!]_{\rho[X\to s'],\delta} \cup \complement\mathsf{Comp}(t)) \cap \mathsf{Comp}(t).$$

d'où la conclusion est immédiate.

Si $\phi$ est la formule $\phi' \vee \phi''$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\exists x.\phi'$, par définition de la satisfiabilité il suffit de montrer que

$$\bigcup_{a\in\Sigma} [\![\phi']\!]_{\rho[X\to s][x\to a],\delta} \cap \mathsf{Comp}(t) = \bigcup_{a\in\Sigma} [\![\phi']\!]_{\rho[X\to s'][x\to a],\delta} \cap \mathsf{Comp}(t),$$

ce qui est vrai par hypothèse de récurrence appliquée à la formule $\phi'$ et aux valuations $\rho[x \rightarrow a], \delta$ pour toute étiquette a dans $\Sigma$.

Si $\phi$ est la formule $\exists Y.\phi'$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\mu\xi(M).\phi'$ nous avons la suite d'égalités d'ensembles suivante :

$$
\begin{aligned}
& [\![\phi]\!]_{\rho[X\rightarrow s],\delta} \cap \mathsf{Comp}(t) && \text{(déf. de la satisfiabilité)}\\
= \ & \bigcap \{S \mid [\![\phi']\!]_{\rho[X\rightarrow s],\delta[\xi\rightarrow S]} \cap \complement M \subseteq S\} \cap \mathsf{Comp}(t) && \text{(point (ii) du lemme A.6)}\\
= \ & \bigcap \{S \mid ([\![\phi']\!]_{\rho[X\rightarrow s],\delta[\xi\rightarrow S]} \cap \complement M) \cap \mathsf{Comp}(t) \subseteq S\} \cap \mathsf{Comp}(t)\\
= \ & \bigcap \{S \mid ([\![\phi']\!]_{\rho[X\rightarrow s],\delta[\xi\rightarrow S]} \cap \mathsf{Comp}(t)) \cap \complement M \subseteq S\} \cap \mathsf{Comp}(t) && \text{(hypothèse de récurrence)}\\
= \ & \bigcap \{S \mid ([\![\phi']\!]_{\rho[X\rightarrow s'],\delta[\xi\rightarrow S]} \cap \mathsf{Comp}(t)) \cap \complement M \subseteq S\} \cap \mathsf{Comp}(t)\\
= \ & \bigcap \{S \mid ([\![\phi']\!]_{\rho[X\rightarrow s'],\delta[\xi\rightarrow S]} \cap \complement M) \cap \mathsf{Comp}(t) \subseteq S\} \cap \mathsf{Comp}(t) && \text{(point (ii) du lemme A.6)}\\
= \ & \bigcap \{S \mid ([\![\phi']\!]_{\rho[X\rightarrow s'],\delta[\xi\rightarrow S]} \cap \complement M) \subseteq S\} \cap \mathsf{Comp}(t) && \text{(déf. de la satisfiabilité)}\\
= \ & [\![\phi]\!]_{\rho[X\rightarrow s'],\delta} \cap \mathsf{Comp}(t)
\end{aligned}
$$

## Preuve du lemme 8.8

Pour deux étiquettes $a$ et $b$, on note $\mathcal{A}_{\overline{ab}}$ l'ensemble des arbres dont aucune des arêtes n'est étiquetée ni par $a$ ni par $b$. Formellement, $\mathcal{A}_{\overline{ab}} = \{t \mid a, b \notin \mathit{etiq}(t)\}$. Nous montrons que pour toute formule $\phi$ et toutes valuations $\rho$ et $\delta$ telles que $\phi$ est close par $\delta$ et pour toutes étiquettes $a, b$ n'appartenant pas à $\mathit{etiq}(\phi, \rho, \delta)$, il est vrai que $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}} = [\![\phi]\!]_{\rho\overline{\rho_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}}$, où $\rho_{ab}$ est une valuation (n'importe laquelle) de domaine libres$(\phi, \rho)$ et telle que pour tout $x$ dans $\mathit{dom}(\rho_{ab}) \cap \mathcal{X}_e$, $\rho_{ab}(x) \in \{a, b\}$. Il est facile de voir que le lemme 8.8 découle de cette propriété. En effet, pour tout arbre $t$, en utilisant les définitions de la satisfiabilité et de $\mathcal{A}_{\overline{ab}}$, $t$ appartient à $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}}$ si et seulement si $t, \rho\rho_{ab}, \delta \models \phi$ et $a, b \notin \mathit{etiq}(t)$. De la même façon, $t$ appartient à $[\![\phi]\!]_{\rho\overline{\rho_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}}$ si et seulement si $t, \rho\overline{\rho_{ab}}, \delta \models \phi$ et $a, b \notin \mathit{etiq}(t)$.

