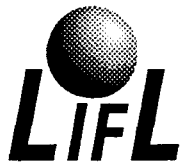


Laboratoire d'Informatique Fondamentale de Lille

THÈSE

Nouveau Régime

Présentée à

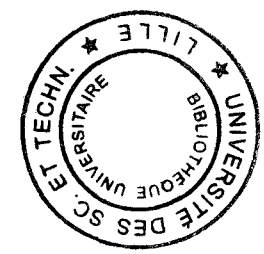
L'UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Pour obtenir le titre de

DOCTEUR EN INFORMATIQUE

par

Franck SEYNHAEVE



AUTOMATES, RÉÉCRITURE ET CONTRAINTES : RÉSULTATS DE DÉCIDABILITÉ ET D'INDÉCIDABILITÉ

Thèse soutenue le [illegible] devant la Commission d'Examen :

Président	:	Mireille CLERBOUT	Université de Lille I
Rapporteurs	:	Hubert COMON	E.N.S. Cachan
		Denis LUGIEZ	Université d'Aix-Marseille I
		Ralf TREINEN	Université de Paris XI
Examinateurs	:	Sophie TISON	Université de Lille I
		Marc TOMMASI	Université de Lille III

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
U.F.R. d'I.E.E.A. Bât. M3. 59655 Villeneuve d'Ascq CEDEX
Tél. 03.20.43.47.24 Fax. 03.20.43.65.66

Remerciements

Comme grand nombre de mes prédécesseurs, j'ai attendu cet instant avec impatience. Ecrire cette page m'emplit de joie car elle marque la fin de quatre années de travaux et me permet de remercier toutes les personnes qui m'ont aidé durant toutes ces années. Mais une certaine angoisse m'envahit également par peur d'oublier de citer certaines de ces personnes. Je tiens donc à m'excuser par avance auprès de celles-ci.

Je tiens tout d'abord à remercier Mireille Clerbout d'avoir bien voulu présider le jury de ma soutenance.

Je remercie également Hubert Comon, Denis Lugiez et Ralf Treinen d'avoir accepté d'être rapporteurs de cette thèse. Je les remercie du temps qu'ils ont passé à la lire, de l'intêret qu'ils ont porté à ce travail et de leurs nombreuses remarques et commentaires.

Je remercie tout particulièrement Sophie Tison et Marc Tommasi de m'avoir suivi tout au long de cette thèse. Merci pour leur disponibilité, leur soutien, leurs conseils et leur bonne humeur quotidienne.

Merci également à Bruno Bogaert, Anne-Cécile Caron et Max Dauchet du L.I.F.L. et à Sándor Vágvölgyi de l'université József Attila de Szeged en Hongrie avec qui j'ai eu le plaisir de collaborer.

Merci à tous les membres du L.I.F.L. qui font de ce lieu un endroit très agréable où travail et joie de vivre se marient à merveille.

Enfin je remercie très sincérement tous ceux qui partagent ma vie de tous les jours (mes amis, ma famille et tout particulièrement mes parents) car sans leur soutien je n'aurais certainement pas eu l'occassion de mettre ce dernier point.

Table des matières

Introduction

Cette thèse présente un ensemble de contributions à l'étude des automates à contraintes, de la réécriture et des contraintes ensemblistes.

Le point de départ de nos travaux est la recherche d'outils de décision les plus puissants possibles. Lors de nos études, nous avons défini un outil de preuves d'indécidabilité que nous nommons la méthode des Arbres-Grilles. Cette méthode est une adaptation aux arbres de la méthode de la grille qui est un outil classique de preuves d'indécidabilité. Notre méthode nous permet de montrer un résultat d'indécidabilité dans chacun des domaines cités ci-dessus. Ces résultats concernent le problème du vide pour les automates à tests entre cousins (sous-termes à la profondeur 2), la décidabilité des formules de la théorie du premier ordre de la réécriture en un pas et la décidabilité des formules de la théorie du premier ordre des contraintes ensemblistes. Par la suite, nous avons poursuivi nos recherches dans chacun de ces domaines en essayant de réduire la frontière entre la décidabilité et l'indécidabilité de chacun des problèmes précédents.

Automates d'arbres à contraintes, Réécriture et Contraintes ensemblistes

Automates. [20] Les automates d'arbres ont été créés, il y a quelques années, dans le cadre de la vérification de circuits. Plusieurs chercheurs ont contribué à cette école menée par A. Church dans la fin des années 50 et au début des années 60: B. Trakhtenbrot, J.R. Büchi, M.O. Rabin, Doner, Thatcher, etc ... Nombre de nouvelles idées surgirent de ce programme. Par exemple, les connexions entre les automates et la logique. La théorie des langages d'arbres reconnaissables s'est développée dans les années 70 et de nombreux résultats furent établis à cette période. Un peu oubliés durant les années 80, les automates d'arbres ont connu un regain d'intérêt ces dernières années pour plusieurs raisons.

Tout d'abord, ils constituent un modèle de calcul simple pour lequel la plupart des propriétés sont décidables avec une complexité praticable. Par exemple, pour un automate $\mathcal{A}$ donné, la question « Est-ce que le langage reconnu par $\mathcal{A}$ est vide? » (problème du vide) est décidable en temps linéaire.

Les automates d'arbres constituent en particulier un outil de décision pour certaines théories logiques, dont la plus remarquable est la théorie monadique faible du second ordre à deux successeurs, ou théorie (faible) de Rabin WS2S. Les propriétés de clôture des automates d'arbres permettent une construction incrémentale d'un système de décision associé à une formule et d'autre part, les opérations de déterminisation, minimisation et nettoyage fournissent des moyens d'optimisation indépendants du type de logique considéré.

Les automates d'arbres sont donc de bons outils de décision mais ils ne peuvent distinguer les termes non linéaires des termes linéaires. Un terme est non linéaire s'il contient deux

occurrences d'une même variable. Par exemple, l'ensemble des termes de la forme $f(t,t)$ n'est pas reconnaissable par un automate d'arbres. En effet, dans un automate, les deux fils de la racine du terme sont reconnus indépendamment et seules des informations de taille finie peuvent être remontées jusqu'à la racine, tandis que t peut être de taille arbitrairement grande. Puisque les termes non linéaires apparaissent très souvent, par exemple en logique, des extensions des automates d'arbres ont été définies afin de prendre en compte la non linéarité des termes. Et donc de pouvoir utiliser des techniques d'automates d'arbres sur des problèmes quelconques (comme la réductibilité inductive).

En 1981, M. Dauchet et J. Mongy [65], ont défini une nouvelle classe d'automates appelés *RATEG*. Dans ces automates, les membres des règles de transition peuvent être non linéaires et contenir des variables à n'importe quelle profondeur. Les règles permettent donc d'imposer des égalités entre sous-termes. Ces automates reconnaissent par exemple l'ensemble des termes clos réductibles par un système de réécriture. La classe des langages reconnus par ces automates conserve les propriétés de clôture par union et intersection de la classe des langages reconnus par les automates d'arbres précédents, dits standards. En revanche, le problème du vide est indécidable pour les *RATEG*, ils ne peuvent donc être utilisés comme outils de décision.

Les automates *RATEG* permettant uniquement de vérifier des égalités, une nouvelle classe d'automates, dits automates à contraintes, a été définie afin d'effectuer à la fois des tests d'égalité et des tests de différence. Ces automates conservent la propriété de clôture par opérations booléennes des automates d'arbres standards. Par contre le problème du vide est indécidable pour ces automates. Afin de pouvoir utiliser les automates à contraintes comme outils de décision, certaines sous-classes d'automates à contraintes pour lesquelles le problème du vide est décidable ont été définies. Les classes les plus remarquables sont :

- *La classe des automates à tests entre frères (ATF).*

  Ces automates autorisent des tests d'égalité et de différence entre sous-termes à la profondeur 1. Par exemple, l'ensemble des termes de la forme $f(t,t)$ est reconnu par un tel automate. Cette classe est intéressante puisque la plupart des propriétés des automates d'arbres standards peuvent y être étendues en remplaçant les conditions de linéarité par des restrictions sur les non linéarités.

- *La classe des automates de réduction (AR).*

  Ces automates, introduits par M. Dauchet, A.-C. Caron et J.-L. Coquidé [23], autorisent un nombre arbitraire de différences mais un nombre fini d'égalités dans chaque calcul des automates. Les automates de réduction permettent en particulier de reconnaître l'ensemble des termes en forme normale d'un système de réécriture quelconque. La décidabilité du problème du vide pour les AR permet de montrer que le problème de satisfiabilité de la théorie de la réduction est décidable.

  Par la suite [15], M. Dauchet, A.-C. Caron et J.-L. Coquidé, en collaboration avec H. Comon et F. Jacquemard, ont généralisé les automates de réduction en autorisant un nombre quelconque de tests d'égalité entre positions frères. Pour ces automates de réduction généralisés (ARG) le problème du vide reste décidable.

Réécriture. La réécriture de termes est une branche de l'informatique théorique qui combine des éléments de logique, d'algèbre universelle, de preuve automatique et de programma-

tion fonctionnelle. La base de celle-ci est la logique équationnelle. La distinction entre la réécriture de termes et la logique équationnelle réside dans le fait que les équations en réécriture sont utilisées comme des règles de remplacement orientées, c'est-à-dire que le membre gauche peut être remplacé par le membre droit, mais pas l'inverse. Par exemple, $f(x,y) \to g(x)$ est une règle de réécriture qui permet de remplacer le motif $f(x,y)$ par le motif $g(x)$.

Les systèmes de réécriture sont utilisées pour la simplification dans toutes les manipulations symboliques, qu'il s'agisse de calcul formel, démonstration automatique, transformation de programmes en précompilation, évaluation en programmation fonctionnelle ...

Par exemple, en démonstration automatique dans des théories égalitaires, pour prouver la validité d'une équation $e = e'$, il suffit de remplacer des égaux par des égaux, c'est-à-dire remplacer des sous-termes de e et e' suivant les égalités d'un ensemble d'équations $\mathcal{E}$, jusqu'à obtenir deux termes syntaxiquement identiques. Dans certains cas, on peut associer à un ensemble d'équations $\mathcal{E}$ un système de réécriture $\mathcal{R}$ par un mécanisme de complétion, le système $\mathcal{R}$ fournissant une stratégie orientée de remplacement, par laquelle on peut calculer des représentants de classes de la congruence engendrée par $\mathcal{E}$ et donc prouver la validité d'équations.

La théorie du premier ordre de la réécriture en un pas pour un système $\mathcal{R}$ sur une signature Σ, et un ensemble de variables $\mathcal{X}$ est la théorie du premier ordre dont la structure est la suivante : son support est l'ensemble des termes clos et son seul prédicat est la relation binaire $\to_{\mathcal{R}}$. Une interprétation associe à chaque variable de $\mathcal{X}$ un terme de $T(\Sigma)$ et la relation $x \to_{\mathcal{R}} y$ est interprétée par "x se réécrit en y en un pas par $\mathcal{R}$". D'importantes propriétés décidables pour des systèmes de réécriture quelconques s'expriment dans cette théorie.

Considérons par exemple la propriété décidable de confluence forte. Tout d'abord, un système de réécriture $\mathcal{R}$ est fortement confluent (définition issue de [26]) si

$$\forall x, y_1, y_2, (x \to_{\mathcal{R}} y_1 \wedge x \to_{\mathcal{R}} y_2 \Rightarrow \exists z (y_1 \xrightarrow{=}_{\mathcal{R}} z \wedge y_2 \xrightarrow{=}_{\mathcal{R}} z))$$

où $x \xrightarrow{=} y$ signifie que $x = y \vee x \to y$. Il est fortement confluent clos si la propriété ci-dessus est satisfaite quand x est restreint à prendre ses valeurs dans l'ensemble des termes clos. On peut montrer que $\mathcal{R}$ est fortement confluent si et seulement s'il est fortement confluent clos dans une signature étendue par de nouvelles constantes. Cette propriété peut être exprimée dans la théorie de la réécriture en un pas, puisque $\mathcal{R}$ est fortement confluent clos si et seulement si la formule

$$\forall x, y_1, y_2, (x \to y_1 \wedge x \to y_2 \Rightarrow \exists z (y_1 \to z \wedge y_2 \to z))$$

est valide pour le système de réécriture $\mathcal{R} \cup \{x \to x\}$.

Considérons maintenant le problème de réductibilité inductive. Un terme t est dit inductivement réductible pour un système de réécriture $\mathcal{R}$ si toutes ses instances closes sont réductibles par $\mathcal{R}$. Cette propriété peut s'exprimer en considérant plusieurs systèmes de réécriture. En effet, si on note S le système $\{t \to t\}$, alors t est inductivement réductible si la formule

$$\forall x \; (x \to_S x \; \Rightarrow \; \exists y \; (x \to_{\mathcal{R}} y))$$

est valide. La décidabilité de la réductibilité inductive a été montrée par D. Plaisted [70].

Cette possibilité d'exprimer ces propriétés décidables permettait d'espérer que la théorie de la réécriture en un pas était décidable. Mais cette conjecture s'avéra fausse lorsque R. Treinen [89] montra l'indécidabilité du fragment $\exists^*\forall^*$ de la théorie de la réécriture en un pas pour tout système de réécriture. Par la suite, plusieurs chercheurs s'intéressèrent à ce

problème. R. Treinen [88] affina son résultat au fragment $\exists^*\forall^*$ pour les systèmes de réécriture linéaires par codage du problème de correspondance de Post, J. Marcinkowski [61] au fragment $\exists^*\forall^*$ pour les systèmes nœthériens clos à droite par codage des arbres Thue réguliers (outils définis dans [62] et [60]), et S. Vorobyov [94] au fragment $\exists^*\forall^*\exists^*$ pour les systèmes linéaires et nœthériens par codage du problème de correspondance de Post.

La décidabilité du fragment existentiel de la théorie de la réécriture en un pas est un problème ouvert. Niehren *et al.* [66] ont tout de même montré que le fragment existentiel positif de cette théorie est décidable. En fait, toute formule de ce fragment est équivalente en terme de solutions à un problème d'unification du second ordre stratifiée, les problèmes de ce type étant montrés décidables par M. Schmidt-Schauss [77]. Dans [46], F. Jacquemard transforme toute formule du fragment existentiel positif de la théorie de la réécriture en un pas en une formule décidable de la théorie faible monadique à deux successeurs, mais ce résultat n'est valable que pour les systèmes de réécriture tels que pour toute règle $l \to r$, toutes les occurrences d'une même variable dans $l \to r$ sont à la même profondeur.

Contraintes ensemblistes. Les contraintes ensemblistes sont des inclusions entre expressions désignant des ensembles de termes ($exp \subseteq exp'$ ou $exp \not\subseteq exp'$), de telles expressions (exp, exp') étant des termes sur une signature Σ, un ensemble de variables $\mathcal{X}$ et des opérateurs ensemblistes ($\cup$, $\cap$, $\mathbf{C}$, etc ...).

L'interprétation des opérateurs ensemblistes est canonique c'est-à-dire que le symbole $\cup$ (respectivement $\cap$, $\mathbf{C}$) est interprété comme l'union (respectivement l'intersection, le complément) ensembliste. Une interprétation $\mathcal{I}$ d'une contrainte ensembliste associe à chaque variable de $\mathcal{X}$ un ensemble de termes clos. Une telle interprétation satisfait une contrainte positive $exp \subseteq exp'$ (respectivement négative $exp \not\subseteq exp'$) si $\mathcal{I}(exp) \subseteq \mathcal{I}(exp')$ (respectivement $\mathcal{I}(exp) \not\subseteq \mathcal{I}(exp')$).

Les contraintes ensemblistes, compte-tenu de leur pouvoir d'expression et de leur simplicité, sont à la base d'une technique d'analyse de programmes, appelée analyse ensembliste [74, 49, 64, 42, 3, 40]. Cette technique peut se résumer en deux étapes: la première est la proposition d'un mécanisme d'extraction de contraintes ensemblistes à partir d'un programme et la preuve de correction de ce mécanisme vis-à-vis de la sémantique du problème. La seconde étape est la proposition d'un algorithme de résolution de ces contraintes sous la forme de calcul d'une solution particulière ou d'un test de satisfiabilité afin d'obtenir des informations utiles sur le programme (par exemple, vérification de sortes). Par exemple, si l'ensemble des entiers naturels est défini comme le plus petit ensemble contenant 0 et le successeur de tout entier (noté s) alors la contrainte suivante formalise cette définition :

$$Nat = 0 \cup s(Nat).$$

En programmation fonctionnelle ou logique, le contrôle de sortes est souvent dynamique. En d'autres termes, la procédure qui se charge de vérifier si une expression est bien sortée intervient durant l'exécution. De grandes libertés sont donc laissées au programmeur aux dépens de la sécurité et de l'efficacité. Pour pallier à ces inconvénients, des procédures d'extraction et de vérification de sortes sont ajoutées à la phase de compilation. On obtient alors un système riche d'informations qui pourront être utilisées pour l'optimisation du programme.

Le texte du programme est analysé automatiquement et les informations extraites sont représentées dans un formalisme adéquat. Si les sortes sont considérées comme des ensembles

de valeurs, alors les expressions ensemblistes sont bien adaptées pour représenter ces valeurs et les contraintes pour exprimer leurs relations. De nombreux algorithmes d'inférence de types en programmation fonctionnelle, logique ou impérative évoluent autour d'un noyau qui consiste en la résolution de contraintes ensemblistes.

Les algorithmes les plus anciens manipulent des contraintes au pouvoir d'expression assez pauvre. Afin d'obtenir des informations plus consistantes, il est nécessaire d'enrichir le vocabulaire des contraintes. Plus ce vocabulaire est riche, plus l'analyse aura les moyens d'être précise et pertinente, mais aussi, plus les solutions seront difficiles à trouver.

Néanmoins pour que l'analyse garde un sens, une propriété essentielle doit être préservée: la décidabilité de la satisfiabilité, c'est-à-dire il doit exister une procédure qui détermine si un système de contraintes ensemblistes possède ou non des solutions. En d'autres termes, il faut que les informations extraites permettent de dire si les objets du programme étudié possèdent un type.

Au cours de nos travaux, nous avons étudié la décidabilité de la théorie du premier ordre des contraintes ensemblistes c'est-à-dire l'ensemble des formules du premier ordre dont les formules atomiques sont les contraintes ensemblistes.

La théorie complète est indécidable en raison de l'indécidabilité de la théorie des algèbres libres finiment engendrées [83], par contre son fragment existentiel est décidable. Plusieurs approches ont été abordées pour établir ce résultat. A. Aiken, D. Kozen, et E. Wimmers [2] réduisent la décidabilité du fragment existentiel de la théorie des contraintes ensemblistes dans le problème d'accessibilité non linéaire pour les inégalités diophantiennes par utilisation des hypergraphes. R. Gilleron, M. Tommasi et S. Tison [37] réduisent ce problème dans le problème du vide pour une nouvelle classe d'automates (les automates d'ensembles d'arbres généralisés). Dans [18], W. Charatonik et L. Pacholski montrent que le fragment existentiel de la théorie des contraintes ensemblistes avec symboles de projection est décidable. Récemment, J.M. Talbot [84] a montré l'indécidabilité du fragment $\forall\exists^*$ de la théorie des contraintes sans opérateur ensembliste par réduction du problème de correspondance de Post.

Présentation des résultats et plan de la thèse

Cette thèse se découpe en cinq parties, chacune d'entre elles divisée en plusieurs chapitres. Ces parties correspondent essentiellement aux différents domaines abordés dans nos travaux.

Les deux premières parties, **Préliminaires** et **Outils de décidabilité et d'indécidabilité**, regroupent les notions théoriques nécessaires pour aborder ce document. Elles fixent les définitions et les notations qui seront utilisées tout au long de cette thèse.

La première partie introduit de manière concise les termes, les systèmes de réécriture, la logique des prédicats du premier ordre utilisée pour la définition des différentes théories étudiées dans cette thèse, ainsi que quelques notions de topologie qui nous servent pour l'étude des ensembles de solutions des systèmes de contraintes ensemblistes.

La seconde se décompose en trois chapitres et comme son nom l'indique regroupe les différents outils que nous utilisons pour la démonstration de nos résultats de décidabilité et d'indécidabilité. Le premier chapitre rappelle les propriétés de clôture et les résultats de décision pour les automates d'arbres finis standards (AAS) et la classe des langages réguliers (classe des langages reconnus par les AAS). Le second est consacré aux automates à contraintes, automates introduits pour la résolution de problèmes non linéaires. Il rappelle

les différentes classes d'automates à contraintes existantes ainsi que leurs propriétés et certaines de leurs applications. Nous établissons également dans ce chapitre un nouveau résultat puisque nous montrons que l'image par un morphisme d'arbres semi-linéaire (un terme est semi-linéaire si toutes ses non linéarités concernent des variables à la profondeur 1) d'un régulier est un langage reconnu par automates à tests d'égalité entre frères (automates à tests entre frères dans lesquels seuls des tests d'égalité sont réalisés).

Dans le dernier chapitre, nous adaptons aux arbres la méthode de la grille afin d'obtenir un outil de preuves d'indécidabilité. La méthode de la grille est un outil classique dans les preuves d'indécidabilité puisqu'elle fournit un moyen pratique pour coder les calculs des machines de Turing. Dans le codage d'un calcul, chaque ligne de la grille contient une configuration de la machine de Turing considérée et deux lignes successives de la grille correspondent à un mouvement de celle-ci. En fait, une cellule (m, n) de la grille contient la $n^{\text{ième}}$ valeur de la $m^{\text{ième}}$ configuration du calcul considéré. L'avantage de cette structure réside en la possibilité de vérifier uniquement par des tests locaux que les lignes successives d'une grille correspondent aux configurations successives d'un calcul réussi de la machine de Turing. Ce codage permet de déduire de l'indécidabilité de l'arrêt pour les machines de Turing d'autres résultats importants d'indécidabilité. Par exemple, ce codage est utilisé dans la preuve de l'indécidabilité de plusieurs variantes du problème de pavage d'un plan par des dominos. Par la suite, ces résultats ont permis de prouver l'indécidabilité de plusieurs sous-classes du calcul des prédicats [50, 95, 39, 58] et l'indécidabilité de la solvabilité des équations diophantiennes exponentielles [93].

L'adaptation de la méthode de la grille que nous effectuons consiste à associer à toute grille finie un arbre fini. Nous définissons ainsi un ensemble d'arbres particuliers codant les calculs réussis d'une machine de Turing débutant sur l'entrée vide et nous obtenons donc un codage du problème indécidable de l'arrêt sur l'entrée vide des machines de Turing dans les arbres. Nous appelons la méthode ainsi définie la méthode des Arbres-Grilles.

La troisième partie, **Automates d'arbres à contraintes**, est composée de trois chapitres. Nous nous intéressons, dans les deux premiers, à la frontière entre la décidabilité et l'indécidabilité du problème du vide pour les automates d'arbres à contraintes. Dans le premier chapitre, nous associons, par notre méthode de la grille, à une machine de Turing $\mathcal{M}$ un automate à tests d'égalité entre cousins (automates à contraintes dont les tests sont des tests d'égalité entre sous-termes à la profondeur 2 : automates *ATEC*) reconnaissant l'ensemble des calculs réussis de $\mathcal{M}$ débutant sur l'entrée vide. Nous en déduisons que bien que le problème du vide soit décidable pour les automates imposant des tests d'égalité et de différence entre termes frères (automates *ATF*), ce problème devient indécidable dès que des tests d'égalités entre termes cousins sont autorisés (résultat précedemment établi par M. Tommasi dans son rapport de D.E.A. [86]).