La preuve de $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}} = [\![\phi]\!]_{\rho\overline{\rho_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}}$ se fait par récurrence sur la structure de la formule $\phi$.

Cas de base: Notons d'abord que si la formule $\phi$ ne contient aucun des éléments de $\mathit{dom}(\rho_{ab})$ comme variable libre, alors la propriété est conséquence immédiate du fait que $[\![\phi]\!]_{\rho\rho_{ab},\delta} = [\![\phi]\!]_{\rho,\delta} = [\![\phi]\!]_{\rho\overline{\rho_{ab}},\delta}$. Ce cas s'applique lorsque $\phi$ est l'une des formules $\top$, $\xi$, $\mathbf{0}$, $X$ avec $X \notin \mathit{dom}(\rho_{ab})$ et $\eta = \eta'$ avec $\eta, \eta' \notin \mathit{dom}(\rho_{ab})$. Regardons maintenant les cas de base qui ne rentrent pas dans cette catégorie.

Si $\phi$ est la variable d'arbre $X$ avec $X \in \mathit{dom}(\rho_{ab})$, on examine deux cas. Si l'arbre $\rho_{ab}(X)$ ne contient pas les étiquettes $a$ et $b$ alors, par définition de $\overline{\rho_{ab}}$, c'est également le cas de $\overline{\rho_{ab}}(X)$ et donc $\rho\overline{\rho_{ab}}(X) = \rho\rho_{ab}(X)$. La conclusion est immédiate de la définition de la satisfiabilité. Si $\rho_{ab}(X)$ contient l'une des étiquettes $a$ ou $b$, alors c'est aussi le cas de $\overline{\rho_{ab}}(X)$ et donc les deux ensembles $[\![X]\!]_{\rho\overline{\rho_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}}$ et $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}}$ sont égaux puisque vides.

Si $\phi$ est la formule $\eta = \eta'$ avec $\eta$ étant une variable d'étiquette dans $\mathit{dom}(\rho_{ab})$ et $\eta'$ étant une étiquette alors $\rho_{ab}(\eta)$ et $\overline{\rho_{ab}}(\eta)$ sont tous les deux égaux à $a$ ou $b$, tandis que l'étiquette $\eta'$ est nécessairement différente de $a$ et $b$ puisque $a, b \notin \mathit{etiq}(\phi, \rho, \delta)$. Alors, par définition de la satisfiabilité pour la formule $\eta = \eta'$, $[\![\phi]\!]_{\rho\rho_{ab},\delta} = [\![\phi]\!]_{\rho\overline{\rho_{ab}},\delta} = \varnothing$. Le cas où $\eta$ est une étiquette et $\eta' \in \mathit{dom}(\rho_{ab})$ est symétrique.

Finalement, si $\eta$ et $\eta'$ sont deux variables dans $dom(\rho_{ab})$, alors par définition de $\rho_{ab}$ et $\overline{\rho_{ab}}$ il est facile de voir que $\rho_{ab}(\eta)$ est égal à $\rho_{ab}(\eta')$ si et seulement si $\overline{\rho_{ab}}(\eta)$ est égal à $\overline{\rho_{ab}}(\eta')$. La conclusion s'en suit par définition de la satisfiabilité pour la formule $\eta = \eta'$.