Dans le second chapitre, nous déduisons des éléments rendant le problème du vide indécidable pour les *ATF*, une nouvelle sous-classe d'automates à contraintes pour laquelle le vide est décidable. Nous obtenons ainsi une réduction de la frontière entre la décidabilité et l'indécidabilité du problème du vide pour les automates d'arbres à contraintes. Ces automates (automates *ATEFCS*) autorisent des tests d'égalité restreints soit entre termes frères (tests entre sous-termes à la profondeur 1), soit entre termes cousins.

Lorsqu'on considère un langage reconnu par un automate à contraintes il est naturel de se demander si ce langage est reconnaissable par un automate d'arbres standards, c'est-à-dire

les contraintes sont-elles réellement nécessaires pour définir le langage? Pouvoir se passer des contraintes est intéressant car cela permet en particulier d'utiliser les algorithmes classiques des langages reconnaissables par les automates d'arbres standard. Pour les automates à contrainte généraux, ce problème contient strictement la décidabilité de la reconnaissabilité de l'ensemble des formes normales d'un système de réécriture, problème résolu mais dont les preuves sont très techniques [92, 54].

Nous montrons dans le troisième chapitre de cette partie que ce problème est décidable pour la classe des langages reconnus par les automates à tests entre frères. En fait, nous définissons un type de minimisation similaire à la minimisation classique (théorème de Myhill-Nerode pour les langages reconnaissables par les automates d'arbres standards [20]) mais prenant en compte les contraintes.

La quatrième partie, **Réécriture**, est découpée en deux chapitres. Le premier est consacré à la théorie de la réécriture en un pas. Après avoir défini cette théorie, nous rappelons les liens existants entre celle-ci et l'unification contextuelle. En particulier, nous montrons qu'il est facile de définir une légère généralisation des contraintes de réécriture ($x \to y$) permettant de positionner les applications de réécriture les unes par rapport aux autres et d'établir une relation étroite entre cette généralisation et l'unification contextuelle stratifiée (J. Niehren et R. Treinen [67]).

Ensuite nous nous intéressons à la frontière entre la décidabilité et l'indécidabilité de la théorie de la réécriture en un pas. Nous associons, par notre méthode de la grille, à une machine de Turing $\mathcal{M}$ une formule du fragment $\exists^*\forall^*$ de la théorie de la réécriture en un pas pour les systèmes de réécriture linéaires, minces et convergents satisfiable si et seulement s'il existe un calcul réussi de $\mathcal{M}$ débutant sur l'entrée vide. Nous en déduisons l'indécidabilité de ce fragment pour les systèmes de réécriture linéaires, minces, nœthériens et confluents.

Finalement nous étudions la décidabilité du fragment existentiel de la théorie du premier ordre de la réécriture en un pas. Nous donnons une procédure de décision basée sur des règles d'inférence pour résoudre les formules existentielles construites sur les formules atomiques $x \to_{\mathcal{R}} y$, $x = y$ (contrainte d'égalité) et $x \in \mathcal{L}$ (contrainte d'appartenance) où $\mathcal{L}$ est un langage reconnaissable par automates à tests entre frères. Cette procédure est définie pour les systèmes de réécriture dont toutes les occurrences de variables dans les règles sont à la profondeur 1 (systèmes de réécriture ultra-minces). Nous en déduisons la décidabilité du fragment existentiel de la théorie construite sur les formules atomiques précédentes pour les systèmes de réécriture ultra-minces. L'expressivité du fragment considéré est relativement faible par rapport aux propriétés classiques utiles en théorie de la réécriture. Par exemple, les formules de ce fragment ne peuvent exprimer la confluence que pour un nombre borné de pas (à cause de la restriction "en un pas") et pour les systèmes ultra-minces. Néanmoins, les résultats d'indécidabilité de la théorie de la réécriture en un pas et les connexions avec l'unification contextuelle montrent que la résolution des formules de réécriture en un pas est difficile et intéressante en elle-même.

Dans le second chapitre, nous nous intéressons tout d'abord au langage $\mathcal{R}^*(\mathcal{L})$ où $\mathcal{R}$ est un système de réécriture et $\mathcal{L}$ un langage régulier. Ce langage, image de $\mathcal{L}$ par application d'un nombre quelconque de pas de réécriture, n'est pas en général régulier même si $\mathcal{R}(\mathcal{L})$, l'image de $\mathcal{L}$ par application d'un pas de réécriture, l'est. Pourtant certains chercheurs ont montré que $\mathcal{R}^*(\mathcal{L})$ est régulier pour certaines classes de systèmes de réécriture: M. Dauchet et S. Tison [24] pour les systèmes clos, K. Salomaa [76] pour les systèmes monadiques linéaires

à droite, J.-L. Coquidé *et al.* [21] pour les systèmes semi-monadiques linéaires.

Les méthodes utilisées dans ces études nous semblant similaires, nous avons cherché une méthode d'étude générique. Nous présentons tout d'abord les résultats de cette recherche puis nous montrons que notre approche nous permet de définir une méthode générale d'étude des questions de décision « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » où $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des langages réguliers et $\mathcal{R}el(\mathcal{L}_1)$ est l'ensemble image des termes de $\mathcal{L}_1$ par une relation $\mathcal{R}el$. Ces questions sont décidables lorsque la relation est un morphisme d'arbres inverse ou un morphisme d'arbres linéaire car alors $\mathcal{R}el(\mathcal{L}_1)$ est un langage régulier.

Finalement nous étudions ces questions pour différentes relation $\mathcal{R}el$ dépendantes d'un système de réécriture. Nous considérons tout d'abord les cas où $\mathcal{R}el$ est l'application d'un pas de réécriture et l'application d'une réécriture parallèle, cette dernière opération consistant à appliquer en un pas de dérivation des réécritures s'appliquant sur des sous-termes à des positions incomparables. Ensuite nous considérons le cas des réécritures en une passe.

Pour qu'un terme t se réécrive en un terme u en une passe, il faut que toutes les occurrences des membres gauches de règles utilisées dans la dérivation de t en u apparaissent dans t sans chevauchement. De plus, à cause des non linéarités, il est indispensable de préciser si les sous-termes sont réécrits avant ou après avoir testé les égalités imposées par les règles. Il existe deux stratégies usuelles pour résoudre ce problème [28, 29, 30]. La première, appelée OI (Outermost–Innermost), consiste à réécrire le terme de la racine vers les feuilles et la seconde, appelée IO (Innermost–Outermost), consiste à réécrire le terme des feuilles vers la racine. Z. Fülöp et al dans [33] introduisent deux nouvelles stratégies: la réécriture en une passe par la racine et la réécriture en une passe par les feuilles. Ces stratégies sont respectivement des cas particuliers de réécriture en une passe OI et en une passe IO dans lesquelles, une réécriture concerne toujours les positions immédiatement adjacentes aux parties du terme réécrit précédemment.

Z. Fülöp et al dans [33] montrent que la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable pour les systèmes linéaires à gauche lorsque $\mathcal{R}el$ est l'une des relations suivantes : calcul des formes normales ou des formes sententielles selon la stratégie de réécriture en une passe par la racine ou par les feuilles. Les formes sententielles d'un terme t sont les termes u pour lesquelles il existe une dérivation de t en u, et les formes normales de t sont les formes sententielles de t ne pouvant plus être réécrites par la stratégie considérée. Nous montrons que, sous certaines conditions, les questions « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » sont décidables lorsque $\mathcal{R}el$ est le calcul des formes normales ou des formes sententielles selon une passe IO, une passe OI, la stratégie de réécriture en une passe par la racine ou par les feuilles.

La cinquième partie, **Contraintes ensemblistes**, est composée de quatre chapitres. Le premier contient les définitions des contraintes ensemblistes ainsi qu'un état de l'art rapide de leurs études.

Dans le second, nous associons, par notre méthode de la grille, à une machine de Turing $\mathcal{M}$ une formule du fragment $\exists^*\forall^*$ de la théorie des contraintes ensemblistes satisfiable si et seulement s'il existe un calcul réussi de $\mathcal{M}$ débutant sur l'entrée vide. Nous en déduisons l'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules contenant pour seul opérateur ensembliste l'union.

Dans le troisième chapitre, nous nous intéressons à l'enrichissement du vocabulaire des contraintes ensemblistes afin, par exemple, de pouvoir obtenir des analyses ensemblistes plus précises et pertinentes. Nous étudions en particulier les contraintes ensemblistes munis d'opé-

rateurs permettant de tester des égalités entre termes frères (opérateurs de diagonalisation). Nous définissons une nouvelle classe d'automates (automates d'ensembles d'arbres généralisés avec tests entre frères) pour laquelle le problème du vide est décidable et nous montrons que l'on peut associer à tout système de contraintes positives avec opérateurs de diagonalisation, un automate d'ensembles d'arbres généralisés avec tests entre frères reconnaissant l'ensemble des solutions du système. Nous en déduisons que le problème de satisfiabilité pour les systèmes de contraintes ensemblistes positives avec opérateurs de diagonalisation est décidable.

Finalement certains résultats sur les contraintes ensemblistes peuvent s'obtenir par une démarche topologique. Par exemple, dans [53], D. Kozen étudie la structure topologique de l'ensemble des solutions des systèmes de contraintes ensemblistes. Il définit une famille d'espaces topologiques, appelés espaces rationnels, et donne une caractérisation complète de l'ensemble de solutions de tout système de contraintes ensemblistes en terme d'espaces rationnels. M. Tommasi, dans [87], montre en utilisant les automates d'ensembles d'arbres que l'ensemble des solutions d'un système de contraintes ensemblistes positives est un compact (donc un fermé). En fait, il associe à chaque système SC de contraintes ensemblistes positives un automate d'ensembles d'arbres particulier, dit sans condition d'acceptation, reconnaissant l'ensemble des solutions de SC, le langage reconnu par un tel automate étant un compact.

Dans le dernier chapitre, nous étudions les propriétés topologiques des ensembles de solution des systèmes de contraintes ensemblistes. Nous montrons directement que cet ensemble est un fermé pour un système de contraintes positives. Et nous montrons que cet ensemble est l'intersection d'un fermé et d'un ouvert pour un système de contraintes positives et négatives. Ensuite nous définissons un système de contraintes positives avec symboles de projection dont nous montrons par le résultat précédent que l'ensemble de solutions n'est solution d'aucun système de contraintes positives et negatives. Nous en déduisons que le pouvoir d'expression en terme d'ensemble de solutions des contraintes ensemblistes positives avec symboles de projection est strictement supérieur à celui des contraintes ensemblistes positives et négatives. Une classe de contraintes ensemblistes $(C1)$ est dite plus expressive qu'une classe $(C2)$ si pour tout système $SC2$ de contraintes de $(C2)$, il existe un système $SC1$ de contraintes de $(C1)$ ayant même ensemble de solutions que $SC2$.

Première partie

Préliminaires

Chapitre 1

Ensembles

Dans ce chapitre, nous donnons quelques notations sur les ensembles et nous définissons les notions de relations et d'ordres que nous utilisons tout au long de ce document.

1.1 Mots

Un *alphabet* A est un ensemble de symboles appelés *lettres*. Un *mot fini* u sur A est une suite finie de lettres de A, c'est-à-dire $u = u_1 \dots u_n$ avec $u_1, \dots, u_n \in A$. Le *mot vide*, noté ϵ, est le mot constitué d'aucune lettre. A^* désigne l'ensemble de tous les mots finis sur A (le mot vide compris) et A^+ l'ensemble A^* privé du mot vide :

$$A^+ = A^* \setminus \{\epsilon\}.$$

Soit u un mot sur A. La *longueur* de u, notée $|u|$, est égale au nombre d'éléments de A le constituant, c'est-à-dire :

- $|u| = 0$ si u est le mot vide;
- $|u| = n$ si $u = u_1 \dots u_n$ avec $u_1, \dots, u_n$ appartenant à A et n entier strictement positif.

Soient $u = u_1 \dots u_n$ et $v = v_1 \dots v_n$ deux mots sur A. La *concaténation* de u et v, notée uv, est le mot $u_1 \dots u_n v_1 \dots v_n$.

1.2 Notations

Nous désignons par :

- $\mathbb{R}$ l'ensemble des réels,
- $\mathbb{Z}$ l'ensemble des entiers relatifs,
- $\mathbb{N}$ l'ensemble des entiers relatifs positifs,
- $\mathbb{N}_+$ l'ensemble des entiers relatifs positifs non nuls,
- $[n]$ l'ensemble $\{1, \dots, n\}$ pour $n \in \mathbb{N}$ (ainsi $[0]$ est une autre notation pour l'ensemble vide $\emptyset$),

- 2^E l'ensemble des sous-ensembles d'un ensemble E :

$$2^E = \{F \mid F \subseteq E\}.$$

Soient A et B deux ensembles. On note $A\Delta B$, la *différence symétrique* de A et B c'est-à-dire $A\Delta B = (A \setminus B) \cup (B \setminus A)$. Et si $B \subseteq A$, on note $\complement_A B$ le complément de B dans A c'est-à-dire $\complement_A B = A \setminus B$. Finalement, on note *card*(A) le *cardinal* de A c'est-à-dire le nombre d'éléments de A. Dans le cas où le cardinal est infini, on le note ∞.

1.3 Relations

Soit E un ensemble et n un entier strictement positif. Une *relation* sur E *d'arité* n est un sous-ensemble de E^n. En particulier une relation R sur E d'arité 2 est dite *binaire* et on note xRy si $(x, y) \in R$. Une relation binaire R sur E est dite :

Réflexive	si et seulement si	$\forall x \in E,\ xRx$
Irréflexive	si et seulement si	$\forall x \in E,\ \neg(xRx)$
Symétrique	si et seulement si	$\forall x, y \in E,\ (xRy) \Rightarrow (yRx)$
Antisymétrique	si et seulement si	$\forall x, y \in E,\ (xRy \wedge yRx) \Rightarrow (x = y)$
Transitive	si et seulement si	$\forall x, y, z \in E,\ (xRy \wedge yRz) \Rightarrow (xRz)$

Soient R et S deux relations binaires sur E. La *composition* de R et S est la relation binaire $R \circ S$ sur E définie par $R \circ S = \{(x, z) \in E^2 \mid \exists y \in E, (x, y) \in R \wedge (y, z) \in S\}$. Pour une relation binaire $\rightarrow$ sur E, nous définissons les compositions suivantes de $\rightarrow$ avec elle-même :

$\rightarrow^0$	$:=$	$\{(x, x) \mid x \in E\}$	Identité
$\rightarrow^1$	$:=$	$\rightarrow$	
$\rightarrow^{m+1}$	$:=$	$\rightarrow^m \circ \rightarrow$	$(m+1)^{\text{ième}}$ composition (m entier de $\mathbb{N}_+$)
$\xrightarrow{+}$	$:=$	$\bigcup_{m \in \mathbb{N}_+} \rightarrow^m$	Clôture transitive
$\xrightarrow{*}$	$:=$	$\xrightarrow{+} \cup \rightarrow^0$	Clôture réflexive et transitive
$\xrightarrow{=}$	$:=$	$\rightarrow \cup \rightarrow^0$	Clôture réflexive

Remarquons que la clôture réflexive $\xrightarrow{=}$ (respectivement la clôture transitive $\xrightarrow{+}$, la clôture réflexive et transitive $\xrightarrow{*}$) de $\rightarrow$ est la plus petite relation binaire réflexive (respectivement transitive, réflexive et transitive) sur E contenant $\rightarrow$.

1.4 Ordres

Soit E un ensemble. Une *relation d'équivalence* $\sim$ sur E est une relation binaire sur E réflexive, symétrique et transitive. $\sim$ induit pour tout x de E une *classe d'équivalence* $[x]_\sim = \{y \in E \mid x \sim y\}$.

Un *ordre partiel* est une relation binaire réflexive, antisymétrique et transitive, notée habituellement $\geq$. Un couple $(E, \geq)$ constitué d'un ensemble E et d'un ordre partiel sur E est

appelé un *ensemble partiellement ordonné*. Un *ordre strict* est une relation binaire irréflexive et transitive, notée habituellement $>$. Les ordres partiels et les ordres stricts sont interdéfinissables puisque :

- Un ordre strict $>$ se déduit de tout ordre partiel $\geq$:

$$(x > y \Leftrightarrow x \geq y \wedge x \neq y).$$

- Un ordre partiel $\geq$ se déduit de tout ordre strict $>$:

$$(x \geq y \Leftrightarrow x > y \vee x = y).$$

Pour un ordre partiel $\geq$, $x \leq y$ signifie $y \geq x$ et $x \not\geq y$ signifie $\neg(x \geq y)$. De même pour un ordre strict $>$, $x < y$ signifie $y > x$ et $x \not> y$ signifie $\neg(x > y)$.

Soient $(E, \geq)$ un ensemble partiellement ordonné et S un sous-ensemble de E. Un *majorant* (respectivement *inférieure*) de S (dans E) est un élément M (respectivement m) de E tel que pour tout élément s de S, on a $M \geq s$ (respectivement $s \geq m$). Un majorant M (respectivement minorant m) de S est *strict* si pour tout élément s de S, on a $M > s$ (respectivement $s > m$).

Chapitre 2

Termes et Arbres

2.1 Signature, termes et contextes

Une *signature* est un couple $(\Sigma, Arité)$, où Σ est un ensemble fini de *symboles de fonctions* et *Arité* est une application de Σ dans $\mathbb{N}$. Pour tout symbole f de Σ, $Arité(f)$ est appelé l'*arité* de f. Les symboles d'arité 0 sont nommés *constantes* et ceux d'arité $1, 2, 3, \ldots, n$ sont dits respectivement unaires, binaires, ternaires, ..., n-aires. Nous notons, Σ_i (respectivement $\Sigma_{>i}$, $\Sigma_{\geq i}$), l'ensemble des symboles de fonction de Σ d'arité i (respectivement d'arité strictement supérieure à i, d'arité supérieure ou égale à i). Dans les exemples, nous utilisons la notation f/n pour désigner un symbole de fonction f d'arité n.

Exemple 1. *La notation $\Sigma_G = \{e/0, i/1, f/2\}$ définie une signature Σ_G composée d'une constante e, d'un symbole de fonction unaire i et d'un symbole de fonction binaire f.*

Remarque 2. *Dans la suite de ce document :*

- *Nous supposons que toute signature considérée contient au moins une constante.*
- *Nous ne faisons plus la distinction entre une signature $(\Sigma, Arité)$ et l'ensemble Σ, en supposant que Arité désigne toujours l'application arité correspondant à Σ.*

Soient Σ une signature et $\mathcal{X}$ un ensemble dénombrable de symboles appelés *variables du premier ordre*. L'ensemble des *termes* construits sur la signature Σ et les variables de $\mathcal{X}$, noté $T(\Sigma, \mathcal{X})$, est défini comme le plus petit ensemble vérifiant :

- Les variables sont des termes c'est-à-dire $\mathcal{X} \subseteq T(\Sigma, \mathcal{X})$,
- Les constantes de Σ sont des termes c'est-à-dire $\Sigma_0 \subseteq T(\Sigma, \mathcal{X})$,
- Si f est un symbole d'arité n, n entier strictement positif, et $t_1, \ldots, t_n$ sont des termes ($t_1, \ldots, t_n$ appartiennent à $T(\Sigma, \mathcal{X})$), alors $f(t_1, \ldots, t_n)$ appartient à $T(\Sigma, \mathcal{X})$.

L'ensemble des variables apparaissant dans un terme t est désigné par $Var(t)$. Un terme ne comportant pas de variables est dit *clos* et $T(\Sigma)$ désigne l'ensemble des termes clos sur Σ. Un terme t est dit *linéaire* si et seulement si chaque variable x de $Var(t)$ apparaît une seule fois dans t.

Exemple 3. *Soient la signature Σ_G de l'exemple 1 et l'ensemble de variables $\mathcal{X} = \{x, y\}$. Alors $f(e, f(x, f(y, e)))$ est un terme linéaire de $T(\Sigma_G, \mathcal{X})$ et $f(e, i(e))$ un terme clos.*

Soit n un entier. Alors $\mathcal{X}_n$ désigne un ensemble de n variables, $\mathcal{X}_n = \{x_1, \ldots, x_n\}$. Un terme linéaire C de $T(\Sigma, \mathcal{X}_n)$ est appelé un *contexte* et l'expression $C[t_1, \ldots, t_n]$ avec $t_1, \ldots, t_n$ termes de $T(\Sigma)$ désigne le terme obtenu à partir de C en remplaçant pour tout i de $[n]$, x_i par t_i. On note $\mathcal{C}^n(\Sigma)$ l'ensemble des contextes sur Σ et $\mathcal{X}_n$, et $\mathcal{C}(\Sigma)$ l'ensemble des contextes contenant une seule variable.

Exemple 4. *Soient la signature Σ_G de l'exemple 1 et l'ensemble de variables $\mathcal{X} = \{x_1, x_2\}$. Le terme $C = f(e, f(x_2, i(x_1)))$ est un contexte de $\mathcal{C}^2(\Sigma)$ et le terme obtenu à partir de C en remplaçant x_1 par e et x_2 par $i(e)$ est $C[e, i(e)] = f(e, f(i(e), i(e)))$.*

Nous venons de voir dans cette section, une définition inductive des termes construits sur une signature Σ et un ensemble de variables $\mathcal{X}$. Dans la section suivante, nous voyons que les termes de $T(\Sigma, \mathcal{X})$ se définissent également en utilisant les notions de positions et d'arbres.

2.2 Positions et arbres

On appelle *position* tout élément de $(\mathbb{N}_+)^*$. Soient p et p' deux positions. On dit que p' et un *préfixe* de p s'il existe une position q telle que $p = p'q$. La relation "est préfixe de" est de façon évidente un ordre partiel sur l'ensemble $(\mathbb{N}_+)^*$ des positions et est appelé *ordre préfixe*. On note $\trianglelefteq$ cet ordre et $\triangleleft$ l'ordre strict associé à $\trianglelefteq$, c'est à dire $p \triangleleft p'$ si et seulement si $p \trianglelefteq p'$ et $p \neq p'$. Deux positions p et p' sont dites *parallèles* (notation $p \parallel p'$) si elles sont incomparables par l'ordre préfixe $\trianglelefteq$ (c'est-à-dire $p \not\trianglelefteq p'$ et $p' \not\trianglelefteq p$).

Exemple 5. *Considérons les positions $p_1 = 21$, $p_2 = 212$ et $p_3 = 22$. Alors $p_1 \triangleleft p_2$ et $p_1 \| p_3$.*

Un *arbre* fini t sur $\Sigma \cup \mathcal{X}$ est une fonction partielle de $(\mathbb{N}_+)^*$ dans $\Sigma \cup \mathcal{X}$ dont le domaine $\mathcal{P}os(t)$ (appelé *ensemble des positions* de t) satisfait les propriétés suivantes :

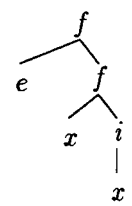
1. $\mathcal{P}os(t)$ est non vide et fini;
2. $\mathcal{P}os(t)$ est *clos par préfixe*, c'est-à-dire pour toute position p de $\mathcal{P}os(t)$, si $p = p'q$ avec p' et q positions alors q appartient à $\mathcal{P}os(t)$;
3. $\mathcal{P}os(t)$ respecte l'arité des symboles de fonctions de Σ, c'est-à-dire que pour toute position p de $\mathcal{P}os(t)$, si $t(p)$ est un symbole de Σ_n avec n entier, alors :
   - Pour tout entier j de $[n]$, pj apparient à $\mathcal{P}os(t)$, et
   - Pour tout entier j strictement supérieur à n, pj n'appartient pas à $\mathcal{P}os(t)$;

4. Pour toute position p de $\mathcal{P}os(t)$, si $t(p)$ est une variable de $\mathcal{X}$, alors pour tout entier j, pj n'appartient pas à $\mathcal{P}os(t)$.

Remarque 6. *De façon évidente tout terme de $T(\Sigma, \mathcal{X})$ est un arbre fini sur $\Sigma \cup \mathcal{X}$ et inversement. A partir de maintenant, nous ne faisons donc plus la distinction entre arbre et terme.*

Tout arbre t peut être représenté par un graphe acyclique orienté dont les sommets sont étiquetés par les symboles de fonctions $t(p)$ pour toute position p de $\mathcal{P}os(t)$ et les flèches relient le sommet $t(p)$ de Σ_n pour toute position p de $\mathcal{P}os(t)$ aux sommets $t(pj)$ pour tout j de $[n]$. Dans ce document, nous dessinons les arbres de sorte que leurs flèches soient orientées du haut vers le bas, ainsi nous remplaçons les flèches par des arcs.

Exemple 7. *Soit l'arbre $t = f(e, f(x, i(x)))$ défini sur la signature Σ_G de l'exemple 1. L'ensemble des positions de t est $\mathcal{P}os(t) = \{\epsilon, 1, 2, 21, 22, 221\}$ et $t(\epsilon) = f$, $t(1) = e$, $t(2) = f$, $t(21) = x$, $t(22) = i$, $t(221) = x$. La représentation graphique de t est:*



Soit t un arbre de $T(\Sigma, \mathcal{X})$. Pour toute position p de $\mathcal{P}os(t)$, $t(p)$ est appelé *étiquette* de t à la position p. Le symbole $t(\epsilon)$ est appelé *symbole de tête* de l'arbre et est notée $tête(t)$. La position ϵ est appelée *racine* de l'arbre et toute position p de $\mathcal{P}os(t)$ est appelée:

- *Position non feuille* ou *nœud interne* de t si $t(p)$ est un symbole de Σ_n avec n entier strictement positif,
- *Position feuille* de t si $t(p)$ est une constante ou une variable,
- *Position de variable* de t si $t(p)$ est une variable.

On note $\mathcal{IP}os(t)$ l'ensemble des positions non feuilles de t, $\mathcal{LP}os(t)$ l'ensemble de ses positions feuilles et $\mathcal{VP}os(t)$ l'ensemble de ses positions de variable. Nous avons alors $\mathcal{P}os(t) = \mathcal{IP}os(t) \cup \mathcal{LP}os(t)$ et $\mathcal{IP}os(t) \cap \mathcal{LP}os(t) = \emptyset$.

Un terme t est dit *semi-linéaire* si et seulement si pour toutes positions distinctes p_1 et p_2 de $\mathcal{VP}os(t)$ telles que $t(p_1) = t(p_2)$, on a $|p_1| = |p_2| = 1$. Donc toutes les non linéarités d'un terme semi-linéaire sont à profondeur 1.

Exemple 8. *Soit la signature $\Sigma = \{h/4, f/2, g/1, a/0\}$. Alors les termes $h(x, f(y, z), x, a)$ et $h(a, g(y), x, x)$ sont semi-linéaires mais pas le terme $h(a, g(y), a, f(x, x))$.*

Un *chemin* π dans t est un sous-ensemble de $\mathcal{P}os(t)$ totalement ordonné (c'est-à-dire pour toutes positions p et q de π, on a $p \trianglelefteq q$ ou $q \trianglelefteq p$) et clos par préfixe. Une *branche* dans t est un chemin de t contenant une position feuille.

Exemple 9. *Pour l'arbre* $t = f(e, f(x, i(x)))$, 2 *est un nœud interne,* 1 *une position feuille,* 221 *une position de variable et* $\pi = \{\epsilon, 2, 21\}$ *une branche.*

Le *sous-arbre* ou *sous-terme* de t à la position p de $\mathcal{P}os(t)$ est l'arbre noté $t|_p$ défini par:

- $\mathcal{P}os(t|_p) = \{p' \mid pp' \in \mathcal{P}os(t)\}$ et,
- $\forall p' \in \mathcal{P}os(t|_p),\ t|_p(p') = t(pp')$.

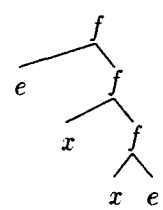
Soient t et t' deux arbres et n entier. t' est appelé sous-arbre de t à la *profondeur* n s'il existe une position p de $\mathcal{P}os(t)$ telle que $t|_p = t'$ et $|p| = n$. Les sous-arbres de t à la profondeur 1 sont appelés les *fils* du symbole $tête(t)$ et $tête(t)$ est appelé leur *père*. Ces sous-arbres sont également appelés les sous-termes *directs* de t. Les sous-arbres de t à la profondeur 2 sont appelés les *cousins* du symbole $tête(t)$. L'ensemble des sous-arbres de t, notée $\mathcal{S}arb(t)$, est $\{t|_p \mid p \in \mathcal{P}os(t)\}$.

Le remplacement d'un arbre t de $T(\Sigma, \mathcal{X})$ par un arbre u de $T(\Sigma, \mathcal{X})$ à une position p de $\mathcal{P}os(t)$ est l'arbre, noté $t[u]_p$, défini par:

- $\mathcal{P}os(t[u]_p) = \{pj \mid j \in \mathcal{P}os(u)\} \cup \{p' \mid p' \in \mathcal{P}os(t), p \not\trianglelefteq p'\}$,
- $\forall p' \in \mathcal{P}os(t[u]_p),\ t[u]_p(p') = t(p')$ si $p \not\trianglelefteq p'$.
  $= u(j)$ si $p' = pj$.

Exemple 10. *Pour les termes* $t = f(e, f(x, i(x)))$ *et* $u = f(x, e)$, *nous avons* $t|_{22} = i(x)$ *et:*

$t[u]_{22} =$



2.3 Applications associées

Nous définissons dans cette section, les applications usuelles associées à la notion d'arbre.

2.3.1 Hauteur

La *hauteur* d'un arbre t de $T(\Sigma, \mathcal{X})$, notée $Haut(t)$, est définie par :

- $Haut(t) = 0$ si t est une constante ou une variable,
- $Haut(t) = 1 + max(\{Haut(t_i) \mid i \in [n]\}$ si $t = f(t_1, \ldots, t_n)$ avec n entier strictement positif.

Pour tout entier k, nous désignons par $T(\Sigma)_{\leq k}$ (respectivement $T(\Sigma)_{<k}$) l'ensemble des termes de $T(\Sigma)$ de hauteur inférieure ou égale (respectivement strictement inférieure) à k c'est-à-dire :

$$\begin{aligned} T(\Sigma)_{\leq k} &= \{t \in T(\Sigma) \mid Haut(t) \leq k\}, \\ T(\Sigma)_{<k} &= \{t \in T(\Sigma) \mid Haut(t) < k\}. \end{aligned}$$

2.3.2 Substitution

Les constantes et les variables sont des symboles d'arité 0 pourtant elles n'ont pas même utilité puisque contrairement aux constantes les variables peuvent être remplaçées par des termes par l'intermédiaire de substitutions.

Une *substitution* sur $T(\Sigma, \mathcal{X})$ est une application de $\mathcal{X}$ dans $T(\Sigma, \mathcal{X})$ qui diffère de l'identité uniquement pour un nombre fini de variables. Le *domaine* $\mathcal{D}om(\sigma)$ d'une substitution σ est l'ensemble des variables de $\mathcal{X}$ telles que $\sigma(x) \neq x$, c'est-à-dire $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$. Une substitution σ de domaine $\{x_1, \ldots, x_n\}$ telle que pour tout i de $[n]$, $\sigma(x_i) = t_i$ terme de $T(\Sigma, \mathcal{X})$, est désignée par $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$. On dit que σ est une *substitution close sur* $T(\Sigma)$) si pour toute variable x de $\mathcal{D}om(\sigma)$, $\sigma(x)$ est un terme de $T(\Sigma)$. L'ensemble des substitutions sur $T(\Sigma, \mathcal{X})$ (respectivement $T(\Sigma)$) est notée $\mathcal{S}ub(T(\Sigma, \mathcal{X}))$ (respectivement $\mathcal{S}ub(T(\Sigma))$).

Toute substitution σ sur $T(\Sigma, \mathcal{X})$ (respectivement $T(\Sigma)$) se prolonge de façon unique en une application $\hat{\sigma}$ de $T(\Sigma, \mathcal{X})$ dans $T(\Sigma, \mathcal{X})$ (respectivement $T(\Sigma)$ dans $T(\Sigma)$) définie par :

- Pour toute variable x de $\mathcal{X}$, $\hat{\sigma}(x) = \sigma(x)$, et
- Pour tout terme t de $T(\Sigma, \mathcal{X}) \setminus \mathcal{X}$ tel que $t = f(t_1, \ldots, t_n)$, $\hat{\sigma}(t) = f(\hat{\sigma}(t_1), \ldots, \hat{\sigma}(t_n))$.

Exemple 11. *Soient la signature Σ_G de l'exemple 1 et un ensemble de variables $\mathcal{X}$. Nous considérons les arbres $s = f(e, x)$ et $t = f(y, f(x, y))$ de $T(\Sigma_G, \mathcal{X})$, et la substitution $\sigma = \{x \leftarrow i(y), y \leftarrow e\}$. Alors $\hat{\sigma}(s) = f(e, i(y))$ et $\hat{\sigma}(t) = f(e, f(i(y), e))$.*

Une substitution σ est appelée *renommage* (de variables) si c'est une bijection sur $\mathcal{X}$ (donc une bijection sur $T(\Sigma, \mathcal{X})$).

La *composition* $\sigma\tau$ de deux substitutions σ et τ est définie par $\sigma\tau(x) = \hat{\sigma}(\tau(x))$. De façon évidente, $\sigma\tau$ est une application de $\mathcal{X}$ dans $T(\Sigma, \mathcal{X})$ et puisque $\sigma\tau(x) = x$ pour toute variable x n'appartenant pas à $\mathcal{D}om(\sigma) \cup \mathcal{D}om(\tau)$, $\sigma\tau$ est encore une substitution. De plus, il est facile de vérifier que la composition est une opération associative. Par définition de la composition de substitutions, le prolongement de la composition $\sigma\tau$ est juste la composition

des prolongements de σ et τ, c'est-à-dire $\hat{\sigma}\tau = \hat{\sigma}\hat{\tau}$ (où $\hat{\sigma}\hat{\tau}$ désigne la composition usuelle de fonctions).

A partir de maintenant, nous confondons toute substitution σ avec son prolongement $\hat{\sigma}$ et nous utilisons la notation postfixée $t\sigma$ pour désigner le résultat de l'application de σ sur un arbre t.

Un arbre t de $T(\Sigma, \mathcal{X})$ est appelé une *instance* d'un arbre s de $T(\Sigma, \mathcal{X})$ s'il existe une substitution σ telle que $s\sigma = t$. L'instance t est dite *close* si t est un arbre clos.

Exemple 12. *Soient les arbres $s = f(e, x)$ et $t = f(e, f(e, e))$ de $T(\Sigma_G, \mathcal{X})$. Alors t est une instance close de s puisque pour la substitution $\sigma = \{x \leftarrow f(e, e)\}$, on a $s\sigma = t$.*

Une substitution σ est dite *plus générale* qu'une substitution τ s'il existe une substitution δ telle que $\tau = \delta\sigma$. Nous écrivons alors $\sigma \lesssim \tau$.

Exemple 13. *Soient les substitutions $\sigma = \{x \leftarrow i(y)\}$ et $\tau = \{x \leftarrow i(e), y \leftarrow e\}$ alors $\sigma \lesssim \tau$ puisque $\tau = \delta\sigma$ avec $\delta = \{y \leftarrow e\}$.*

Soient t et t' deux termes de $T(\Sigma, \mathcal{X})$. Alors t et t' sont dits *unifiables* s'il existe une substitution σ telle que $t\sigma = t'\sigma$, et σ est appelée *unificateur* de t et t'. Une substitution σ est l'un des *unificateurs les plus généraux* de t et t' si σ satisfait les conditions suivantes :

- σ est un unificateur de t et t' et,
- $\sigma \lesssim \tau$ pour tout unificateur τ de t et t'.

Remarquons que la relation $\lesssim$ n'est pas antisymétrique puisque pour $\sigma = \{x \leftarrow y\}$ et $\tau = \{y \leftarrow x\}$, nous avons $\sigma \lesssim \tau$, $\tau \lesssim \sigma$ et $\sigma \neq \tau$. Finalement nous pouvons montrer qu'il n'y a pas unicité de l'unificateur le plus général de deux termes.

2.4 Langages d'arbres

Soit Σ une signature. Un *langage d'arbres* sur Σ est un sous-ensemble d'arbres de $T(\Sigma)$ et une *classe de langages* est un ensemble de langages d'arbres.

Soient $\mathcal{X}$ un ensemble de variables, n un entier strictement positif, C un contexte de $\mathcal{C}^n(\Sigma)$ et $L_1, \ldots, L_n$, n langages d'arbres de $T(\Sigma)$. Alors $C[L_1, \ldots, L_n]$ désigne le langage d'arbres suivant :

$$C[L_1, \ldots, L_n] = \{C[t_1, \ldots, t_n] \mid \forall i \in [n], t_i \in L_i\}.$$

Dans la suite de ce document, nous nous intéressons aux propriétés de clôture des classes de langages reconnus par différentes classes d'automates. Une classe de langages $\mathcal{CL}$ est close par une propriété si l'application de cette propriété à tout tuple de langages de $\mathcal{CL}$ est un langage de $\mathcal{CL}$. Nous définissons ci-dessous les propriétés de clôture des langages d'arbres qui nous intéressent en les divisant en deux groupes : les clôtures booléennes et les clôtures par morphismes d'arbres.

2.4.1 Clôtures booléennes

Les opérations que nous considérons dans cette section sont les opérations booléennes ensemblistes suivantes : l'union $\cup$, l'intersection $\cap$ et le complément $\mathbf{C}$. Soit $\mathcal{CL}$ une classe de langages d'arbres sur Σ. La classe $\mathcal{CL}$ est :

Close par union si et seulement si $\mathcal{L}_1 \cup \mathcal{L}_2 \in \mathcal{CL}$ pour tous langages $\mathcal{L}_1$, $\mathcal{L}_2$ de $\mathcal{CL}$
Close par intersection si et seulement si $\mathcal{L}_1 \cap \mathcal{L}_2 \in \mathcal{CL}$ pour tous langages $\mathcal{L}_1$ et $\mathcal{L}_2$ de $\mathcal{CL}$
Close par complément si et seulement si $\mathbf{C}\mathcal{L} \in \mathcal{CL}$ pour tout langage $\mathcal{L}$ de $\mathcal{CL}$.

2.4.2 Clôtures par morphismes d'arbres

Les morphismes d'arbres sont des transformations d'arbres conservant la structure des arbres. Ils sont une généralisation des morphismes de mots au cas des signatures quelconques. Avant de définir la propriété de clôture par morphismes d'arbres, nous définissons les morphismes d'arbres.

Soient Σ et Γ deux signatures, et $\mathcal{X}$ un ensemble dénombrable de variables. Soit φ une application de Σ dans $T(\Gamma, \mathcal{X})$ qui à tout symbole de fonction f de Σ d'arité n, n entier, associe un terme t_f de $T(\Gamma, \mathcal{X}_n)$ (nous rappelons que $\mathcal{X}_n = \{x_1, \ldots, x_n\}$). Le *morphisme d'arbres* Φ de $T(\Sigma)$ dans $T(\Gamma)$ déterminé par φ est défini comme suit :

- $\Phi(a) = t_a$ pour toute constante a de Σ (remarquons que t_a est clos puisque $\mathcal{X}_0 = \emptyset$);
- $\Phi(f(t_1, \ldots, t_n)) = t_f\{x_1 \leftarrow \Phi(t_1), \ldots, x_n \leftarrow \Phi(t_n)\}$ pour tout symbole f de Σ d'arité n supérieure ou égale à un et pour tous termes $t_1, \ldots, t_n$ de $T(\Sigma)$.

Soient Φ un morphisme d'arbres de $T(\Sigma)$ dans $T(\Gamma)$ et $\mathcal{L}$ un langage d'arbres sur Σ. L'*image* de $\mathcal{L}$ par Φ est le langage d'arbres sur Γ défini par $\Phi(\mathcal{L}) = \{\Phi(t) \mid t \in \mathcal{L}\}$. Soit $\mathcal{L}'$ un langage d'arbres sur $T(\Gamma)$. L'*image inverse* de $\mathcal{L}'$ par Φ est le langage d'arbres sur Σ défini par $\Phi^{-1}(\mathcal{L}') = \cup_{t \in \mathcal{L}'} \Phi^{-1}(t)$.

Exemple 14. *Soient* $\Sigma = \{f/2, g/1, a/0\}$, $\Gamma = \{h/1, a/0\}$ *et* Φ *le morphisme d'arbres déterminé par l'application* φ *définie par :*

$$\begin{aligned} \varphi(a) &= a \\ \varphi(g) &= h(x_1) \\ \varphi(f) &= h(x_2) \end{aligned}$$

Alors l'image par Φ *du langage d'arbres* $\{f(a, g^n(a)) \mid n \in \mathbb{N}\}$ *est le langage* $\{h^{n+1}(a) \mid n \in \mathbb{N}\}$.

Nous définissons maintenant différents types de morphismes d'arbres. Soit Φ un morphisme d'arbres de $T(\Sigma)$ dans $T(\Gamma)$ déterminé par une application φ. Le morphisme Φ est :

- **Linéaire** si pour tout symbole de fonction f de Σ_n, $\varphi(f)$ est un terme linéaire de $T(\Gamma, \mathcal{X}_n)$, c'est-à-dire un contexte de $\mathcal{C}^n(\Gamma)$.

- **Semi-linéaire** si pour tout symbole de fonction f de Σ_n, $\varphi(f)$ est un terme semi-linéaire de $T(\Gamma, \mathcal{X}_n)$.
- **ϵ-libre** si pour tout symbole de fonction f de Σ, f n'est pas réduit à une variable, c'est-à-dire $\varphi(f)$ n'est pas une variable de $\mathcal{X}$.
- **Symbole-symbole** si pour tout symbole de fonction f de Σ, il existe un symbole g de Γ d'arité p et $y_1, \ldots, y_p$ variables de $\mathcal{X}_n$ tels que $\varphi(f) = g(y_1, \ldots, y_p)$. Donc par application d'un morphisme d'arbres linéaire et symbole-symbole les étiquettes de l'arbre d'entrée sont modifiées, quelques sous-termes peuvent être effaçés et l'ordre des sous-termes peut être modifié.
- **Complet** si pour tout symbole de fonction f de Σ_n, $Var(t_f) = \mathcal{X}_n$.
- **Alphabétique** si Φ est un morphisme d'arbres complet, linéaire et symbole-symbole. En fait, un tel morphisme change les étiquettes de l'arbre d'entrée et peut modifier l'ordre des sous-termes.
- **Un réétiquetage** si pour tout symbole f de Σ_n, il existe un symbole g de Γ d'arité n tel que $\varphi(f) = g(x_1, \ldots, x_n)$. Donc seules les étiquettes de l'arbre d'entrée sont modifiées par application d'un tel morphisme.

Exemple 15. *Soient* $\Sigma = \{f/2, g/1, a/0\}$ *et* $\Gamma = \{f'/2, g'/1, a'/0\}$. *Considérons différents morphismes d'arbres* Φ *déterminés par différentes applications* φ.

$$\left.\begin{array}{l}\varphi(a) = a' \\ \varphi(g) = g'(x_1) \\ \varphi(f) = g'(f'(x_1, x_1))\end{array}\right\} \Phi \textit{ est ni linéaire, ni complet, ni symbole-symbole.}$$

$$\left.\begin{array}{l}\varphi(a) = a' \\ \varphi(g) = f'(x_1, x_1) \\ \varphi(f) = x_1\end{array}\right\} \Phi \textit{ est ni linéaire, ni } \epsilon\textit{-libre, ni complet.}$$

$$\left.\begin{array}{l}\varphi(a) = a' \\ \varphi(g) = f'(x_1, x_1) \\ \varphi(f) = g'(x_1)\end{array}\right\} \Phi \textit{ est symbole-symbole mais ni linéaire, ni complet.}$$

$$\left.\begin{array}{l}\varphi(a) = a' \\ \varphi(g) = g'(x_1) \\ \varphi(f) = f'(x_2, x_1)\end{array}\right\} \Phi \textit{ est alphabétique.}$$

$$\left.\begin{array}{l}\varphi(a) = a' \\ \varphi(g) = g'(x_1) \\ \varphi(f) = f'(x_1, x_2)\end{array}\right\} \Phi \textit{ est un réétiquetage.}$$

$$\left.\begin{array}{l}\varphi(a) = a' \\ \varphi(g) = g'(g'(x_1)) \\ \varphi(f) = f'(x_2, x_1)\end{array}\right\} \Phi \textit{ est linéaire et complet mais pas symbole-symbole.}$$

Les morphismes d'arbres étant introduits, nous définissons la notion de clôture par morphismes. Soit $\mathcal{CL}$ une classe de langages d'arbres. Alors la classe :

- $\mathcal{CL}$ est *close par morphismes d'arbres* si et seulement si $\Phi(\mathcal{L})$ appartient à $\mathcal{CL}$ pour tout langage $\mathcal{L}$ de $\mathcal{CL}$ et tout morphisme d'arbres Φ.
- $\mathcal{CL}$ est *close par morphismes d'arbres inverses* si et seulement si $\Phi^{-1}(\mathcal{L})$ appartient à $\mathcal{CL}$ pour tout langage $\mathcal{L}$ de $\mathcal{CL}$ et tout morphisme d'arbres Φ.

Ces définitions s'étendent à tous les sous-ensembles de morphismes d'arbres possibles. Par exemple, on dit que la classe $\mathcal{CL}$ est close par morphismes d'arbres alphabétiques inverses si et seulement si $\Phi^{-1}(\mathcal{L})$ appartient à $\mathcal{CL}$ pour tout langage $\mathcal{L}$ de $\mathcal{CL}$ et tout morphisme d'arbres alphabétique.

Chapitre 3

Systèmes de réécriture

Les définitions et notations de ce chapitre sont habituelles et peuvent être trouvées dans tout document relatif à la réécriture, en particulier dans l'article de synthèse de N. Dershowitz et J.P. Jouannaud [26] et le livre de F. Baader et T. Nipkow [5].

3.1 Définitions

Soient Σ une signature et $\mathcal{X}$ un ensemble dénombrable de variables. Une *règle de réécriture* est un couple (l, r) de termes de $T(\Sigma, \mathcal{X})$, noté en général $l \to r$, tel que $\mathcal{V}ar(l) \supseteq \mathcal{V}ar(r)$. On parle de *membre gauche* (respectivement de *membre droit*) d'une règle de réécriture $l \to r$ pour désigner l (respectivement r). Un ensemble de règles de réécriture est appelé *système de réécriture*. Par la suite, nous considérons toujours des systèmes de réécriture finis, c'est-à-dire ne comportant qu'un nombre fini de règles.