Cas d'induction: Si $\phi$ est la formule $\eta[\phi']$, alors par définition de la satisfiabilité et pour tout arbre $t$, $t$ appartient à $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}}$ si et seulement si (i) il existe un arbre $t'$ tel que $t = \{\!|\rho\rho_{ab}(\eta)[t']|\!\}$, (ii) $t'$ est dans $[\![\phi']\!]_{\rho\rho_{ab},\delta}$ et (iii) $t$ est dans $\mathcal{A}_{\overline{ab}}$. Maintenant, par définition de $\mathcal{A}_{\overline{ab}}$, $t$ est dans $\mathcal{A}_{\overline{ab}}$ si et seulement si $\rho\rho_{ab}(\eta) \notin \{a, b\}$ et $t'$ est dans $\mathcal{A}_{\overline{ab}}$. De plus, par hypothèse de récurrence, $t'$ est dans $[\![\phi']\!]_{\rho\rho_{ab},\delta}$ et $t'$ dans $\mathcal{A}_{\overline{ab}}$ si et seulement si $[\![\phi']\!]_{\rho\overline{\rho_{ab}},\delta}$ et $t'$ dans $\mathcal{A}_{\overline{ab}}$. La conclusion est maintenant facile en utilisant de nouveau la définition de $\mathcal{A}_{\overline{ab}}$ et de la satisfiabilité.

Si $\phi$ est la formule $\phi' \,|\, \phi''$, la preuve est similaire au cas précédent.

Si $\phi$ est la formule $\neg\phi'$, nous savons par hypothèse de récurrence que $\complement([\![\phi']\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}}) = \complement([\![\phi']\!]_{\rho\overline{\rho_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}})$. Alors, par définition de la satisfiabilité, $[\![\neg\phi']\!]_{\rho\rho_{ab},\delta} \cup \complement\mathcal{A}_{\overline{ab}} = [\![\neg\phi']\!]_{\rho\overline{\rho_{ab}},\delta} \cup \complement\mathcal{A}_{\overline{ab}}$. Nous concluons en calculant l'intersection des deux côtés de cette égalité par $\mathcal{A}_{\overline{ab}}$.

Si $\phi$ est la formule $\phi' \vee \phi''$, comme dans le cas précédent la preuve s'obtient simplement en utilisant la définition de satisfiabilité et l'hypothèse de récurrence et quelques propriétés simples des ensembles.

Si $\phi$ est la formule $\exists x.\phi'$, en utilisant la définition de la satisfiabilité, il est facile de voir que $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}}$ est égal à $\bigcup_{c\in\Sigma}([\![\phi']\!]_{\rho\rho_{ab}[x\to c],\delta} \cap \mathcal{A}_{\overline{ab}})$, ce qui est égal à l'union des trois ensembles $S_1 = \bigcup_{c\in\Sigma\setminus\{a,b\}}[\![\phi']\!]_{\rho\rho_{ab}[x\to c],\delta} \cap \mathcal{A}_{\overline{ab}}$, $S_2 = [\![\phi']\!]_{\rho\rho_{ab}[x\to a],\delta} \cap \mathcal{A}_{\overline{ab}}$ et $S_3 = [\![\phi']\!]_{\rho\rho_{ab}[x\to b],\delta} \cap \mathcal{A}_{\overline{ab}}$. En remarquant que pour tout $c$ dans $\Sigma\setminus\{a, b\}$, $a, b \notin etiq(\phi', \rho[x \to c], \delta)$, nous pouvons appliquer l'hypothèse de récurrence à la formule $\phi'$ et les valuations $\rho[x \to c]$ et $\delta$, ce qui nous permet de déduire que l'ensemble $S_1$ est égal à l'ensemble $T_1 = \bigcup_{c\in\Sigma\setminus\{a,b\}}[\![\phi']\!]_{\rho\overline{\rho_{ab}}[x\to c],\delta} \cap \mathcal{A}_{\overline{ab}}$. Soient maintenant ${\rho'}_{ab}$ et ${\rho''}_{ab}$ les valuations $\rho_{ab}[x \to a]$ et $\rho_{ab}[x \to b]$ respectivement, dont le domaine est $dom(\rho) \cup \{x\} = \text{libres}(\phi', \rho)$. En appliquant l'hypothèse de récurrence sur la formule $\phi'$ et les valuations ${\rho'}_{ab}$ et $\delta$ et d'une part et sur la formule $\phi'$ et les valuations ${\rho''}_{ab}$ et $\delta$ d'autre part, nous obtenons que $S_2$ est égal à l'ensemble $T_2 = [\![\phi']\!]_{\overline{\rho\rho'_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}}$ et $S_3$ est égal à l'ensemble $T_3 = [\![\phi']\!]_{\overline{\rho\rho''_{ab}},\delta} \cap \mathcal{A}_{\overline{ab}}$. Maintenant, par définition de $\overline{\rho'_{ab}}$ et $\overline{\rho''_{ab}}$ nous avons que $\overline{\rho'_{ab}} = \overline{\rho_{ab}}[x \to b]$ et $\overline{\rho''_{ab}} = \overline{\rho_{ab}}[x \to a]$. Il est maintenant facile de voir, en utilisant la définition de satisfiabilité, que $T_1 \cup T_2 \cup T_3 = [\![\exists x.\phi']\!]_{\rho\overline{\rho_{ab}},\delta}$. La conclusion découle alors du fait que $T_1 \cup T_2 \cup T_3 = S_1 \cup S_2 \cup S_3 = [\![\exists x.\phi']\!]_{\rho\rho_{ab},\delta}$.