Un terme t de $T(\Sigma, \mathcal{X})$ se *réécrit* en un terme t' de $T(\Sigma, \mathcal{X})$ par un système de réécriture $\mathcal{R}$ si et seulement s'il existe une règle $l \to r$ de $\mathcal{R}$, une position p de $\mathcal{P}os(t)$ et une substitution σ sur $T(\Sigma, \mathcal{X})$ telles que $l\sigma = t|_p$ et $t' = t[r\sigma]_p$. On écrit alors $t \to_{\mathcal{R}} t'$ ou $t \to t'$ s'il n'y a pas d'ambiguïté pour $\mathcal{R}$.

On écrit également, si t et t' sont deux termes de $T(\Sigma, \mathcal{X})$, $\mathcal{R}$ un système de réécriture, r une règle de réécriture et p une position de $\mathcal{P}os(t)$:

$t \underset{p}{\to} t'$ si $t \to t'$ par application d'une règle de $\mathcal{R}$ à la position p de t;

$t \xrightarrow{r} t'$ si $t \to t'$ par application de la règle r;

$t \underset{p}{\xrightarrow{r}} t'$ si $t \to t'$ par application de la règle r à la position p de t;

$t \not\to t'$ si $\neg(t \to t')$.

$\to_{\mathcal{R}}$ et $\xrightarrow{r}$ sont des relations binaires sur $T(\Sigma, \mathcal{X})$ dites *relations de réécriture* car elles sont closes :

- **Par contextes** c'est-à-dire si $t \to_{\mathcal{R}} t'$ (respectivement $t \xrightarrow{r} t'$), alors $C[t] \to_{\mathcal{R}} C[t']$ (respectivement $C[t] \xrightarrow{r} C[t']$) pour tout contexte C de $\mathcal{C}(\Sigma)$ et,
- **Par substitutions** c'est-à-dire si $t \to_{\mathcal{R}} t'$ (respectivement $t \xrightarrow{r} t'$), alors $t\sigma \to_{\mathcal{R}} t'\sigma$ (respectivement $t\sigma \xrightarrow{r} t'\sigma$) pour toute substitution σ sur $T(\Sigma, \mathcal{X})$.

Pour toute relation de réécriture, sa clôture réflexive, sa clôture transitive et sa clôture réflexive et transitive sont également des relations de réécriture.

Un terme t de $T(\Sigma, \mathcal{X})$ est dit *réductible* par un système de réécriture $\mathcal{R}$ s'il existe un terme t' de $T(\Sigma, \mathcal{X})$ tel que $t \rightarrow t'$. Un terme non réductible par $\mathcal{R}$ est *irréductible* par $\mathcal{R}$, ou en *forme normale* pour $\mathcal{R}$ ou encore une forme normale de $\mathcal{R}$. Un terme t' est une forme normale d'un terme t, si $t \xrightarrow{*}_{\mathcal{R}} t'$ et t' est une forme normale de $\mathcal{R}$.

Une *chaîne de dérivation* ou *dérivation* dans un système de réécriture $\mathcal{R}$ est une séquence de réécritures $t_0 \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_k \rightarrow_{\mathcal{R}} \ldots$. Pour une telle dérivation, les termes $t_1, \ldots, t_k$ sont appelés *formes sententielles* de t_0 dans $\mathcal{R}$. En fait, l'ensemble des formes sententielles de t_0 dans $\mathcal{R}$ est l'ensemble des termes t' tels que $t_0 \xrightarrow{*}_{\mathcal{R}} t'$. On définit alors l'ensemble des formes sententielles dans $\mathcal{R}$ d'un langage $\mathcal{L}$ sur Σ comme l'ensemble des formes sententielles des termes de $\mathcal{L}$ c'est-à-dire $\{t' \in T(\Sigma) \mid \exists t \in \mathcal{L}, t \xrightarrow{*}_{\mathcal{R}} t'\}$.

3.2 Règles et systèmes particuliers

Une règle de réécriture est *close à gauche* (respectivement *à droite*) si son membre gauche (respectivement droit) est clos. Elle est *close* si elle est close à gauche et à droite. Un système de réécriture est *clos à gauche* (respectivement *à droite*) si toutes ses règles sont closes à gauche (respectivement à droite). Il est *clos* si toutes ses règles sont closes.

Une règle de réécriture est *linéaire à gauche* (respectivement *à droite*) si son membre gauche (respectivement droit) est linéaire. Elle est *linéaire* si elle est linéaire à gauche et à droite. Un système de réécriture est *linéaire à gauche* (respectivement *à droite*) si toutes ses règles sont linéaires à gauche (respectivement à droite). Il est linéaire si toutes ses règles sont linéaires.

Une règle de réécriture est *semi-linéaire à gauche* (respectivement *à droite*) si son membre gauche (respectivement droit) est semi-linéaire. Elle est *semi-linéaire* si elle est linéaire à gauche et à droite. Un système de réécriture est *semi-linéaire à gauche* (respectivement *à droite*) si toutes ses règles sont semi-linéaires à gauche (respectivement à droite). Il est semi-linéaire si toutes ses règles sont semi-linéaires.

Une règle de réécriture $l \rightarrow r$ est *effaçante* si $\mathcal{V}ar(r) \subsetneq \mathcal{V}ar(l)$, ou si l et r sont clos et $card(\mathcal{P}os(r))$ est strictement inférieur à $card(\mathcal{P}os(l))$.

Une règle de réécriture $l \rightarrow r$ est *mince* si pour toute position p de variables (p appartient à $\mathcal{VP}os(l)$ ou $\mathcal{VP}os(r)$), la profondeur de p est inférieure ou égale à un. Un système de réécriture est *mince* si toutes ses règles sont minces.

Une règle de réécriture $l \rightarrow r$ est *ultra-mince* si toutes les occurrences de variables apparaissant dans ses membres gauche et droit sont à la profondeur un. Un système de réécriture est *ultra-mince* si toutes ses règles sont ultra-minces.

Finalement une règle de réécriture $l \rightarrow r$ est *un réétiquetage* si il existe n entier, et f et g symboles de Σ tels que $l = f(x_1, \ldots, x_n)$ et $r = g(x_1, \ldots, x_n)$.

3.3 Propriétés

3.3.1 Terminaison

Un système de réécriture est *nœthérien* s'il n'existe pas de chaîne de dérivation infinie dans le système. On dit également que le système *termine*. Le problème de terminaison, énoncé ci-dessous, est en général indécidable.

Problème de terminaison
Entrée : *Un système de réécriture* $\mathcal{R}$.
Question : $\mathcal{R}$ *est-il nœthérien ?*

Ce problème reste indécidable pour les systèmes constitués d'une unique règle de réécriture linéaire à gauche [22] mais devient décidable pour les systèmes clos à droite [25]. N. Dershowitz et J.P. Jouannaud dans [26] et F. Baader et T. Nipkow dans [5] décrivent des méthodes permettant de prouver qu'un système de réécriture est nœthérien. L'une de ces méthodes consiste à utiliser un ordre de simplification (définition 16 et théorème 17). Nous détaillons cette méthode car elle est utilisée dans la preuve du lemme 240 de ce document.

Soient Σ une signature et $\mathcal{X}$ un ensemble dénombrable de variables.

Définition 16 (F. Baader et T. Nipkow [5]). *Un ordre strict* $\succ$ *sur* $T(\Sigma, \mathcal{X})$ *est un ordre de simplification si et seulement si* $\succ$ *satisfait les propriétés suivantes :*

- *Clôture par contextes :*

$$\forall s_1, s_2 \in T(\Sigma, \mathcal{X}), \forall C \in \mathcal{C}(Sigma),\ s_1 \succ s_2 \Rightarrow C(s_1) \succ C(s_2).$$

- *Clôture par substitutions :*

$$\forall s_1, s_2 \in T(\Sigma, \mathcal{X}), \forall \sigma \in \mathcal{S}ub(T(\Sigma, \mathcal{X})),\ s_1 \succ s_2 \Rightarrow s_1\sigma \succ s_2\sigma.$$

- *Propriété des sous-arbres :* $\forall t \in T(\Sigma, \mathcal{X}), \forall p \in \mathcal{P}os(t) \setminus \{\epsilon\},\ t \succ t|_p$.

Théorème 17. *Soit* $\mathcal{R}$ *un système de réécriture sur* Σ. *S'il existe un ordre de simplication* $\succ$ *sur* $T(\Sigma, \mathcal{X})$ *tel que* $l \succ r$ *pour toute règle* $l \to r$ *de* $\mathcal{R}$ *alors* $\mathcal{R}$ *est nœthérien.*

Plusieurs types d'ordres de simplification existent : les ordres de simplification polynomiaux, les ordres récursifs de chemins, les ordres de Knuth-Bendix. Nous rappelons ci-dessous la construction d'un ordre récursif de chemins particulier : l'ordre lexicographique de chemins (définition 18) Un tel ordre est défini récursivement. En fait, deux termes sont comparés tout d'abord en comparant leurs symbole de tête, puis en comparant récursivement leurs sous-termes directs en les considérant comme des tuples ordonnés.

Définition 18. *Soit* $\succ$ *un ordre strict sur* Σ. *L'ordre lexicographique de chemins* $\succ_{lpo}$ *sur* $T(\Sigma, \mathcal{X})$ *induit par* $\succ$ *est défini comme suit. Pour tous termes* s *et* t *de* $T(\Sigma, \mathcal{X})$, *nous avons* $s \succ_{lpo} t$ *si et seulement si :*

- $t \in \mathcal{V}ar(s)$ *et* $s \neq t$, *ou* (LPO1)
- $s = f(s_1, \ldots, s_m), t = g(t_1, \ldots, t_n)$ *et* (LPO2)
  - *Il existe* i *dans* $[m]$ *tel que* $s_i \succeq_{lpo} t$, *ou* (LPO2a)
  - $f \succ g$ *et* $s \succ_{lpo} t_j$ *pour tout* j *de* $[n]$, *ou* (LPO2b)
  - $f = g, s \succ_{lpo} t_j$ *pour tout* j *de* $[n]$, *et il existe* i *dans* $[m]$, *tel que* $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$ *et* $s_i \succ_{lpo} t_i$ (LPO2c)

Tout ordre lexicographique de chemins est un ordre de simplication donc nous déduisons du théorème 17, le théorème suivant.

Théorème 19. *Soit $\mathcal{R}$ un système de réécriture sur Σ. S'il existe un ordre lexicographique de chemins $\succ_{lpo}$ tel que $l \succ_{lpo} r$ pour toute règle $l \to r$ de $\mathcal{R}$, alors $\mathcal{R}$ est nœthérien.*

3.3.2 Confluence

Un système de réécriture est *confluent* si :

$$\forall x, y_1, y_2 \in T(\Sigma, \mathcal{X}), (x \xrightarrow{*} y_1 \wedge x \xrightarrow{*} y_2) \Rightarrow \exists z \in T(\Sigma, \mathcal{X}), (y_1 \xrightarrow{*} z \wedge y_2 \xrightarrow{*} z)$$

Le problème de confluence, énoncé ci-dessous, est en général indécidable.

Problème de confluence
Entrée : *Un système de réécriture $\mathcal{R}$.*
Question : *$\mathcal{R}$ est-il confluent ?*

Ce problème devient décidable pour les systèmes clos [24]. G. Huet dans [44] et, F. Baader et T. Nipkow dans [5] décrivent des méthodes permettant de prouver si un système de réécriture est confluent. Nous donnons ci-dessous les notions de base de la méthode des paires critiques (D.E. Knuth et P.B. Bendix [51], G. Huet [44]) utilisée dans la preuve du lemme 240 de ce document.

Soient Σ une signature, $\mathcal{X}$ un ensemble dénombrable de variables et $\mathcal{R}$ un système de réécriture.

Définition 20. *Soient $l_1 \to r_1$ et $l_2 \to r_2$ deux règles de $\mathcal{R}$ telles que l_2 ne soit pas une variable et dont les variables ont été renommées de sorte que $Var(l_1) \cap Var(l_2) = \emptyset$. Soit p une position de l_1 telle que $l_1|_p$ ne soit pas une variable et telle que $l_1|_p$ et l_2 s'unifient. Alors si θ est l'un des unificateurs les plus généraux de $l_1|_p$ et l_2, $\langle r_1\theta, (l_1\theta)[r_2\theta]_p \rangle$ est appelé une* paire critique *de $\mathcal{R}$.*

Exemple 21. *Soit les règles (1) $:= f(f(x,y),z) \to f(x,f(y,z))$ et (2) $:= f(i(x_1),x_1) \to e$. Le sous-arbre $f(x,y)$ du membre gauche de la règle (1) et le membre gauche de la règle (2) s'unifient et $\theta = \{x \leftarrow i(x_1), y \leftarrow x_1\}$ est l'un de leurs unificateurs les plus généraux. La paire critique associée à θ est $\langle f(i(x_1), f(x_1,z)), f(e,z) \rangle$.*

Nous disons que deux termes t_1 et t_2 de $T(\Sigma, \mathcal{X})$ sont *joignables* si et seulement s'il existe un terme s tel que $t_1 \xrightarrow{*} s$ et $t_2 \xrightarrow{*} s$. Nous avons alors le théorème suivant :

Théorème 22. *Un système de réécriture nœthérien est confluent si et seulement si toutes ses paires critiques sont joignables.*

Pour terminer ce chapitre, nous donnons une définition pour les systèmes de réécriture combinant les propriétés de terminaison et de confluence.

Définition 23. *Un système de réécriture est* convergent *s'il est nœthérien et confluent.*

Chapitre 4

Logique des prédicats du premier ordre

Dans ce chapitre, nous présentons la logique des prédicats du premier ordre que nous utilisons dans le chapitre **Théorie de la réécriture en un pas** de la partie **Réécriture** et le chapitre **Indécidabilité** de la partie **Contraintes ensemblistes**. Dans un premier temps, nous définissons les formules constituant cette logique (section 4.1). Puis nous donnons la sémantique de cette logique (section 4.2). Pour la suite de ce chapitre, nous considérons une signature Σ et un ensemble dénombrable de variables $\mathcal{X}$.

4.1 Formules

Soit $\mathcal{P}$ un ensemble de symboles appelés *prédicats*. L'ensemble des formules de la logique du premier ordre construites sur Σ, $\mathcal{X}$ et $\mathcal{P}$ est défini comme suit. Une *formule atomique* est une expression de la forme $R(t_1, \ldots, t_n)$ où $t_1, \ldots, t_n$ sont des termes de $T(\Sigma, \mathcal{X})$ et R un symbole de prédicat de $\mathcal{P}$ d'arité n.

L'ensemble des *formules du premier ordre* est alors le plus petit ensemble vérifiant :

- Toute formule atomique est une formule;
- Si φ et φ' sont deux formules, alors $\neg\varphi$ (négation), $\varphi \vee \varphi'$ (disjonction), $\varphi \wedge \varphi'$ (conjonction), $\varphi \Rightarrow \varphi'$ (implication) et $\varphi \Leftrightarrow \varphi'$ (équivalence) sont des formules;
- Si φ est une formule et x une variable, alors $\exists x\varphi$ (quantification existentielle) et $\forall x\varphi$ (quantification universelle) sont des formules.

Une *sous-formule* d'une formule φ est une suite consécutive de symboles de φ formant une formule.

Une occurrence o d'une variable x dans une formule φ est dite *liée* s'il existe une sous-formule ψ de φ contenant o telle que ψ commence par $\exists x$ ou $\forall x$. Une occurrence de x dans φ est dite *libre* si elle n'est pas liée. Pour une formule φ, nous notons $\mathcal{V}_l(\varphi)$ l'ensemble des variables libres de φ. Une *formule close* est une formule dans laquelle toutes les occurrences de variables sont liées.

Exemple 24. *Soit la signature* $\Sigma_{\mathbb{N}} = \{a/0, s/1\}$ *et un symbole de prédicat binaire* R. *La formule* $\psi := \exists x \forall z\ R(x,z)$ *est une formule close et est sous-formule de la formule* $\varphi := \exists y\ R(x,y) \wedge \exists x \forall z\ R(x,z)$. *Dans* φ, *les occurrences de* y *et* z, *et la seconde occurrence de* x *sont liées tandis que la première occurrence de* x *est libre.*

4.2 Structure

Une structure fournit une interprétation aux symboles de fonction et aux symboles de prédicat. Formellement, une $(\Sigma, \mathcal{P})$-*structure* $\mathcal{A}$ consiste en :

1. Un ensemble non vide A appelé *support*;
2. Une application qui associe à toute constante e de Σ, un élément $e^{\mathcal{A}}$ de A;
3. Une application qui associe à tout symbole de fonction f de Σ d'arité n strictement positive une fonction $f^{\mathcal{A}}$ de A^n dans A;
4. Une application qui associe à tout symbole de prédicat R de $\mathcal{P}$ d'arité n, une relation $R^{\mathcal{A}}$ sur A d'arité n (c'est-à-dire un prédicat d'arité n).

Soit $\mathcal{A}$ une $(\Sigma, \mathcal{P})$-structure. Une *valuation* v sur le support A de $\mathcal{A}$ est une application d'un ensemble de variables $V_{\mathcal{X}}$ ($V_{\mathcal{X}} \subseteq \mathcal{X}$) vers A. Lorsque $V_{\mathcal{X}}$ est fini, nous désignons v par $\{x_1 \leftarrow d_1^{\mathcal{A}}, \ldots, x_n \leftarrow d_n^{\mathcal{A}}\}$ si $V_{\mathcal{X}} = \{x_1, \ldots, x_n\}$ et pour tout entier i de $[n]$, $v(x_i) = d_i^{\mathcal{A}}$.

L'*interprétation* d'un terme t de $T(\Sigma, \mathcal{X})$ dans une structure $\mathcal{A}$ sous une valuation v de $Var(t)$ dans le domaine A de $\mathcal{A}$ est un élément de A, noté $\lfloor t \rfloor_\sigma^{\mathcal{A}}$, défini inductivement par :

- Si $t = x$ avec x variable, alors $\lfloor t \rfloor_v^{\mathcal{A}} = v(x)$;
- Si $t = e$ avec e constante, alors $\lfloor t \rfloor_v^{\mathcal{A}} = e^{\mathcal{A}}$;
- Si $t = f(t_1, \ldots, t_n)$ avec n entier strictement positif et $t_1, \ldots, t_n$ termes de $T(\Sigma, \mathcal{X})$, alors $\lfloor t \rfloor_v^{\mathcal{A}} = f^{\mathcal{A}}(\lfloor t_1 \rfloor_v^{\mathcal{A}}, \ldots, \lfloor t_n \rfloor_v^{\mathcal{A}})$.

Exemple 25. *Soient la signature* $\Sigma_{\mathbb{N}}$ *et le prédicat binaire* R *introduits dans l'exemple 24. Dans la structure* $\mathcal{N}$ *de domaine* $\mathbb{N}$ *associant* 0 *à la constante* a, *la fonction* $x \to x+1$ *au symbole* s *et la relation binaire "inférieur ou égal"* $\leq$ *sur* $\mathbb{N}$ *au symbole de prédicat* R, *l'interprétation de* $s^n(a)$ *est* n *pour tout entier* n.

Si x *est une variable et* v *une valuation telle que* $v(x) = s(a)$, *alors l'interprétation de* $s^n(x)$ *dans* $\mathcal{N}$ *sous* v *est* $n+1$ *pour tout entier* n.

Nous rappelons maintenant dans quelle mesure une formule est satisfiable dans une structure donnée.

Soient $\mathcal{A}$ une $(\Sigma, \mathcal{P})$-structure, φ une formule du premier ordre sur Σ, $\mathcal{X}$ et $\mathcal{P}$, et v une valuation de $\mathcal{V}_l(\varphi)$ dans A. Alors φ est *satisfaite* dans la structure $\mathcal{A}$ sous la valuation v et on note $\mathcal{A}, v \models \varphi$ si et seulement si :

- φ est une formule atomique $R(t_1, \ldots, t_n)$ et $R^{\mathcal{A}}(\lfloor t_1 \rfloor_v^{\mathcal{A}}, \ldots, \lfloor t_n \rfloor_v^{\mathcal{A}})$.
- φ est une négation $\neg\psi$ et $\mathcal{A}, v \not\models \psi$.
- φ est une conjonction $\varphi_1 \wedge \varphi_2$, et $\mathcal{A}, v \models \varphi_1$ et $\mathcal{A}, v \models \varphi_2$.
- φ est une disjonction $\varphi_1 \vee \varphi_2$, et $\mathcal{A}, v \models \varphi_1$ ou $\mathcal{A}, v \models \varphi_2$.
- φ est une implication $\varphi_1 \Rightarrow \varphi_2$, et $\mathcal{A}, v \not\models \varphi_1$ ou $\mathcal{A}, v \models \varphi_2$.
- φ est une équivalence $\varphi_1 \Leftrightarrow \varphi_2$, et soit $\mathcal{A}, v \models \varphi_1$ et $\mathcal{A}, v \models \varphi_2$, soit $\mathcal{A}, v \not\models \varphi_1$ et $\mathcal{A}, v \not\models \varphi_2$.
- φ est une quantification existentielle $\exists x\psi$ et il existe une valuation v' de $\mathcal{V}_l(\psi)$ dans A telle que pour toute variable y différente de x, $v'(y) = v(y)$ et $\mathcal{A}, v' \models \psi$.
- φ est une quantification universelle $\forall x\psi$ et pour toute valuation v' de $\mathcal{V}_l(\psi)$ dans A telle que pour toute variable y différente de x, $v'(y) = v(y)$ et $\mathcal{A}, v' \models \psi$.

La formule φ est dite *satisfiable* dans la structure $\mathcal{A}$ et on note $\mathcal{A} \models \varphi$ s'il existe une valuation v de $\mathcal{V}_l(\varphi)$ dans A telle que $\mathcal{A}, v \models \varphi$.

Si pour toute valuation v de $\mathcal{V}_l(\varphi)$ dans A, $\mathcal{A}, v \models \varphi$, $\mathcal{A}$ est appelé un *modèle* de φ.

Exemple 26. *Reprenons les éléments introduits dans l'exemple 25. Soit v une valuation de $\{x\}$ dans $\mathbb{N}$. Il existe un entier n tel que $v(x) = n$. Alors $\lfloor 0 \rfloor_v^{\mathcal{N}} = 0$ et $\lfloor x \rfloor_v^{\mathcal{N}} = n$ d'où $\lfloor 0 \rfloor_v^{\mathcal{N}} \leq \lfloor x \rfloor_v^{\mathcal{N}}$. Donc pour toute valuation v de $\{x\}$ dans $\mathbb{N}$, nous avons $\mathcal{N}, v \models R(0, x)$. Nous en déduisons que la formule $\forall x\, R(0, x)$ est satisfiable dans $\mathcal{N}$.*

Par contre la formule $\exists x\, (R(x, 0) \wedge R(s(0), x))$ est insatisfiable dans $\mathcal{N}$ puisque pour tout entier n, on ne peut avoir $n \leq 0$ et $1 \leq n$.

Soient φ et ψ deux formules du premier ordre. φ est *conséquence logique* de ψ (noté $\psi \models \varphi$) si tout modèle $\mathcal{A}$ de ψ est un modèle de φ. Si $\psi \models \varphi$ et $\varphi \models \psi$, alors φ et ψ sont dites *équivalentes* et on note $\varphi \equiv \psi$.

Une formule φ est dite en forme *prénexe* si tous ses quantificateurs apparaissent en son début, c'est-à-dire φ s'écrit $Q\varphi'$ où φ' est une formule ne comportant pas de quantificateurs et Q est une suite de quantifications. Pour toute formule close φ, il existe (au moins) une formule close φ_p en forme prénexe qui lui est équivalente. On peut alors distinguer les formules selon la forme de la quantification de leur(s) forme(s) prénexe(s). Pour une forme donnée, l'ensemble des formules qui mises en forme prénexe ont cette forme de quantification est appelé *fragment*.

Exemple 27. *La formule* $\varphi := \forall x \forall y \, [\exists z \, (R(x,z) \wedge r(y,z)) \Rightarrow \exists u \, Q(x,y,u)]$ *a pour prénexe* $\forall x \forall y \forall z \exists u \, [\neg(R(x,z) \wedge r(y,z)) \vee Q(x,y,u)]$. *La formule* φ *appartient donc au fragment* $\forall^* \exists^*$ *(ou plus précisément* $\forall^3 \exists$*) de la logique du premier ordre.*

L'un des problèmes classiques de la logique des prédicats du premier ordre est celui de la *décidabilité* de cette logique.

Formellement, étant donné une $(\Sigma, \mathcal{P})$-structure $\mathcal{A}$, la logique des prédicats du premier ordre est décidable pour la structure $\mathcal{A}$ si pour toute formule φ de cette logique, on peut décider si $\mathcal{A} \models \varphi$.

Chapitre 5

Topologie

Dans ce chapitre, nous faisons quelques rappels de topologie que nous utilisons dans le chapitre **Topologie - Expressivité** de la partie **Contraintes ensemblistes**. Nous rappelons tout d'abord la notion d'espace topologique (section 5.1). Puis nous rappelons que l'on peut associer un espace topologique à toute distance (section 5.2). Finalement nous définissons une distance sur l'ensemble $(2^{T(\Sigma)})^n$ où Σ est une signature et n un entier strictement positif (section 5.3).

La plupart des notions développées dans ce chapitre se retrouvent dans les livres de topologie et en particulier dans le livre de J. Dugundji [27].

5.1 Espace topologique

On appelle *topologie* sur un ensemble E toute famille Top_E de sous-ensembles de E vérifiant les propriétés suivantes :

- $\emptyset$ et E sont éléments de Top_E;
- Toute union d'éléments de Top_E est un élément de Top_E;
- Toute intersection finie d'éléments de Top_E est un élément de Top_E.

Un couple (E, Top_E) constitué d'un ensemble E et d'une topologie Top_E sur E est appelé un *espace topologique* et les éléments de Top_E sont appelés les *ouverts* de l'espace topologique (E, Top_E).

Soit (E, Top_E) un espace topologique. Un sous-ensemble F de E est dit *fermé* si $\complement_E F$ est un ouvert. L'ensemble $\mathcal{F}$ des fermés de l'espace topologique (E, Top_E) vérifie les propriétés suivantes :

- $\emptyset$ et E sont éléments de $\mathcal{F}$;
- Toute intersection d'éléments de $\mathcal{F}$ est un élément de $\mathcal{F}$;
- Toute union finie d'éléments de $\mathcal{F}$ est un élément de $\mathcal{F}$.

Soient (E, Top_E) et (G, Top_G) deux espaces topologiques. Une application $f : E \to G$ est dite *continue* si l'image inverse par f de tout ouvert de (G, Top_G) est un ouvert de (E, Top_E).

Puisque les ouverts et les fermés d'un espace topologique sont duaux par complément, nous avons de façon évidente qu'une application $f : E \to G$ est continue si et seulement si l'image inverse par f de tout fermé de (G, Top_G) est un fermé de (E, Top_E).

Finalement nous considérons (H, Top_H) un espace topologique et, $f : E \to G$ et $g : G \to H$ deux applications. Alors si f et g sont continues, leur composée $g \circ f : E \to H$ est continue.

5.2 Espace topologique associé à une distance

Soit E un ensemble. Une *distance* sur E est une application $d : E \times E \to \mathbb{R}$ telle que pour tout x, y, z de E :

- $d(x, y) \geq 0$;
- $d(x, y) = 0$ si et seulement si $x = y$;
- $d(x, y) = d(y, x)$ (d est symétrique);
- $d(x, z) \leq d(x, y) + d(y, z)$ (d satisfait l'*inégalité triangulaire*).

$d(x, y)$ est appelé distance entre x et y, et un couple (E, d) constitué d'un ensemble E et d'une distance d est appelé *espace métrique.*

Dans la suite, nous considérons un espace métrique (E, d) et nous rappelons comment associer un espace topologique à (E, d) (théorème 28).

Tout d'abord, nous définissons les notions de boules ouvertes et de boules fermés. Soit x un élément de E et r un réel strictement positif. L'ensemble $\mathcal{B}(x, r) = \{y \mid d(x, y) < r\}$ est appelé *boule ouverte* de centre x et de rayon r. Soit r un réel positif, alors l'ensemble $\mathcal{B}_F(x, r) = \{y \mid d(x, y) \leq r\}$ est appelé *boule fermée* de centre x et de rayon r. De façon évidente $\mathcal{B}(x, 0) = \emptyset$, et $\mathcal{B}(x, r) \subseteq \mathcal{B}(x, r')$ si $r \leq r'$.

Théorème 28. *L'ensemble Top_E des sous-ensembles A de E satisfaisant:*

- *Soit $A = \emptyset$,*
- *Soit pour tout x de A, il existe un réel r strictement positif tel que $\mathcal{B}(x, r) \subseteq A$,*

définit un espace topologique (E, Top_E).

Remarquons que toute boule ouverte (respectivement toute boule fermée) de E est un ouvert (respectivement un fermé) de l'espace topologique (E, Top_E).

Nous nous intéressons maintenant aux produits d'espaces métriques et à la continuité des applications dans ces espaces.

Soient n un entier strictement positif et $(E_1, d_1), \ldots, (E_n, d_n)$ des espaces métriques. Alors l'application d définie ci-dessous est une distance sur l'ensemble $E_1 \times \cdots \times E_n$.

$$\begin{array}{rcl} d : (E_1 \times \cdots \times E_n) \times (E_1 \times \cdots \times E_n) & \to & \mathbb{R} \\ ((S_1, \ldots, S_n), (S'_1, \ldots, S'_n)) & \to & max(\{d_i(S_i, S'_i) \mid i \in [n]\}) \end{array}$$

On peut donc associer un espace topologique à l'espace produit $E_1 \times \cdots \times E_n$. Considérons maintenant un autre espace métrique (E, d), pour tout i de $[n]$ une application $f_i : E \to E_i$ et l'application f suivante :

$$\begin{array}{rcl} f : E & \to & E_1 \times \cdots \times E_n \\ x & \to & (f_1(x), \ldots, f_n(x)) \end{array}$$

Alors f est continue si et seulement si f_i est continue pour tout i de $[n]$. Dans la suite toute application f définie comme ci-dessus à partir de n applications f_i sera notée $(f_1, \ldots, f_n)$. Ce résultat de continuité et cette notation seront utilisés dans le chapitre **Topologie - Expressivité** de la partie **Contraintes ensemblistes.**

Finalement nous rappelons rapidement la notion de suite dans un espace métrique. Une *suite* dans E est une famille $(u_n)_{n\in\mathbb{N}}$ d'éléments de E. Une suite (u_n) de l *converge* vers un élément l de E si $\lim_{n\to\infty} d(u_n, l) = 0$ c'est-à-dire :

$$\forall \eta > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0,\ d(u_n, l) < \eta.$$

Les suites convergentes dans un fermé F ont la particularité de converger vers un élément de F :

Théorème 29. *Soit F un fermé de E et (u_n) une suite dans F. Si (u_n) converge vers l, alors l est un élément de F.*

5.3 Distance sur l'ensemble $\left(2^{T(\Sigma)}\right)^n$

Dans cette section, nous utilisons la méthode usuelle de définition d'une distance sur un produit cartésien d'espaces métriques pour définir une distance sur l'ensemble $\left(2^{T(\Sigma)}\right)^n$ où Σ est une signature et n un entier strictement positif.

Tout d'abord nous définissons une distance sur $2^{T(\Sigma)}$. Soit l'application d_1 définie par :

$$\begin{array}{rcl} d_1 : 2^{T(\Sigma)} \times 2^{T(\Sigma)} & \to & \mathbb{R} \\ (S, S') & \to & 0 \text{ si } S = S' \\ & \to & 2^{-m} \text{ si } S \neq S' \text{ avec } m = min(\{Haut(t) \mid t \in S \Delta S'\}) \end{array}$$

Proposition 30. *d_1 est une distance sur $2^{T(\Sigma)}$.*

Preuve : d_1 satisfait trivialement les trois premières propriétés des distances. Montrons que d_1 satisfait l'inégalité triangulaire. Soient S^1, S^2, S^3 trois éléments de $2^{T(\Sigma)}$. Nous distinguons deux cas suivant la valeur de $d_1(S^1, S^3)$.

Premier cas $d(S^1, S^3) = 0$: Puisque $d(S^1, S^2) \geq 0$ et $d(S^2, S^3) \geq 0$, l'inégalité $d_1(S^1, S^3) \leq d_1(S^1, S^2) + d_1(S^2, S^3)$ est satisfaite ce qui termine le premier cas.

Second cas $d_1(S^1, S^3) = 2^{-m}$: Alors il existe un terme t de $T(\Sigma)$ de hauteur m appartenant à $S^1 \Delta S^3$. Tout d'abord montrons par l'absurde que t appartient à $(S^1 \Delta S^2) \cup (S^2 \Delta S^3)$.

Supposons que t n'appartienne pas à $(S^1 \Delta S^2) \cup (S^2 \Delta S^3)$. Comme t appartient à $S^1 \Delta S^3$, il appartient soit à S^1, soit à S^3. Supposons que t appartienne à S^1. Alors t appartient

à S^2 puisque d'après l'hypothèse faite, t n'appartient pas à $S^1 \Delta S^2$. Donc t appartient à S^3 puisque t n'appartient pas à $S^2 \Delta S^3$. Finalement t n'appartient pas à $S^1 \Delta S^3$ ce qui contredit l' hypothèse. Nous obtenons une contradiction similaire en supposant que t appartienne à S^3. Donc l'hypothèse faite est fausse, d'où t appartient à $(S^1 \Delta S^2) \cup (S^2 \Delta S^3)$.

Nous en déduisons que $m \geq min(\{Haut(t) \mid t \in S^1 \Delta S^2\})$ ou $m \geq min(\{Haut(t) \mid t \in S^2 \Delta S^3)\})$. Donc $d_1(S^1, S^3) \leq d_1(S^1, S^2)$ ou $d_1(S^1, S^3) \leq d_1(S^2, S^3)$. Finalement $d_1(S^1, S^3) \leq d_1(S^1, S^2) + d_1(S^2, S^3)$.

Donc d_1 satisfait l'inégalité triangulaire. Finalement d_1 est une distance sur $2^{T(\Sigma)}$. □

Soit n un entier strictement positif. L'application d_1 étant une distance sur $2^{T(\Sigma)}$, nous en déduisons que l'application d_n définie ci-dessous est une distance sur $\left(2^{T(\Sigma)}\right)^n$.

$$\begin{array}{rcl} d_n : \left(2^{T(\Sigma)}\right)^n \times \left(2^{T(\Sigma)}\right)^n & \rightarrow & \mathbb{R} \\ ((S_1, \ldots, S_n), (S'_1, \ldots, S'_n)) & \rightarrow & max(\{d_1(S_i, S'_i) \mid i \in [n]\}) \end{array}$$

De façon évidente, pour tous n-uplets $S = (S_1, \ldots, S_n)$ et $S' = (S'_1, \ldots, S'_n)$ de $\left(2^{T(\Sigma)}\right)^n$, on a :

- $d_n(S, S') = 0$ si $S = S'$,
- Sinon $d_n(S, S') = 2^{-m}$ avec $m = min(\{Haut(t) \mid \exists i \in [n], t \in S_i \Delta S'_i\})$.

Finalement remarquons que d_n est une distance *ultra-métrique* puisque pour tous éléments S^1, S^2, S^3 de $\left(2^{T(\Sigma)}\right)^n$:

$$d_n(S^1, S^3) \leq max(\{d_n(S^1, S^2), d_n(S^2, S^3)\}).$$

Deuxième partie

Outils de décidabilité et d'indécidabilité

Cette partie, décomposée en trois chapitres, est consacrée aux outils de décidabilité et d'indécidabilité que nous utilisons dans ce document pour démontrer nos différents résultats de décision.

Le premier chapitre rappelle les propriétés de clôture et les résultats de décision pour les automates d'arbres finis standards. Le second est consacré aux automates à contraintes, automates introduits pour la résolution de problèmes non linéaires. Il rappelle les différentes classes d'automates à contraintes existantes ainsi que leurs propriétés et certaines de leurs applications. Nous établissons également dans ce chapitre un nouveau résultat puisque nous montrons que l'image d'un régulier par un morphisme d'arbres semi-linéaire est un langage reconnu par automates à tests d'égalité entre frères. Finalement, dans le dernier chapitre, nous adaptons la méthode de la grille aux arbres afin d'obtenir un outil de preuves d'indécidabilité appelé méthode des Arbres-Grilles. En fait, nous définissons un ensemble d'arbres particuliers codant les calculs réussis d'une machine de Turing et donc de coder le problème indécidable de l'arrêt d'une machine de Turing.

Chapitre 1

Automates d'arbres finis

Les automates d'arbres sont une extension des automates de mots au cas des signatures possédant des symboles d'arité quelconque puisque les mots sur un alphabet fini peuvent être vus comme des termes composés de symboles de fonction unaires.

Dans ce chapitre, nous présentons les définitions et résultats de base concernant les automates d'arbres. Nous nous restreignons au cas des automates d'arbres finis ascendants et nous définissons les automates comme des systèmes de réécriture particuliers. Tous les résultats de ce chapitre peuvent être retrouvés dans les ouvrages traitant des automates d'arbres finis, et en particulier dans la thèse de F.Jacquemard [46] et le livre sur les automates d'arbres de H. Comon *et al.* [20].

1.1 Définitions de base

Définition 31. *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres finis *est un quadruplet $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini de symboles d'arité 1 appelés* états *($\Sigma \cap \mathcal{Q} = \emptyset$);*
3. *$\mathcal{Q}_f \subseteq \mathcal{Q}$ est l'ensemble des* états finaux*;*
4. *Δ est un système de réécriture dont les règles sont de la forme :*

$$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$$

où f est un symbole de Σ_n, n entier, $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$ et $x_1, \ldots, x_n$ des variables distinctes de $\mathcal{X}$.

Les règles de Δ sont appelées *règles de transition* et Δ *ensemble des règles de transition.*

Remarque 32. *Dans la suite de ce document, on utilise les expressions* automate d'arbres standard *et AAS pour désigner un automate d'arbres finis.*

Soient $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ un *AAS*, t un terme clos de $T(\Sigma)$ et q un état de $\mathcal{Q}$. Si $t \xrightarrow{*}_\Delta q(t)$, on dit que t se *dérive* en l'état q et si q est un état final de $\mathcal{Q}_f$, t est dit *reconnu* ou *accepté* par $\mathcal{A}$. Dans la suite, nous écrivons $\to_\mathcal{A}$ (respectivement $\xrightarrow{*}_\mathcal{A}$) au lieu de $\to_\Delta$ (respectivement $\xrightarrow{*}_\Delta$) et nous omettons l'indice $\mathcal{A}$ lorsqu'il n'y a pas d'ambiguïté possible sur l'automate utilisé.

Le langage d'arbres $\mathcal{L}(\mathcal{A})$ reconnu par $\mathcal{A}$ est l'ensemble des termes clos acceptés par $\mathcal{A}$. Un langage d'arbres $\mathcal{L}$ sur Σ est dit *reconnaissable* si $\mathcal{L} = \mathcal{L}(\mathcal{A})$ pour un automate d'arbres $\mathcal{A}$. Tout langage reconnaissable par AAS est dit *régulier* et la classe des langages réguliers est notée REG. Deux automates d'arbres sont *équivalents* s'ils reconnaissent le même langage d'arbres.

Pour tout état q de $\mathcal{Q}$, on note $\mathcal{L}(\mathcal{A}, q)$ l'ensemble des termes clos t de $T(\Sigma)$ tels que $t \xrightarrow{*}_{\mathcal{A}} q(t)$. Un état q est dit *accessible* si $\mathcal{L}(\mathcal{A}, q) \neq \emptyset$.

Exemple 33. *Soient la signature* $\Sigma = \{f/2, g/1, a/0\}$ *et l'AAS* $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ *avec* $\mathcal{Q} = \{q_a, q_g, q_f\}$, $\mathcal{Q}_f = \{q_f\}$ *et* Δ *constitué des règles suivantes :*

$$\begin{array}{rcl@{\qquad}rcl} a & \to & q_a(a) & g(q_a(x)) & \to & q_g(g(x)) \\ g(q_g(x)) & \to & q_g(g(x)) & f(q_g(x), q_g(y)) & \to & q_f(f(x,y)) \end{array}$$

Alors

$$\begin{array}{rcl@{\qquad}rcl} & & f(g(a), g(a)) & \xrightarrow{*}_{\mathcal{A}} & f(g(q_a(a)), g(q_a(a))) \\ \xrightarrow{*}_{\mathcal{A}} & f(q_g(g(a)), q_g(g(a))) & \to_{\mathcal{A}} & & q_f(f(g(a), g(a)) \end{array}$$

Donc le terme $f(g(a), g(a))$ *est accepté par l'automate* $\mathcal{A}$.

Les langages finis ont la particularité d'être tous réguliers. En effet pour chaque terme t d'un langage fini, on définit un état pour chaque position de t. Puis on prend pour règles celles permettant de reconstruire le terme t à partir de ses états. On obtient un nombre d'états fini puisque le nombre de termes dans le langage est fini.

Propriété 34. *Tout langage d'arbres fini est régulier.*

Exemple 35. *Soient la signature* $\Sigma = \{f/2, g/1, a/0\}$, *et les termes* $s = f(a, g(a))$ *et* $t = g(a)$.

Pour le terme s, *on considère l'ensemble d'états* $\mathcal{Q}_s = \{q_s^\epsilon, q_s^1, q_s^2, q_s^{21}\}$ *et l'ensemble de règles* Δ_s *:*

$$\begin{array}{rcl} a & \to & q_s^1(a) \\ a & \to & q_s^{21}(a) \\ g(q_s^{21}(x)) & \to & q_s^2(g(x)) \\ f(q_s^1(x_1), q_s^2(x_2)) & \to & q_s^\epsilon(f(x_1, x_2)) \end{array}$$

Et pour le terme t, *on considère l'ensemble d'états* $\mathcal{Q}_t = \{q_t^\epsilon, q_t^1\}$ *et l'ensemble de règles* Δ_t *:*

$$\begin{array}{rcl} a & \to & q_t^1(a) \\ g(q_t^1(x)) & \to & q_t^\epsilon(g(x)) \end{array}$$

Alors l'automate $\mathcal{A} = (\Sigma, \mathcal{Q}_s \cup \mathcal{Q}_t, \{q_s^\epsilon, q_t^\epsilon\}, \Delta_s \cup \Delta_t)$ *reconnaît le langage* $\{s, t\}$.

Considérant une dérivation $t \xrightarrow{*}_{\mathcal{A}} q(t)$, avec t terme clos et q état, il est parfois utile de se souvenir de l'historique de cette dérivation, c'est-à-dire de se rappeler dans quels états sont dérivés les sous-termes de t. Nous considérons pour cela les définitions suivantes.

Soient $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ un *AAS* et t un terme clos de $T(\Sigma)$. Un *calcul* r de $\mathcal{A}$ sur t est une application r de $\mathcal{P}os(t)$ dans $\mathcal{Q}$ *compatible avec les règles* de Δ, c'est-à-dire pour toute position p de $\mathcal{P}os(t)$, si $t(p) = f$ symbole de Σ_n, $r(p) = q$, $r(pi) = q_i$ pour tout i de $[n]$, alors $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ est une règle de Δ. Un calcul de $\mathcal{A}$ est *réussi* si $r(\epsilon)$ est un état final et un terme t clos est accepté par $\mathcal{A}$ si et seulement s'il existe un calcul réussi de $\mathcal{A}$ sur t.

Comme dans le cas des automates de mots, il est parfois commode dans la dérivation d'un terme de modifier uniquement l'état courant. Ceci est réalisé en ajoutant un nouveau type de règles à l'ensemble des règles de transition d'un automate. Un *AAS* avec *ϵ-transitions* est un automate d'arbres dont l'ensemble des règles de transition est constitué de règles de la forme habituelle, c'est-à-dire $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$, et de règles de la forme $q(x) \to q'(x)$ (ϵ-transitions) où q et q' sont des états.

Les ϵ-transitions ne permettent pas d'élargir la classe des langages reconnus par les automates d'arbres finis (théorème 36) mais elles sont utiles dans certaines constructions et simplifient certaines preuves sur les automates.

Théorème 36. *Si $\mathcal{L}$ est un langage reconnu par un AAS avec ϵ-transitions, alors $\mathcal{L}$ est reconnu par un AAS sans ϵ-transition.*

Nous donnons l'idée de la construction de l'automate sans ϵ-transition puisque cette construction est utilisée pour montrer que l'image d'un régulier par un morphisme d'arbres semi-linéaire est reconnaissable par un automate à tests d'égalité entre frères (propriété 137).

Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ un *AAS* avec ϵ-transitions. Nous considérons Δ^ϵ l'ensemble des ϵ-transitions de Δ et nous notons pour tout état q, *ϵ-clôture*(q) l'ensemble des états q' tels qu'il existe des états $q_1, \ldots, q_n$ satisfaisant :

$$\begin{array}{rcll} \forall i \in [n-1],\ q_i(x) & \to & q_{i+1}(x) & \in \Delta \\ q & = & q_1 & \\ q' & = & q_n & \end{array}$$

Le calcul d'un tel ensemble est équivalent à savoir quels sommets peuvent être atteints à partir d'un sommet donné dans un graphe orienté. Ceci peut être fait en temps quadratique. Maintenant considérons l'automate d'arbres $\mathcal{A}' = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta')$ où Δ' est défini par :

$$\Big(f(q_1(x_1), \ldots, q_n(x_n)) \to q'(f(x_1, \ldots, x_n)) \in \Delta'\Big)$$

$$\Leftrightarrow$$

$$\Big(f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n)) \in \Delta \text{ et } q' \in \textit{ϵ-clôture}(q)\Big)$$

L'équivalence des deux automates se montrent par induction sur la longueur des dérivations.

1.2 Déterminisation et complétion

Notre définition des automates d'arbres correspond à la notion d'automates d'arbres finis non déterministes, au sens où pour un automate $\mathcal{A}$, il peut exister plusieurs dérivations possibles pour un terme clos donné. Il est possible de définir la notion de déterminisme soit en considérant les règles des automates, soit en considérant les calculs des automates. Pour cet ouvrage, nous avons décidé d'utiliser la première solution.