Si $\phi$ est la formule $\exists X.\phi$, en utilisant la définition de la satisfiabilité, il est facile de voir que $[\![\phi]\!]_{\rho\rho_{ab},\delta} \cap \mathcal{A}_{\overline{ab}}$ est égal à l'union des deux ensembles $S_1 = \bigcup_{t\in\mathcal{A}_{\overline{ab}}}[\![\phi']\!]_{\rho\rho_{ab}[X\to t],\delta} \cap \mathcal{A}_{\overline{ab}}$ et $S_2 = \bigcup_{t\in\complement\mathcal{A}_{\overline{ab}}}[\![\phi']\!]_{\rho\rho_{ab}[X\to t],\delta} \cap \mathcal{A}_{\overline{ab}}$. En remarquant que pour $t \in \mathcal{A}_{\overline{ab}}$, $a, b \notin etiq(\phi', \rho[X \to t], \delta)$, l'hypothèse de récurrence appliquée à la formule $\phi'$ et les valuations $\rho[X \to t]$ et $\delta$ nous permet d'établir que $S_1$ est égal à l'ensemble $T_1 = \bigcup_{t\in\mathcal{A}_{\overline{ab}}}[\![\phi']\!]_{\rho\overline{\rho_{ab}}[X\to t],\delta} \cap \mathcal{A}_{\overline{ab}}$. Pour chaque $t$ dans $\complement\mathcal{A}_{\overline{ab}}$, soit ${\rho^t}_{ab}$ la valuation $\rho_{ab}[X \to t]$ de domaine $dom(\rho_{ab}) \cup \{X\} = \text{libres}(\phi', \rho)$. En utilisant l'hypothèse de récurrence sur la formule $\phi'$ et les valuations $\rho$ et $\delta$, il est facile de déduire que $S_2$ est égal à l'ensemble $T_2 = \bigcup_{t\in\complement\mathcal{A}_{\overline{ab}}}[\![\phi']\!]_{\overline{\rho{\rho^t}_{ab}},\delta}$. Par définition de $\overline{{\rho^t}_{ab}}$, pour chaque arbre $t$, $\overline{{\rho^t}_{ab}} = \overline{\rho_{ab}}[X \to t\{a \leftrightarrow b\}]$. Donc, $T_2 = \bigcup_{t\in T}[\![\phi']\!]_{\rho\overline{\rho_{ab}}[X\to t],\delta}$, où $T$ est l'ensemble d'arbres $\{t\{a \leftrightarrow b\} \mid t \in \complement\mathcal{A}_{\overline{ab}}\}$. Il est facile de déduire de la définition de $\mathcal{A}_{\overline{ab}}$ que $T = \complement\mathcal{A}_{\overline{ab}}$, et on en déduit que $T_2 = \bigcup_{t\in\complement\mathcal{A}_{\overline{ab}}}[\![\phi']\!]_{\rho\overline{\rho_{ab}}[X\to t],\delta}$. Maintenant, par définition de satisfiabilité, il est facile de voir que $T_1 \cup T_2 = [\![\exists X.\phi']\!]_{\rho\overline{\rho_{ab}},\delta}$. La conclusion découle alors du fait que $S_1 = T_1$ et $S_2 = T_2$.

Si $\phi$ est la formule $\mu\xi(M).\phi'$, la preuve est identique à celle du lemme 8.6 et le cas d'induction pour $\phi = \mu\xi(M).\phi'$, en remplaçant dans cette preuve $\rho[X \to s]$ par $\rho\rho_{ab}$, $\rho[X \to s']$ par $\rho\overline{\rho_{ab}}$ et $\mathsf{Comp}(t)$ par $\mathcal{A}_{\overline{ab}}$.

# Index