Un *AAS*, d'ensemble de règles Δ, est *déterministe* si et seulement si Δ ne contient pas d'ϵ-transition et deux règles de Δ ayant le même membre gauche modulo renommage de variables ont le même membre droit modulo renommage de variables. Donc si un automate est déterministe, il existe au plus un calcul pour tout terme clos, c'est-à-dire pour tout terme t clos, il existe au plus un état q tel que $t \xrightarrow{*} q(t)$. Remarquons que cette propriété n'est pas suffisante pour dire qu'un automate est déterministe (exemple 37).

Exemple 37. *Soient la signature* $\Sigma = \{g/1, a/0\}$ *et l'AAS* $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ *avec* $\mathcal{Q} = \{q_0, q_1, q\}$, $\mathcal{Q}_f = \{q_0\}$ *et* Δ *constitué des règles suivantes:*

$$\begin{array}{rcl@{\qquad}rcl} a & \to & q_0(a) & g(q_0(x)) & \to & q_1(g(x)) \\ g(q_1(x)) & \to & q_0(g(x)) & g(q(x)) & \to & q_0(g(x)) \\ g(q(x)) & \to & q_1(g(x)) & & & \end{array}$$

$\mathcal{A}$ n'est pas déterministe puisqu'il existe deux règles de membre gauche $g(q(x))$ et de membres droits différents, pourtant il existe au plus un calcul sur tout terme clos de $T(\Sigma)$ puisque l'état q est inaccessible.

Il est aussi intéressant de considérer des automates pour lesquels il existe au moins un calcul sur chaque terme clos, d'où la notion de complétude. Un automate d'arbres $\mathcal{A}$ est *complet* s'il existe au moins une règle $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ dans Δ pour tout symbole de fonction f de Σ_n et tous états $q_1, \ldots, q_n$. Donc pour un automate complet, il existe pour tout terme t clos au moins un état q tel que $t \xrightarrow{*} q(t)$. Remarquons que pour tout automate déterministe et complet, il existe exactement un calcul sur tout terme clos.

Les notions de déterminisme et de complétude ne modifient en rien la classe des réguliers (théorèmes 38 et 39) mais elles sont utiles pour simplifier certaines preuves de la théorie des langages d'arbres.

Théorème 38. *Pour tout automate d'arbres standard $\mathcal{A}$, il existe un automate d'arbres standard $\mathcal{A}'$ déterministe tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Théorème 39. *Pour tout automate d'arbres standard $\mathcal{A}$, il existe un automate d'arbres standard $\mathcal{A}'$ complet tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

De plus lorsque l'algorithme de complétion habituel (que l'on trouve en particulier dans [20]) est utilisé, si $\mathcal{A}$ est déterministe, $\mathcal{A}'$ l'est aussi. Cette remarque et les deux théorèmes précédents, nous permettent d'énoncer le théorème suivant.

Théorème 40. *Pour tout automate d'arbres standard $\mathcal{A}$, il existe un automate d'arbres standard $\mathcal{A}'$ déterministe et complet tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Exemple 41. *Soient la signature* $\Sigma = \{f/2, g/1, a/0\}$ *et l'AAS* $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ *avec* $\mathcal{Q} = \{q_a, q_g, q_f\}$, $\mathcal{Q}_f = \{q_f\}$ *et* Δ *constitué des règles suivantes :*

$$\begin{array}{rcl@{\qquad}rcl} a & \to & q_a(a) & g(q_a(x)) & \to & q_g(g(x)) \\ g(q_g(x)) & \to & q_g(g(x)) & f(q_g(x), q_g(y)) & \to & q_f(f(x,y)) \end{array}$$

$\mathcal{A}$ est déterministe mais pas complet, puisque par exemple il n'y a pas de règle de transition de membre gauche $f(q_a(x), q_a(y))$. Il est facile de définir un automate déterministe et complet $\mathcal{A}'$ reconnaissant le même langage que $\mathcal{A}$. Pour cela, on ajoute un état q_p dit état puits. L'automate $\mathcal{A}' = (\Sigma, \mathcal{Q}', \mathcal{Q}_f, \Delta')$ est obtenu en ajoutant q_p à l'ensemble des états ($\mathcal{Q}' = \mathcal{Q} \cup \{q_p\}$) et les règles suivantes à Δ :

$$\begin{array}{rcl@{\qquad}rcl} g(q_f(x)) & \to & q_p(g(x)) & g(q_p(x)) & \to & q_p(g(x)) \\ f(q_a(x), q_a(y)) & \to & q_p(f(x,y)) & f(q_a(x), q_g(y)) & \to & q_p(f(x,y)) \\ & \cdots & & f(q_p(x), q_p(y)) & \to & q_p(f(x,y)) \end{array}$$

Pour des raisons pratiques, il est parfois utile de considérer des automates dans lesquels les états inutiles (états inaccessibles) sont éliminés. Soit $\mathcal{A}$ un *AAS*. L'automate $\mathcal{A}$ est *réduit* si tous ses états sont accessibles (l'automate $\mathcal{A}$ de l'exemple 41 est réduit).

Théorème 42. *Pour tout automate d'arbres standard $\mathcal{A}$, il existe un automate d'arbres standard $\mathcal{A}'$ réduit tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

1.3 Minimisation

Nous rappelons dans cette section le théorème de Myhill-Nerode permettant de montrer que pour tout automate, il existe un unique automate déterministe de taille minimum en nombre d'états qui lui est équivalent. Nous donnons ici les notions principales nécessaires à la démonstration de ce résultat car nous les utilisons pour la résolution du problème de reconnaissabilité des langages reconnus par automates d'arbres à tests entre frères (chapitre **Reconnaissabilité** de la partie **Automates d'arbres à contraintes**).

Avant d'énoncer le théorème de Myhill-Nerode, nous définissons la notion de congruence. Soit Σ une signature. Une relation d'équivalence $\equiv$ sur $T(\Sigma)$ est une *congruence* sur $T(\Sigma)$ si pour tout symbole de fonction f de Σ d'arité quelconque n et tous termes $s_1, \ldots, s_n, t_1, \ldots, t_n$ de $T(\Sigma)$:

$$(\forall i \in [n], s_i \equiv t_i) \Rightarrow f(t_1, \ldots, t_n) \equiv f(t_1, \ldots, t_n).$$

De façon équivalente, une congruence est une relation d'équivalence close par contexte (pour tout contexte de $\mathcal{C}(\Sigma)$, si $s \equiv t$ alors $C[s] \equiv C[t]$). Une congruence est dite d'*indice fini* si le nombre de classes d'équivalence induites par $\equiv$ est fini.

Pour un langage d'arbres $\mathcal{L}$ donné, soit la congruence $\equiv_{\mathcal{L}}$ sur $T(\Sigma)$ définie par : pour tout terme s et t de $T(\Sigma)$, $s \equiv_{\mathcal{L}} t$ si pour tout contexte C de $\mathcal{C}(\Sigma)$,

$$C[s] \in \mathcal{L} \Leftrightarrow C[t] \in \mathcal{L}.$$

Théorème 43 (Myhill-Nerode). *Les trois affirmations suivantes sont équivalentes :*

1. *$\mathcal{L}$ est régulier.*

2. *$\mathcal{L}$ est l'union de classes d'équivalence d'une congruence d'indice fini.*

3. *La relation $\equiv_{\mathcal{L}}$ est une congruence d'indice fini.*

La preuve de ce théorème n'est pas donnée ici mais nous définissons l'automate $\mathcal{A}_{min}$ construit pour la preuve de l'assertion $3 \Rightarrow 1$ dont nous nous inspirons pour la résolution du problème de reconnaissabilité des langages reconnus par automates d'arbres à tests entre frères. Supposons que la relation $\equiv_{\mathcal{L}}$ soit une congruence d'index fini. Nous définissons alors l'automate $\mathcal{A}_{min} = (\Sigma, \mathcal{Q}_{min}, \mathcal{Q}_{min_f}, \Delta_{min})$ avec $\mathcal{Q}_{min}$ l'ensemble des classes d'équivalence de la relation $\equiv_{\mathcal{L}}$, $\mathcal{Q}_{min_f} = \{[t]_{\equiv_{\mathcal{L}}} \mid t \in \mathcal{L}\}$ et Δ_{min} constitué des règles suivantes :

$$f\Big([t_1]_{\equiv_{\mathcal{L}}}(x_1), \ldots, [t_n]_{\equiv_{\mathcal{L}}}(x_n)\Big) \to [f(t_1, \ldots, t_n)]_{\equiv_{\mathcal{L}}}\Big(f(x_1, \ldots, x_n)\Big)$$

pour tout symbole f d'arité n quelconque et tous termes $t_1, \ldots, t_n$ de $T(\Sigma)$. Nous pouvons montrer que $\mathcal{L}(\mathcal{A}_{min}) = \mathcal{L}$.

Nous déduisons du théorème de Myhill-Nerode, la propriété suivante :

Propriété 44. *Pour tout régulier $\mathcal{L}$, l'automate $\mathcal{A}_{min}$ défini par la construction ci-dessus est le plus petit AAS déterministe en nombre d'états reconnaissant $\mathcal{L}$, et $\mathcal{A}_{min}$ est unique modulo renommage des états.*

1.4 Propriétés de clôture

Dans cette section, nous nous intéressons aux propriétés de clôture de la classe des langages réguliers.

Tout d'abord, la classe *REG* possède d'excellentes propriétés de clôtures booléennes car elle est close par toutes les opérations booléennes ensemblistes.

Propriété 45 (Opérations booléennes). *La classe des réguliers est close par opérations booléennes (union, intersection et complément).*

Pour les propriétés de clôtures par morphismes d'arbres, la classe des réguliers se distingue du cas des mots. En effet, la classe des reconnaissables par automates de mots est close par morphismes et morphismes inverses de mots, par contre la classe des réguliers n'est pas close par morphismes car elle n'est pas close par morphismes d'arbres non linéaires (à cause de la non-linéarité des termes images). Voici ci-dessous les principales propriétés de clôtures par morphismes d'arbres de la classe des réguliers.

Propriété 46 (Morphismes non linéaires). *La classe des réguliers n'est pas close par morphismes d'arbres non linéaires.*

Propriété 47 (Morphismes linéaires et morphismes inverses). *La classe des réguliers est close par morphismes d'arbres linéaires et par morphismes d'arbres inverses.*

1.5 Problèmes de décision

Dans cette section, nous rappelons quelques problèmes de décision des automates d'arbres et nous donnons leur solution dans le cas des automates d'arbres standards.

Problème d'appartenance

Problème d'appartenance
Entrées : *Un automate d'arbres $\mathcal{A}$ et un terme t clos.*
Question : *Est-ce que t appartient à $\mathcal{L}(\mathcal{A})$?*

Théorème 48. *Le problème d'appartenance est décidable pour les AAS.*

Problème du vide

Problème du vide
Entrée : *Un automate d'arbres $\mathcal{A}$.*
Question : *Est-ce que $\mathcal{L}(\mathcal{A}) = \emptyset$?*

Théorème 49. *Le problème du vide est décidable pour les AAS.*

Problème de finitude

Problème de finitude
Entrée : *Un automate d'arbres $\mathcal{A}$.*
Question : *Est-ce que $\mathcal{L}(\mathcal{A})$ est fini ?*

Théorème 50. *Le problème de finitude est décidable pour les AAS.*

Problème du plein

Problème du plein
Entrée : *Un automate d'arbres $\mathcal{A}$ sur la signature Σ.*
Question : *Est-ce que $\mathcal{L}(\mathcal{A}) = \mathcal{T}(\Sigma)$?*

Théorème 51. *Le problème du plein est décidable pour les AAS.*

Problème d'inclusion

Problème d'inclusion
Entrées : *Deux automates d'arbres $\mathcal{A}_1$ et $\mathcal{A}_2$.*
Question : *Est-ce que $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_1)$?*

Théorème 52. *Le problème d'inclusion est décidable pour les AAS.*

Problème d'équivalence

Problème d'équivalence
Entrées : *Deux automates d'arbres $\mathcal{A}_1$ et $\mathcal{A}_2$.*
Question : *Est-ce que $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_1)$?*

Théorème 53. *Le problème d'équivalence est décidable pour les AAS.*

1.6 Applications

Dans cette dernière section, nous citons quelques applications des automates d'arbres. Nous considérons une signature Σ et un ensemble dénombrable de variables $\mathcal{X}$.

1.6.1 Reconnaissabilité des instances closes

La linéarité d'un terme permet de construire un automate d'arbres standard reconnaissant l'ensemble de ses instances closes (propriété 54 et exemple 55).

Propriété 54 (*AAS* reconnaissabilité des instances closes). *Si t est un contexte de $T(\Sigma, \mathcal{X})$, alors l'ensemble des instances closes de t est un régulier.*

Exemple 55. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$, le terme linéaire $t = f(x, g(y))$ et l'automate d'arbres standard $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où $\mathcal{Q} = \{q, q_g, q_f\}$, $\mathcal{Q}_f = \{q_f\}$ et Δ est constitué des règles suivantes:*

$$\begin{array}{rcl@{\qquad}rcl}
a & \to & q(a) & f(q(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\
g(q(x_1)) & \to & q_g(g(x_1)) & f(q_g(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\
g(q_g(x_1)) & \to & q_g(g(x_1)) & f(q(x_1), q_g(x_2)) & \to & q_f(f(x_1, x_2)) \\
f(q_g(x_1), q_g(x_2)) & \to & q_f(f(x_1, x_2)) & & &
\end{array}$$

L'automate $\mathcal{A}$ reconnaît l'ensemble des instances closes de t.

1.6.2 Reconnaissabilité des termes clos réductibles et des formes normales

Soit $\mathcal{R}$ un système de réécriture linéaire à gauche. Nous montrons que l'ensemble des termes clos réductibles par $\mathcal{R}$ et l'ensemble des termes clos en forme normale pour $\mathcal{R}$ sont réguliers. Nous définissons tout d'abord la notion d'entourage.

Définition 56 (Entourage). *Soient t un terme de $T(\Sigma, \mathcal{X})$ et u un terme de $T(\Sigma)$. On dit que u* entoure *t s'il existe une substitution σ telle que $t\sigma$ soit un sous-terme de u.*

L'entourage joue un rôle important en réécriture. En effet, un terme s est réductible par un système de réécriture $\mathcal{R}$ si et seulement si s entoure au moins une partie gauche de règle.

La relation avec les automates d'arbres est donnée par la propriété suivante :

Propriété 57. *Si t est un contexte de $T(\Sigma, \mathcal{X})$, alors l'ensemble des termes entourant t est régulier.*

Exemple 58. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$, le contexte $t = f(x, g(y))$ et l'automate d'arbres standard $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où $\mathcal{Q} = \{q, q_{g(y)}, q_{f(x,g(y))}\}$, $\mathcal{Q}_f = \{q_{f(x,g(y))}\}$ et Δ est constitué des règles suivantes:*

$$\begin{array}{rcl@{\qquad}rcl}
a & \to & q(a) & f(q(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\
g(q(x_1)) & \to & q(g(x_1)) & f(q(x_1), q_{g(y)}(x_2)) & \to & q_{f(x,g(y))}(f(x_1, x_2)) \\
g(q(x_1)) & \to & q_{g(y)}(g(x_1)) & f(q_{f(x,g(y))}(x_1), q(x_2)) & \to & q_{f(x,g(y))}(f(x_1, x_2)) \\
g(q_{f(x,g(y))}(x_1)) & \to & q_{f(x,g(y))}(g(x_1)) & f(q(x_1), q_{f(x,g(y))}(x_2)) & \to & q_{f(x,g(y))}(f(x_1, x_2))
\end{array}$$

L'automate $\mathcal{A}$ reconnaît l'ensemble des termes entourant t.

On déduit directement de la propriété précédente que :

Propriété 59 (*AAS* reconnaissabilité des termes clos réductibles). *Soit $\mathcal{R}$ un système de réécriture linéaire à gauche. Alors l'ensemble des termes clos réductibles par $\mathcal{R}$ est régulier.*

Par la propriété précédente, l'ensemble des termes clos réductibles par $\mathcal{R}$ est reconnaissable par un automate d'arbres standard $\mathcal{A}$. Le complément de $\mathcal{L}(\mathcal{A})$ est l'ensemble des termes clos en forme normale pour $\mathcal{R}$. Cet ensemble est donc régulier d'après la clôture par complément de la classe des réguliers (propriété 45).

Propriété 60 (*AAS* reconnaissabilité des formes normales closes). *Soit $\mathcal{R}$ un système de réécriture linéaire à gauche. Alors l'ensemble des termes clos en forme normale pour $\mathcal{R}$ est régulier.*

1.6.3 Théorie de la réduction

Nous définissons une théorie, appelée théorie de la réduction, en associant à tout terme t de $T(\Sigma, \mathcal{X})$ un prédicat unaire $red_t(.)$ capturant l'ensemble des termes clos entourant t.

La *théorie de la réduction* est l'ensemble des formules logiques du premier ordre construites sur la signature Σ, l'ensemble de variables $\mathcal{X}$ et l'ensemble de prédicats $\mathcal{P} = \{red_t(.) \mid t \in T(\Sigma, \mathcal{X})\}$. La structure, notée $\mathcal{R}ed_\Sigma$, de la théorie de la réduction que nous considérons ici est définie comme suit :

- Le support est l'ensemble des termes construits sur Σ c'est-à-dire $T(\Sigma)$,
- L'élément (respectivement la fonction) associé à toute constante e (respectivement tout symbole de fonction d'arité au moins un) est canonique, dans le sens où pour toute constante e de Σ (respectivement pour tout symbole de fonction f d'arité n strictement positive), $e^{\mathcal{R}ed_\Sigma} = e$ (respectivement $f^{\mathcal{R}ed_\Sigma}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ pour tous termes $t_1, \ldots, t_n$ de $T(\Sigma)$).
- Pour tout terme t de $T(\Sigma, \mathcal{X})$, La relation unaire associée au symbole de prédicat $red_t(.)$ est l'ensemble des termes clos entourant t.

Donc pour tout terme t de $T(\Sigma, \mathcal{X})$ et tout terme clos u, on a $\mathcal{R}ed_\Sigma \models red_t(u)$ si et seulement si u entoure t.

Donc par la propriété 57, pour tout terme t linéaire, il existe un *AAS* $\mathcal{A}_t$ tel que $\mathcal{L}(\mathcal{A})_t = \{u \in T(\Sigma) \mid \mathcal{R}ed_\Sigma \models red_t(u)\}$. Nous déduisons alors le théorème suivant de la décidabilité du problème du vide pour les *AAS* (théorème 49), de la clôture par opérations booléennes de la classe des réguliers (propriété 45), ainsi que d'autres propriétés non énocées dans ce document (clôture par cylindrification et projection des langages de tuples reconnaissables par automates d'arbres).

Théorème 61. *Le problème de satisfiabilité de la théorie de la réduction restreinte aux prédicats $red_t(.)$ où t est un terme linéaire est décidable.*

1.6.4 Réductibilité inductive

La réductibilité inductive est l'une des seules propriétés de systèmes de réécriture décidable en général. La preuve de cette propriété est due à D. Plaisted [70].

Dans cette section, nous définissons le problème de réductibilité inductive et nous citons une instance particulière de ce problème dont la preuve de décidabilité utilise les *AAS* (théorème 65).

Définition 62 (Réductibilité inductive). *Soient $\mathcal{R}$ un système de réécriture et t un terme de $T(\Sigma, \mathcal{X})$. On dit que t est* inductivement réductible *pour $\mathcal{R}$ si toutes les instances closes de t sont réductibles par $\mathcal{R}$.*

Exemple 63. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$ et le système de réécriture $\mathcal{R} := \{f(x, f(y, x)) \to g(x)\}$.*

Alors le terme $f(x, f(y, x))$ est de façon évidente inductivement réductible par $\mathcal{R}$. Par contre le terme $g(x)$ ne l'est pas puisque $g(a)$ est une instance close de $g(x)$ non réductible par $\mathcal{R}$.

Nous définissons ci-dessous le problème de réductibilité inductive.

Problème de réductibilité inductive
Entrées : *Σ une signature, $\mathcal{X}$ un ensemble de variables, t un terme de $T(\Sigma, \mathcal{X})$ et $\mathcal{R}$ un système de réécriture sur Σ.*
Question : *Est-ce que t est inductivement réductible pour $\mathcal{R}$?*

Ce problème est montré décidable pour les systèmes linéaires à gauche en utilisant les *AAS* (théorème 65). L'ensemble des instances closes de t n'est pas un régulier en général puisque t n'est pas forcément linéaire. Il en est de même pour l'ensembles des instances closes de t en forme normale pour $\mathcal{R}$ mais on peut montrer la propriété suivante :

Propriété 64. *Soient $\mathcal{R}$ un système de réécriture linéaire à gauche sur Σ et t un terme de $T(\Sigma, \mathcal{X})$. Il existe un AAS $\mathcal{A}_t$ tel que $\mathcal{L}(\mathcal{A}_t) = \emptyset$ si et seulement si t est inductivement réductible pour $\mathcal{R}$.*

On déduit alors de la décidabilité du problème du vide pour les *AAS*, le théorème suivant :

Théorème 65 (Problème de réductibilité inductive). *Le problème de réductibilité inductive est décidable pour les systèmes linéaires à gauche.*

1.6.5 Vérification de sortes

La dernière application des automates d'arbres standards à laquelle nous nous intéressons est la vérification de sortes. Avant d'aller plus loin, nous fixons les notions de signature multi-sortée et de terme bien sorté.

Les systèmes de réécriture sont un outil privilégié de preuve dans les types abstraits algébriques. Dans ce cadre, il est naturel d'assigner aux termes des types, ou sortes. Les

symboles de fonction se voient alors associer non seulement une arité mais aussi un type et un domaine de définition, on parle alors de signature multi-sortée.

Formellement, une *signature multi-sortée* est une paire $(\mathcal{S}, \Sigma)$ où $\mathcal{S}$ est un ensemble fini de *sortes* et Σ une signature finie de symboles de fonction, avec pour chaque symbole f de Σ, une arité n et un *profil* c'est-à-dire un tuple de $n+1$ sortes que l'on écrit sous la forme :

$$f : s_1 \times \cdots \times s_n \to s.$$

Le tuple de sortes $(s_1, \ldots, s_n)$ ($n = 0$ pour une constante) est le tuple des sortes des arguments de f et la sorte s est la sorte de f.

Exemple 66. *Considérons l'exemple classique de la signature multi-sortée des piles d'entiers (N. Dershowitz et J.P. Jouannaud [26]). Cette signature utilise deux sortes : Nat et Pile. Dans la signature Σ de cette signature multi-sortée, on distingue un sous-ensemble C de symboles de constructeurs, qui sont les symboles permettant de construire des entiers et des piles d'entiers :*

$$\begin{aligned} 0 &: && Nat \\ s &: && Nat && \to && Nat \\ \wedge &: && Pile \\ push &: && Nat \times Pile && \to && Pile \end{aligned}$$

Une pile d'entiers $\begin{bmatrix} i_n \\ \vdots \\ i_1 \\ i_0 \end{bmatrix}$ *est alors représentée par le terme suivant construit sur l'ensemble C des constructeurs :*

$$push(\underbrace{s(\ldots(s(0))\ldots)}_{i_n}, push(\ldots, push(\underbrace{s(\ldots(s(0))\ldots)}_{i_0}, \wedge)\ldots))$$

Les autres symboles de Σ sont des symboles de fonction pour manipuler les piles d'entiers. Ces fonctions sont : top(p) qui retourne l'entier en tête d'une pile p, pop(p) qui retourne le reste de p et alternate(p) qui combine deux piles p_1 et p_2.

$$\begin{aligned} top &: && Pile && \to && Nat \\ pop &: && Pile && \to && Pile \\ alternate &: && Pile \times Pile && \to && Pile \end{aligned}$$

On considère maintenant une signature multi-sortée $(\mathcal{S}, \Sigma)$ et un ensemble $\mathcal{X}$ de variables partitionné en ensembles infinis dénombrables de variables sortées :

$$\mathcal{X} = \bigcup_{s \in \mathcal{S}} \mathcal{X}_s.$$

Pour vérifier qu'un terme de $T(\Sigma, \mathcal{X})$ est bien formé, il faut non seulement vérifier que le nombre d'arguments de chaque symbole de fonction est correct (suivant son arité) mais de plus que les sortes des symboles sont respectées.

Formellement, un terme t de $T(\Sigma, \mathcal{X})$ est *bien sorté* si et seulement si :

- Ou bien t une variable de $\mathcal{X}_s$, s étant une sorte, auquel cas la sorte de t est s,

- Ou bien t est une constante a de Σ de profil $a : s$ et la sorte de t est s,
- Ou bien $t = f(t_1, \ldots, t_n)$, le profil de f est $s_1 \times \cdots \times s_n \rightarrow s$ et les termes $t_1, \ldots, t_n$ sont bien sortés de sortes respectives $s_1, \ldots, s_n$. La sorte de t est alors s.

Exemple 67. *Nous considérons la signature multi-sortée pour les piles d'entiers définie dans l'exemple 66. Alors le terme*

$$alternate(push(s(0), \wedge), pop(push(0, \wedge)))$$

est bien sorté mais pas le terme $s(\wedge)$.

Il a été montré que l'ensemble des termes clos de $T(\Sigma)$ bien sortés suivant la signature $(\mathcal{S}, \Sigma)$ est un régulier.

Propriété 68. *Soit une sorte s de $\mathcal{S}$. Il existe un AAS $\mathcal{A}_s$ reconnaissant l'ensemble des termes clos de sorte s.*

La construction de l'automate $\mathcal{A}_s$ est simple puisqu'il suffit de considérer pour ensemble d'états l'ensemble de sorte $\mathcal{S}$, pour seul état final s et de construire les règles de transition comme des déclarations des profils des symboles de Σ.

Exemple 69. *Nous considérons toujours la signature multi-sortée pour les piles d'entiers définie dans l'exemple 66. Soit l'AAS* $\mathcal{A}_{Pile} = (\Sigma, \mathcal{S}, \{Pile\}, \Delta)$ *dont les règles sont:*

$$\begin{aligned}
0 &\rightarrow Nat(0) \\
s(Nat(x_1)) &\rightarrow Nat(s(x_1)) \\
\wedge &\rightarrow Pile(\wedge) \\
push(Nat(x_1), Pile(x_2)) &\rightarrow Pile(push(x_1, x_2)) \\
top(Pile(x_1)) &\rightarrow Nat(top(x_1)) \\
pop(Pile(x_1)) &\rightarrow Pile(pop(x_1)) \\
alternate(Pile(x_1), Pile(x_2)) &\rightarrow Pile(alternate(x_1, x_2))
\end{aligned}$$

L'automate $\mathcal{A}_{Pile}$ reconnaît l'ensemble des termes clos de sorte Pile.

La propriété précédente permet d'étendre le résultat de décidabilité pour la théorie de la réduction (théorème 61) à un fragment de la théorie de la réduction avec sortes.

La *théorie de la réduction avec sortes*, notée $\mathcal{R}edgen_\Sigma$, est l'extension de le théorie de la réduction obtenue en considérant pour toute sorte s de $\mathcal{S}$ un prédicat supplémentaire unaire, notée $\cdot \in s$, dont l'interprétation est l'ensemble des termes clos de sorte s. Donc pour toute sorte s de $\mathcal{S}$ et tout terme clos u, on a $\mathcal{R}edgen_\Sigma \models u \in s$ si et seulement si u est de sorte s. On déduit alors de la propriété 68 et des propriétés des AAS (propriété 45 et théorème 49) le théorème suivant :

Théorème 70. *Le problème de satisfiabilité du fragment de la théorie de la réduction avec sortes dans lequel les prédicats $red_t(.)$ sont tels que t soit un terme linéaire est décidable.*

Chapitre 2

Automates d'arbres à contraintes

Les automates d'arbres standards ne peuvent être utilisés pour la résolution de problèmes non linéaires que dans certains cas très restreints. Par exemple, le langage $\{f(t,t) \mid t \in T(\Sigma)\}$, Σ étant une signature contenant le symbole binaire f, n'est pas régulier. En effet, les deux fils de la racine sont reconnus indépendamment et seules des informations de taille finie peuvent être remontées jusqu'à la racine, tandis que t peut être de taille arbitrairement grande.

Pour pouvoir utiliser des techniques d'automates d'arbres sur des problèmes concernant des systèmes non linéaires, des auteurs tels que M. Dauchet et J. Mongy [65], B. Bogaert et S. Tison [9, 10, 11], A.C. Caron [14], M. Dauchet *et al.* [15, 16] ont introduit des classes d'automates d'arbres étendus, les *automates à contraintes*, dont les transitions sont contraintes par des tests d'égalité et de différence. Le pouvoir d'expression de ces automates est strictement supérieur au pouvoir d'expression des automates d'arbres standards mais malheureusement certaines bonnes propriétés de ces derniers automates sont perdues. En particulier, le problème du vide est indécidable pour la classe la plus générale d'automates à contraintes. Malgré tout, certaines sous-classes d'automates à contraintes pour lesquelles le vide est décidable ont été définies.

Dans la suite de ce chapitre, nous définissons certaines classes d'automates à contraintes et nous rappelons certaines propriétés et applications de celles-ci. Les éléments de ce chapitre peuvent être retrouvés dans les ouvrages traitant des automates à contraintes, et en particulier dans la thèse de F. Jacquemard [46] et le livre de H. Comon *et al.* [20].

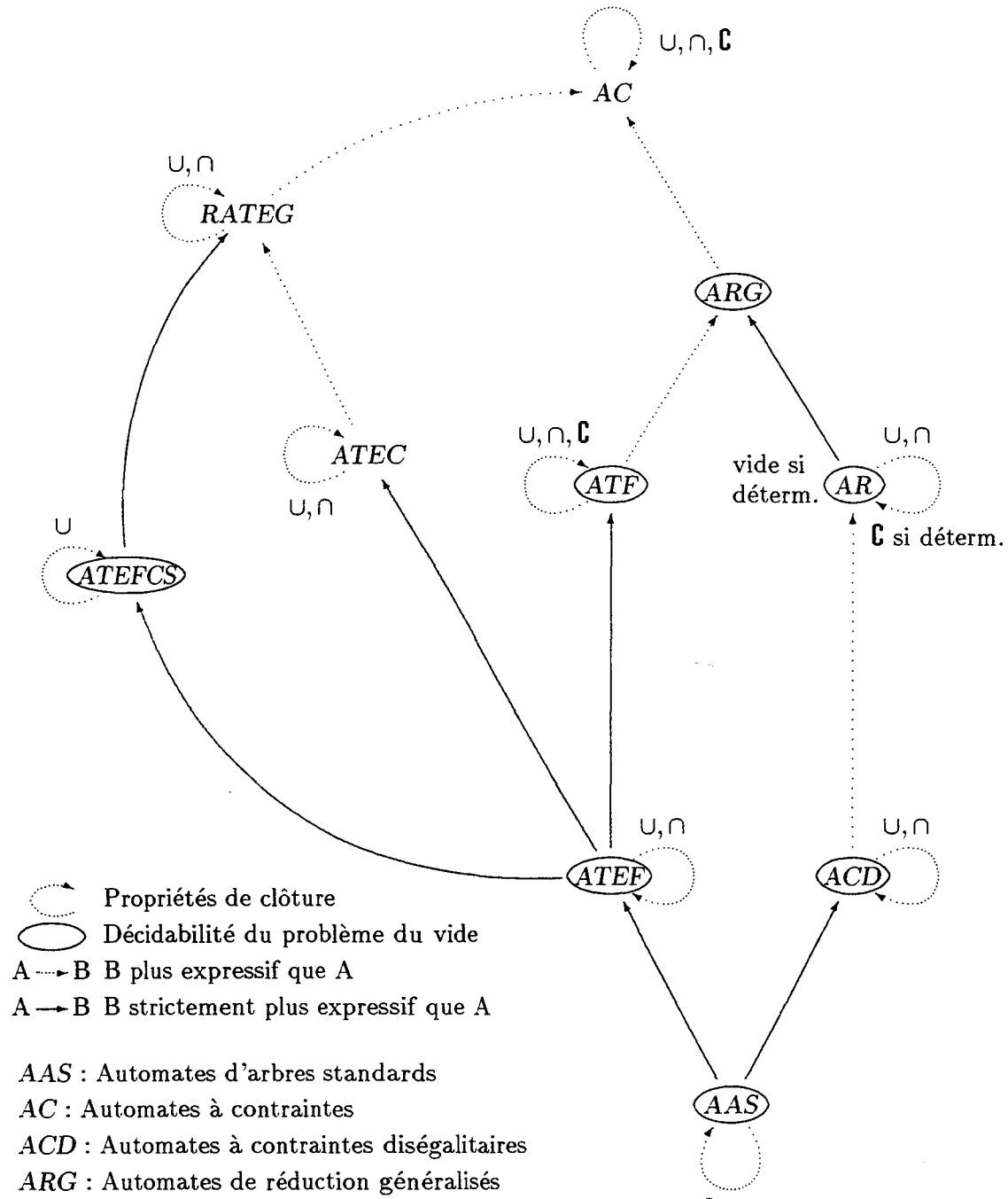
2.1 Expressivité

Avant d'aller plus loin, nous définissons la notion d'expressivité permettant de comparer les différentes classes d'automates entre elles suivant leurs langages reconnus.

Définition 71. *Soient $\mathcal{CA}_1$ et $\mathcal{CA}_2$ deux classes d'automates. Alors le* pouvoir d'expression *de $\mathcal{CA}_1$ est supérieur à celui de $\mathcal{CA}_2$ si la classe des langages reconnus par les automates de $\mathcal{CA}_2$ est incluse dans la classe des langages reconnus par les automates de $\mathcal{CA}_1$.*

Dans la figure 1, nous avons regroupé les résultats d'expressivité, de clôture et de décidabilité du problème du vide pour les différentes classes de langages d'arbres de ce document.

FIG. 1 - Propriétés des automates d'arbres.



2.2 Les *RATEG*

Ces automates ont été introduits en 1981 au Laboratoire d'Informatique de Lille par M. Dauchet et J. Mongy [65]. Le système de transition d'un tel automate est un système de réécriture d'une forme plus générale que pour les automates d'arbres standards.

2.2.1 Définition

Définition 72 (*RATEG*). *Soit $\mathcal{X}$ un ensemble dénombrable de variables. Un automate RATEG est un quadruplet $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini d'états ($\Sigma \cap \mathcal{Q} = \emptyset$);*
3. *$\mathcal{Q}_f \subseteq \mathcal{Q}$ est l'ensemble des états finaux;*
4. *Δ est un système de réécriture dont les règles ont l'une des formes suivantes :*
   (a) *$t_0 \to q(t_0)$ où t_0 est un terme de $T(\Sigma)$ et q un état de $\mathcal{Q}$;*
   (b) *$f(q_1(t_1), \ldots, q_n(t_n)) \to q(f(t_1, \ldots, t_n))$ où f est un symbole de Σ d'arité n supérieure ou égale à un, $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$ et $t_1, \ldots, t_n$ des termes de $T(\Sigma, \mathcal{X})$.*

Les termes $t_1, \ldots, t_n$ des membres gauches des règles de Δ peuvent avoir des variables communes et même être non linéaires. Cette non linéarité (exemple 77) permet de prendre en compte la non linéarité des systèmes.

La reconnaissance d'un terme t de $T(\Sigma)$ par un *RATEG* $\mathcal{A}$ est définie comme pour les automates d'arbres standards par une relation de réécriture $t \xrightarrow{*}_{\Delta} q_f(t)$ où q_f est un état final.

2.2.2 Propriétés de clôture

La classe des langages reconnus par les *RATEG* possède des propriétés intéressantes puisque :

Propriété 73 (Union, intersection). *La classe des langages reconnus par les automates RATEG est close par union et intersection.*

Propriété 74 (Morphismes alphabétiques). *La classe des langages reconnus par les automates RATEG est close par morphismes alphabétiques.*

2.2.3 Problème du vide

Contrairement aux *AAS*, le problème du vide est indécidable pour les *RATEG*. Intuitivement, cette indécidabilité vient de la possibilité de superposer des tests d'égalité lors du calcul d'un tel automate, sans pouvoir borner le nombre de ces superpositions. J. Mongy dans sa thèse [65] se sert de cette propriété pour construire un *RATEG* qui reconnaît les solutions d'un problème de correspondance de Post donné.

Propriété 75. *Le problème du vide est indécidable pour les automates RATEG.*

2.2.4 Applications

Les non linéarités autorisées par les *RATEG* permettent de généraliser certaines applications des *AAS* (section **Applications** du chapitre **Automates d'arbres finis**). Mais compte-tenu de l'indécidabilité du problème du vide pour les *RATEG*, nous ne pouvons déduire de ces généralisations des résultats décidables.

Reconnaissabilité des termes clos réductibles

La reconnaissabilité par *AAS* des termes clos réductibles pour les systèmes de réécriture linéaires à gauche (propriété 59) se généralise à l'ensemble des systèmes de réécriture.

Propriété 76 (*RATEG* reconnaissabilité des termes clos réductibles). *Soient Σ une signature, $\mathcal{X}$ un ensemble de variables et $\mathcal{R}$ un système de réécriture sur Σ. L'ensemble des termes clos réductibles par $\mathcal{R}$ est reconnaissable par RATEG.*

La construction d'un tel automate est donnée ci-dessous (exemple 77) pour un système de réécriture simple.

Exemple 77. *Nous considérons le RATEG $\mathcal{A} = (\Sigma, Q, Q_f, \Delta)$ avec $\Sigma = \{f/2, g/1, a/0\}$, $Q = \{q, q_f\}$, $Q_f = \{q_f\}$ et Δ composé des règles suivantes :*

$$\begin{array}{rcl} a & \to & q(a) \\ g(q_f(x)) & \to & q_f(g(x)) \\ g(q(x)) & \to & q(g(x)) \\ f(q(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) \end{array} \qquad \begin{array}{rcl} f(q(x_1), q(f(x_2, x_1))) & \to & q_f(f(x_1, f(x_2, x_1))) \\ f(q(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\ f(q_f(x_1), q(x_2)) & \to & q_f(f(x_1, x_2)) \end{array}$$

Cette automate reconnaît l'ensemble des termes clos réductibles par le système de réécriture $\mathcal{R} := \{f(x, f(y, x)) \to a\}$.

2.2.5 Vérification de sortes

Dans la section 1.6.5 du chapitre **Automates d'arbres finis**, nous avons vus que ceux-ci permettaient de reconnaître l'ensemble des termes clos bien sortés suivant une signature multi-sortée donnée (propriété 68). Les automates *RATEG* permettent d'étendre ce résultat aux signatures avec déclarations de sortes.

Une *signature avec déclarations de sortes* est définie par un ensemble fini de sortes $\mathcal{S}$, une signature finie de symboles de fonction Σ, un ensemble de variables $\mathcal{X}$ partitionné en ensembles infinis dénombrables de variables sortées ($\mathcal{X} = \bigcup_{s \in \mathcal{S}} \mathcal{X}_s$) et des déclarations de sortes de la forme $t \in s$ avec t terme de $T(\Sigma, \mathcal{X})$ et s sorte de $\mathcal{S}$.

Remarque 78. *De façon évidente, toute signature multi-sortée est un cas particulier de signature avec déclarations de sortes.*

Soit une signature avec déclarations de sortes et $\mathcal{R}$ le système de réécriture composé des règles $t \to s$ où $t \in s$ est une déclaration de sorte de la signature. Alors un terme clos t est de sorte s si t se dérive en s par le système $\mathcal{R}$, c'est-à-dire $t \to_{\mathcal{R}}^* s$.

Propriété 79. *Soit une sorte s de $\mathcal{S}$. Il existe un RATEG $\mathcal{A}_s$ reconnaissant l'ensemble des termes clos de sorte s.*

La construction de l'automate $\mathcal{A}_s$ est la même que dans le cas des AAS (propriété 68) puisqu'il suffit de considérer pour ensemble d'états l'ensemble de sortes $\mathcal{S}$, pour seul état final s et de construire les règles de transition comme des déclarations des profils des symboles de Σ.

2.3 Automates d'arbres à contraintes

Les *RATEG* permettent de vérifier des égalités dans les termes sur lesquels on calcule. Nous présentons maintenant les automates à contraintes qui non seulement sont capables de réaliser ce type de tests mais peuvent aussi tester des différences entre sous-termes. B. Bogaert et S. Tison ont introduit dans [10, 11] des automates d'arbres qui, pour effectuer certaines transitions à partir d'une position p dans un terme t, testent des égalités ou différences entre les fils de p. Par exemple, ils vérifient que $t_{p1} = t_{p2}$ ou bien que $t_{p2} \neq t_{p3}$. Il s'agit d'automates d'arbres dans lesquels certaines règles ont été contraintes, ce qui fait de l'ensemble des règles de transition un système de réécriture contraint. Les contraintes d'une règle de transition $f(q_1(x_1), q_2(x_2), q_3(x_3)) \rightarrow q(f(x_1, x_2, x_3))$ peuvent s'exprimer :

- Soit par des égalités et différences entre positions (par exemple $1 = 3$ pour vérifier que dans un terme t on a $t|_1 = t|_3$, ou bien $2 \neq 3$ pour vérifier que $t|_2 \neq t|_3$),
- Soit par l'utilisation d'un terme $f(q_1(x_1), q_2(x_2), q_3(x_3))$ non linéaire dans le cas où seuls des tests d'égalités sont à exprimer (par exemple $x_1 = x_3$).

Ces deux formes d'expression sont bien évidemment équivalentes lorsque seules des égalités sont à tester.

Par la suite, dans les travaux de A.-C. Caron [14], ces automates ont été généralisés en autorisant des contraintes qui testent des égalités et différences entre sous-termes situés à des profondeurs plus éloignées. Par exemple, pour effectuer une transition à partir d'une position p dans un terme t, un tel automate sera éventuellement amené à vérifier que $t_{p11} = t_{p221}$ ou bien que $t_{p2} \neq t_{p311}$. Les contraintes pour les règles de transition de tels automates peuvent aussi s'exprimer des deux façons définies ci-dessus.

Dans la suite de cette section, nous nous intéressons à la classe des automates d'arbres à contraintes et dans les sections suivantes à certaines de ses sous-classes. Afin de simplifier les définitions et notations, la forme d'expression utilisée pour les contraintes des règles de transition varie suivant les automates considérés.

2.3.1 Définitions

Nous donnons tout d'abord la définition formelle et générale des contraintes qui vont être ajoutées aux règles de transition. Ensuite nous définissons les automates à contraintes.

Les *contraintes égalitaires* sont des prédicats sur les termes de $T(\Sigma)$, Σ étant une signature quelconque. Syntaxiquement ces prédicats sont la constante $\top$ (*contrainte nulle*) ou bien un prédicat $\pi = \pi'$ où π et π' sont des éléments de $(\mathbb{N}_+)^+$ (l'ensemble des positions différentes de ϵ). La négation $\neg(\pi = \pi')$ d'une contrainte égalitaire est notée $\pi \neq \pi'$ et appelée *contrainte diségalitaire*. La négation de $\top$ est $\bot$ (*contrainte pleine*).

Soit t un terme de $T(\Sigma)$. On dit que t satisfait une contrainte égalitaire $\pi = \pi'$ et on note $t \vDash \pi = \pi'$ si π et π' sont des positions de t et $t|_\pi = t|_{\pi'}$. Dans le cas contraire, nous notons $t \vDash \pi \neq \pi'$. D'autre part, $t \vDash \top$ et $t \nvDash \bot$ pour tout terme t de $T(\Sigma)$.

Finalement la relation $\vDash$ est prolongée de façon usuelle à toute combinaison booléenne de contraintes d'égalité.

Exemple 80. *Soit la signature* $\Sigma = \{f/2, g/1, a/0\}$. *Alors*

$\vDash (1 = 21) \wedge (1 \neg 2)$.

Définition 81 (Automates à contraintes). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres à contraintes *(AC) est un quadruplet $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini d'états ($\Sigma \cap \mathcal{Q} = \emptyset$);*
3. *$\mathcal{Q}_f \subseteq \mathcal{Q}$ est l'ensemble des états finaux;*
4. *Δ est un système de réécriture dont les règles sont de la forme :*
   - *(a) $a \to q(a)$ où a est une constante de Σ et q un état de $\mathcal{Q}$;*
   - *(b) $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ où f est un symbole de Σ d'arité n supérieure ou égale à n, $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$, $x_1, \ldots, x_n$ des variables distinctes de $\mathcal{X}$ et c une combinaison booléenne de contraintes égalitaires.*

Les contraintes diségalitaires étant les négations des contraintes égalitaires, la combinaison booléenne de contraintes égalitaires de chaque règle est une formule formée de conjonctions et disjonctions de contraintes égalitaires et diségalitaires.

Par la suite, la contrainte nulle $\top$ est en général omise dans les règles où elle doit normalement apparaître.

Soit $\mathcal{A}$ un automate à contraintes. L'ensemble des règles Δ de $\mathcal{A}$ est un système de réécriture contraint, c'est-à-dire qu'un terme t de $T(\Sigma \cup \mathcal{Q})$ se réécrit en un terme t' de $T(\Sigma \cup \mathcal{Q})$ par application d'une règle contrainte $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ à une position p de t si et seulement si :

1. t se réécrit en t' par application de la règle $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ à la position p,
2. $f(t|_{p11}, \ldots, t|_{pn1}) \vDash c$.

Remarque 82. *Dans la formule* $f(t|_{p11}, \ldots, t|_{pn1}) \models c$, *la position* 1 *est ajoutée aux positions* $p1, \ldots, pn$ *car celles-ci correspondent aux positions des états, alors que la contrainte* c *est vérifiée sur les termes de* $T(\Sigma)$.

La relation binaire $\rightarrow_\Delta$ (notée en général $\rightarrow_\mathcal{A}$) est définie comme la relation de réécriture contrainte par une règle de Δ sur $T(\Sigma \cup \mathcal{Q})$. Un terme t de $T(\Sigma)$ est dit *reconnu* ou *accepté* par $\mathcal{A}$ si $t \xrightarrow{*}_\Delta q(t)$ avec q un état final $\mathcal{Q}_f$. Le langage des termes clos reconnus par $\mathcal{A}$ est noté $\mathcal{L}(\mathcal{A})$. Nous désignons par RAC la classe des langages d'arbres reconnus par les AC.

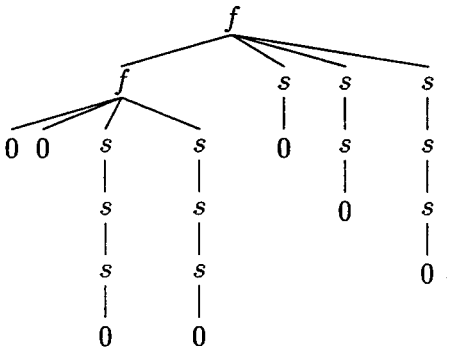
Le *membre gauche* (respectivement le *membre droit*) d'une règle $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ de $\mathcal{A}$ est $f(q_1(x_1), \ldots, q_n(x_n))[c]$ (respectivement $q(f(x_1, \ldots, x_n))$).

Exemple 83. *Nous considérons l'AC* $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ *avec* $\Sigma = \{f/4, s/1, 0/0\}$, $\mathcal{Q} = \{q_n, q_0, q_f\}$, $\mathcal{Q}_f = \{q_f\}$ *et* Δ *composé des règles suivantes:*

$$
\begin{array}{rcl}
0 & \rightarrow & q_0(0) \\
s(q_0(x_1)) & \rightarrow & q_n(s(x_1)) \\
s(q_n(x_1)) & \rightarrow & q_n(s(x_1)) \\
f(q_0(x_1), q_0(x_2), q_n(x_3), q_n(x_4)) & \xrightarrow{[3=4]} & q_f(f(x_1, x_2, x_3, x_4)) \\
f(q_0(x_1), q_0(x_2), q_0(x_3), q_0(x_4)) & \rightarrow & q_f(f(x_1, x_2, x_3, x_4)) \\
f(q_f(x_1), q_n(x_2), q_n(x_3), q_n(x_4)) & \xrightarrow{[14=4 \wedge 21=12 \wedge 131=3]} & q_f(f(x_1, x_2, x_3, x_4))
\end{array}
$$

Cet automate calcule la somme de deux entiers relatifs positifs écrit en base 1. *En effet, on a de façon évidente,* $\mathcal{L}(\mathcal{A}, q_0) = \{0\}$ *et* $\mathcal{L}(\mathcal{A}, q_n) = \{s^k(0) \mid k \in \mathbb{N}_+\}$. *De plus si* t *est reconnu par* $\mathcal{A}$, *alors on peut montrer qu'il existe un terme* t_1 *et* n, m *entiers tels que* $t = f(t_1, s^n(0), s^m(0), s^{n+m}(0))$.

Inversement pour tous n, m *entiers, il existe un terme* $f(t_1, s^n(0), s^m(0), s^{n+m}(0))$ *accepté par* $\mathcal{A}$. *Par exemple, l'arbre ci-dessous est accepté par* $\mathcal{A}$.



Remarquons que pour tout automate à contraintes $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$, il existe un automate à contraintes $\mathcal{A}' = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta')$ tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ et les règles de Δ' ne soient contraintes que par des conjonctions de contraintes égalitaires et diségalitaires. Pour construire Δ', il suffit de mettre la contrainte c de chaque règle de Δ sous forme normale disjonctive

($c = c_1 \vee \cdots \vee c_k$ avec les $c_1, \ldots, c_k$ conjonctions de contraintes égalitaires et diségalitaires) puis d'écrire une règle pour chacune des conjonctions de la forme normale.

Pour en terminer avec les définitions sur les automates à contraintes, nous définissons comme pour les automates d'arbres standards la notion de calcul.

Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ un automate à contraintes et t un terme clos de $T(\Sigma)$. Un *calcul* r de $\mathcal{A}$ sur t est une application r de $\mathcal{P}os(t)$ dans $\mathcal{Q}$ *compatible avec les règles* de Δ, c'est-à-dire pour toute position p de $\mathcal{P}os(t)$, si $t|_p = f(t_1, \ldots, t_n)$, $r(p) = q$, $r(pi) = q_i$ pour tout i de $[n]$, alors il existe une contrainte c telle que $f(t_1, \ldots, t_n) \vDash c$ et

$$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$$

soit une règle de Δ. Un calcul de $\mathcal{A}$ est *réussi* si $r(\epsilon)$ est un état final et un terme clos t de $T(\Sigma)$ est accepté par $\mathcal{A}$ si et seulement s' il existe un calcul réussi de $\mathcal{A}$ sur t.

2.3.2 Expressivité

Pour tout langage $\mathcal{L}$ reconnu par un *RATEG*, il existe un automate à contraintes $\mathcal{A}$ tel que $\mathcal{L} = \mathcal{L}(\mathcal{A})$. Nous en déduisons la propriété suivante.

Propriété 84. *Le pouvoir d'expression des automates à contraintes est supérieur à celui des RATEG.*

2.3.3 Déterminisation et complétion

Les notions de déterminisme et de complétion se définissent pour les automates à contraintes de façon similaire au cas des automates d'arbres standards.

Un automate à contraintes $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ est *déterministe* si et seulement si pour toute paire de règles $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c']} q'(f(x_1, \ldots, x_n))$ de Δ, la contrainte $c \wedge c'$ est insatisfiable si $q \neq q'$.

Donc si un automate est déterministe, pour tout terme t clos il existe au plus un état q tel que $t \xrightarrow{*} q(t)$.

Propriété 85 (Déterminisation, A.-C. Caron [14]). *Pour tout AC $\mathcal{A}$, il existe un AC $\mathcal{A}'$ déterministe tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Un automate à contraintes est *complet* si et seulement si pour toute terme t clos il existe au moins un état q tel que $t \xrightarrow{*} q(t)$.

Propriété 86 (Complétion, A.-C. Caron [14]). *Pour tout AC $\mathcal{A}$, il existe un AC $\mathcal{A}'$ complet tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

De plus, dans le cas de l'algorithme de complétion de A.-C. Caron [14], si $\mathcal{A}$ est déterministe alors $\mathcal{A}'$ est déterministe. Donc les *AC* ont les mêmes propriétés de déterminisation et de complétion que les *AAS*.

2.3.4 Propriétés de clôture

La classe *RAC* a les mêmes propriétés de clôture par opérations booléennes que la classe des langages réguliers.

Propriété 87 (Opérations booléennes). *La classe RAC est close par opérations booléennes (union, intersection et complément).*

2.3.5 Problème du vide

Nous venons de voir que les automates à contraintes conservent certaines bonnes propriétés des automates d'arbres standards. Malheureusement la décision du vide n'est pas conservée.

Propriété 88 (Problème du vide). *Le problème du vide est indécidable pour les AC.*

Cette propriété se déduit de l'indécidabilité du problème du vide pour les *RATEG* (propriété 75) et de la propriété 84.

La classe des automates à contraintes ne peut donc être utilisée pour résoudre des problèmes de décision concernant des systèmes non linéaires. Pour utiliser ces automates dans la résolution de tels problèmes, des sous-classes d'automates d'arbres à contraintes pour lesquelles le vide est décidable ont été considérées. Ces sous-classes sont caractérisées par des restrictions sur la syntaxe des contraintes d'égalité ou sur les positions où s'appliquent de telles contraintes. Après avoir cité quelques applications possibles des *AC*, nous rappelons les sous-classes de la classe des *AC* pour lesquelles le vide est décidable.

2.3.6 Applications

Les automates à contraintes permettent de généraliser aux termes et systèmes de réécriture non linéaires les résultats de reconnaissabilité de la section **Applications** du chapitre **Automates d'arbres finis**.

Pour l'ensemble de cette section, Σ est une signature et $\mathcal{X}$ un ensemble de variables.

Propriété 89 (*AC* reconnaissabilité des instances closes). *Si t est un terme de l'ensemble $T(\Sigma, \mathcal{X})$, alors l'ensemble des instances closes de t appartient à RAC.*

Dans un premier temps, on considère $\tilde{t}$, le terme linéarisé de t, c'est-à-dire que $\tilde{t}$ est linéaire et qu'il existe un renommage de variables (éventuellement non injectif) σ tel que $t = \tilde{t}\sigma$. D'après la propriété 54, il existe un automate d'arbres standard $\mathcal{A}$ reconnaissant l'ensemble des instances closes de $\tilde{t}$.

Ensuite, on remplace chaque règle de $\mathcal{A}$ donnant un état final par une règle, de même ossature, contrainte par les égalités que doivent satisfaire les termes pour satisfaire les non linéarités de t. On obtient un *AC* reconnaissant l'ensemble des instances closes de t.

Exemple 90. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$ et le terme $t = f(x, g(x))$. Le terme linéarisé de t est $\tilde{t} = f(x, g(y))$ et l'ensemble de ses instances closes est reconnu par l'AAS $\mathcal{A}$ de l'exemple 55. On en déduit l'AC $\mathcal{A}_t = (\Sigma, \{q, q_g, q_f\}, \{q_f\}, \Delta_t)$ dont les règles sont:*

$$
\begin{array}{rcl@{\qquad}rcl}
a & \to & q(a) & f(q(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\
g(q(x_1)) & \to & q_g(g(x_1)) & f(q_g(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\
g(q_g(x_1)) & \to & q_g(g(x_1)) & f(q(x_1), q_g(x_2)) & \xrightarrow{[1=21]} & q_f(f(x_1, x_2)) \\
 & & & f(q_g(x_1), q_g(x_2)) & \xrightarrow{[1=21]} & q_f(f(x_1, x_2))
\end{array}
$$

Le langage reconnu par $\mathcal{A}_t$ est l'ensemble des instances closes de t.

Puisque l'ensemble des termes clos réductibles par un système de réécriture est reconnaissable par *RATEG* (propriété 76) et le pouvoir d'expression des automates à contraintes est strictement supérieur à celui des *RATEG* (propriété 84), nous avons la propriété suivante.

Propriété 91 (*AC* reconnaissabilité des termes clos réductibles). *Soit $\mathcal{R}$ un système de réécriture sur Σ. L'ensemble des termes clos réductibles par $\mathcal{R}$ appartient à RAC.*

En fait, pour une règle de réécriture $l \to r$, on construit un *AC* reconnaissant l'ensemble des instances closes de l par la construction réalisée dans la preuve de la propriété 89. On en déduit un *AC* reconnaissant l'ensemble des termes clos réductibles par la règle $l \to r$. Les contraintes égalitaires de cet automate permettent de vérifier qu'à une certaine position d'un terme on peut appliquer la règle $l \to r$. Elles assurent les égalités des sous-termes aux positions correspondant à des occurrences multiples de variables dans l (exemple 92).

Exemple 92. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$ et le système de réécriture $\mathcal{R} := \{f(x, g(x)) \to a\}$. On part de l'automate $\mathcal{A}_t$ reconnaissant l'ensemble des instances closes de $f(x, g(x))$ (exemple 90) et on lui ajoute les règles suivantes :*

$$\begin{array}{rcl@{\qquad}rcl} g(q_f(x_1)) & \to & q_f(g(x_1)) & f(q_f(x_1), q(x_2)) & \to & q_f(f(x_1, x_2)) \\ f(q(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) & & & \end{array}$$

L'automate obtenu reconnaît l'ensemble des termes clos réductibles par $\mathcal{R}$.

La classe *RAC* étant close par complément (propriété 87), on obtient en prenant l'automate complément de l'automate de la propriété 91 :

Propriété 93 (*AC* reconnaissabilité des formes normales). *Soit $\mathcal{R}$ un système de réécriture sur Σ. L'ensemble des termes clos en forme normale pour $\mathcal{R}$ appartient à RAC.*

Exemple 94. *Nous reprenons l'exemple précédent (exemple 92) et nous considérons l'AC $\mathcal{A}_n = (\Sigma, \{q, q_g\}, \{q, q_g\}, \Delta_n)$ dont les règles sont :*

$$\begin{array}{rcl@{\qquad}rcl} a & \to & q(a) & f(q(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\ g(q(x_1)) & \to & q_g(g(x_1)) & f(q_g(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\ g(q_g(x_1)) & \to & q_g(g(x_1)) & f(q(x_1), q_g(x_2)) & \xrightarrow{[1\neq 21]} & q(f(x_1, x_2)) \\ & & & f(q_g(x_1), q_g(x_2)) & \xrightarrow{[1\neq 21]} & q(f(x_1, x_2)) \end{array}$$

Chaque fois que l'automate rencontre dans un terme t un sous-terme de la forme $f(t_1, g(t_2))$ avec t_1, t_2 termes, il vérifie que ce terme n'est pas réductible à cette position ($t_1 \neq t_2$). Cette vérification est réalisé par la dernière règle ci-dessus.

On en déduit que le langage reconnu par $\mathcal{A}_n$ est l'ensemble des termes clos en forme normale pour $\mathcal{R}$.

2.4 Automates à contraintes diségalitaires

Dans sa thèse [46], F. Jacquemard restreint la classe des automates d'arbres à contraintes en interdisant toute contrainte égalitaire dans les règles de transition. Les automates ainsi obtenus sont appelés automates à contraintes diségalitaires et le vide est décidable pour ceux-ci. En revanche, certaines propriétés de clôture des automates à contraintes sont perdues, comme le complément ou la déterminisation. Mais ces automates n'en ont pas moins d'importantes applications, en particulier ils permettent la reconnaissance de l'ensemble des formes normales closes de n'importe quel système de réécriture.

2.4.1 Définition

Définition 95 (*ACD*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres à contraintes diségalitaires *(ACD) est un automate à contraintes (Σ, Q, Q_f, Δ) dont les règles de transition de Δ sont la forme :*

1. *$a \to q(a)$ où a est une constante de Σ et q un état de Q;*

2. *$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ où f est un symbole de Σ d'arité n strictement supérieure à un, $q_1, \ldots, q_n, q$ des états de Q, $x_1, \ldots, x_n$ des variables distinctes de $\mathcal{X}$ et c une combinaison booléenne de contraintes diségalitaires ne contenant que des connecteurs $\vee$ et $\wedge$.*

Nous désignons par *RACD* la classe des langages d'arbres reconnus par les *ACD*.

2.4.2 Déterminisation et complétion

Les *ACD* conservent la propriété de complétion des automates des *AC*. Par contre la propriété de déterminisation n'est pas conservée.

Propriété 96 (Complétion, F. Jacquemard [46]). *Pour tout ACD $\mathcal{A}$, il existe un ACD $\mathcal{A}'$ complet tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Propriété 97 (Déterminisation, F. Jacquemard [46]). *La classe des ACD n'est pas close par déterminisation.*

2.4.3 Propriétés de clôture

Toutes les propriétés de clôture des *AC* ne sont pas vérifiées par les *ACD*. En effet, la classe des langages reconnus par les automates à contraintes diségalitaires n'est pas close par complément (propriété 99). Par contre :

Propriété 98 (Union, Intersection, F. Jacquemard [46]). *La classe RACD est close par union et intersection.*

Propriété 99 (Complément, F. Jacquemard [46]). *La classe RACD n'est pas close par complément.*

2.4.4 Problème du vide

Les *ACD* ne possèdent donc pas toutes les propriétés des *AC*. Mais contrairement aux *AC*, le problème du vide est décidable pour ces automates.

Propriété 100 (Problème du vide). *Le problème du vide est décidable pour les ACD.*

La preuve de cette propriété établie par F. Jacquemard [46] s'inspire de la preuve de décidabilité du problème du vide pour les automates d'arbres standards. F. Jacquemard définit une opération de réduction sur les calculs d'un automate à contraintes diségalitaires $\mathcal{A}$, généralisant l'opération de pompage sur les calculs des automates d'arbres standards. Ensuite, il détermine une condition suffisante de réductibilité par pompage sur les calculs de l'automate $\mathcal{A}$. Il s'agit d'une borne en profondeur $b(\mathcal{A})$ telle que si un calcul r de $\mathcal{A}$ est plus grand que $b(\mathcal{A})$, alors on peut réduire par pompage r en un calcul r' de $\mathcal{A}$. Donc il suffit d'énumérer tous les termes de $T(\Sigma)$ de hauteur inférieure ou égale à $b(\mathcal{A})$ et de tester leur appartenance à $\mathcal{L}(\mathcal{A})$ pour savoir si $\mathcal{L}(\mathcal{A})$ est vide ou non.

2.4.5 Expressivité

Tout *AAS* étant un *ACD*, le pouvoir d'expression des *ACD* est supérieur à celui des *AAS*. De plus la classe des réguliers étant close par complément (propriété 45) mais pas la classe *RACD* (propriété 99), nous avons la propriété suivante.

Propriété 101. *Le pouvoir d'expression des ACD est strictement supérieur à celui des AAS.*

2.4.6 Applications

Nous avons vus qu'étant donnée une signature Σ et un système de réécriture $\mathcal{R}$, il existe un *AC* reconnaissant l'ensemble des termes clos en forme normale pour $\mathcal{R}$ (propriété 93). En fait, on peut remarquer que l'automate construit est un *ACD*. En effet, les seuls tests qu'un automate à contraintes doit effectuer dans ses transitions pour vérifier qu'un sous-terme n'est pas réductible par une règle de $\mathcal{R}$ sont des inégalités de sous-termes (exemple 94). De tels automates peuvent donc être construits dans la sous-classe *ACD*.

Propriété 102 (*ACD* reconnaissabilité des formes normales, F. Jacquemard [46]). *Soient Σ une signature, $\mathcal{X}$ un ensemble de variables et $\mathcal{R}$ un système de réécriture sur Σ. L'ensemble des termes clos en forme normale pour $\mathcal{R}$ appartient à RACD.*

Les *ACD* permettant de généraliser la propriété 64 utilisant les *AAS* pour montrer que le problème de réductibilité inductive est décidable pour les systèmes de réécriture linéaires à gauche (théorème 65).

Propriété 103. *Soient Σ une signature, $\mathcal{X}$ un ensemble de variables, $\mathcal{R}$ un système de réécriture sur Σ et t un terme de $T(\Sigma, \mathcal{X})$. Il existe un ACD $\mathcal{A}_t$ tel que $\mathcal{L}(\mathcal{A}_t) = \emptyset$ si et seulement si t est inductivement réductible pour $\mathcal{R}$.*

La décidabilité du problème de réductibilité inductive se déduit alors de la décidabilité du problème du vide pour les *ACD* (propriété 100). Nous rappelons que la première preuve de cette propriété est due à D. Plaisted [70].

Théorème 104 (Problème de réductibilité inductive). *Le problème de réductibilité inductive est décidable.*

2.5 Automates à tests entre frères

La classe des automates à tests entre frères, introduite par B. Bogaert et S. Tison [11], est une sous-classe de la classe AC pour laquelle les tests d'égalité et de différence sont restreints à des termes frères (tests à la profondeur 1). L'ensemble des résultats mentionnés dans cette section est issu de l'article de B. Bogaert et S. Tison [11].

2.5.1 Définition

Définition 105 (*ATF*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres à tests entre frères *(ATF) est un quadruplet $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini d'états ($\Sigma \cap \mathcal{Q} = \emptyset$);*
3. *$\mathcal{Q}_f \subseteq \mathcal{Q}$ est l'ensemble des états finaux;*
4. *Δ est un système de réécriture dont les règles sont de la forme :*
   (a) *$a \to q(a)$ où a est une constante de Σ et q un état de $\mathcal{Q}$;*
   (b) *$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ où f est un symbole de Σ d'arité n supérieure ou égale à un, $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$, $x_1, \ldots, x_n$ des variables distinctes de $\mathcal{X}$ et c une combinaison booléenne de contraintes égalitaires de la forme $\pi = \pi'$ avec $|\pi| = |\pi'| = 1$.*

Nous désignons par *RATF* la classe des langages d'arbres reconnus par les *ATF*.

2.5.2 Déterminisation et complétion

Les *ATF* conservent les propriétés de déterminisation et de complétion des *AC*.

Propriété 106 (Déterminisation, complétion, B. Bogaert et S. Tison [11]). *Pour tout ATF $\mathcal{A}$, il existe un ATF $\mathcal{A}'$ déterministe et complet tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

L'algorithme utilisé par B. Bogaert et S. Tison pour démontrer cette propriété a pour entrée un automate à contraintes pleines (définition 108) et donne en sortie un automate à contraintes pleines. Donc si $\mathcal{A}$ est un automate à contraintes pleines, $\mathcal{A}'$ l'est aussi.

2.5.3 Normalisation

Dans la définition des *ATF*, les règles peuvent contenir des contraintes imposant des égalités entre des sous-termes atteignant des états différents (par exemple la règle $f(q_1(x_1), q_2(x_2)) \xrightarrow{[1=2]} q(f(x_1, x_2))$ avec $q_1 \neq q_2$). Pourtant, il est souvent intéressant et simplificateur de pouvoir supposer que les égalités ne sont imposées qu'entre des sous-termes atteignant des états identiques. Les *ATF* satisfaisant cette propriété sont dits normalisés.

Définition 107 (*ATF* normalisé). *Un ATF est* normalisé *si pour toute règle*

$$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$$

de l'automate et toute contrainte égalitaire $\pi = \pi'$ de c, on a $q_\pi = q_{\pi'}$.

Dans le chapitre **Automates à tests d'égalité entre frères et cousins** de la partie **Automates à contraintes**, nous voyons que l'association des propriétés de complétude et de normalisation est également intéressante. Cette association donne des automates dans lesquels, pour tout terme t de $T(\Sigma, \mathcal{Q})$, il existe une et une seule règle applicable à t. Ces automates nécessitant une manipulation particulière des contraintes des règles, nous commençons par introduire un type particulier de contraintes.

Définition 108. *Soient Σ une signature et n un entier inférieur ou égal à l'arité maximale des symboles de fonction de Σ. Une* contrainte pleine *c sur n positions est une conjonction de contraintes égalitaires et diségalitaires telle que :*

- *Pout toute contrainte égalitaire $\pi = \pi'$ (diségalitaire $\pi \neq \pi'$), on a $|\pi| = |\pi'| = 1$, c'est-à-dire π et π' sont des entiers strictement positifs;*
- *Pour tous entiers π, π' inférieurs ou égaux à n, soit la contrainte égalitaire $\pi = \pi'$, soit la contrainte diségalitaire $\pi \neq \pi'$ apparaît dans c;*
- *c est satisfiable c'est-à-dire l'ensemble des termes t de $T(\Sigma)$ tels que $t \models c$ est non vide.*

En d'autres termes, si c est une contrainte pleine sur n positions, il existe une partition $(E_i)_{i\in I}$ de $[n]$, I ensemble d'indices, telle que :

$$c := \bigwedge_{k\in I} \bigwedge_{l,l'\in E_k} l = l' \ \wedge \bigwedge_{k,k'\in I, k\neq k'} \bigwedge_{l\in E_k, l'\in E_{k'}} l \neq l'.$$

Pour simplifier l'écriture, nous notons $c = (E_i)_{i\in I}$ et pour toute paire d'entiers k, l inférieurs ou égaux de n,

- $k \approx_c l$ si la contrainte $k = l$ apparaît dans c, et
- $k \not\approx_c l$ si la contrainte $k \neq l$ apparaît dans c.

Pour tout entier n strictement positif, nous désignons par EC'_n, l'ensemble des contraintes pleines sur n positions. Dans le cas particulier où $n = 0$, nous notons $EC'_0 = \{\top\}$.

Exemple 109. *Les contraintes pleines constituant l'ensemble EC'_3 sont les suivantes :*

$1=2 \ \wedge \ 1=3 \ \wedge \ 2=3$; $\quad 1\neq 2 \ \wedge \ 1=3 \ \wedge \ 2\neq 3$;
$1\neq 2 \ \wedge \ 1\neq 3 \ \wedge \ 2=3$; $\quad 1=2 \ \wedge \ 1\neq 3 \ \wedge \ 2\neq 3$;
$1\neq 2 \ \wedge \ 1\neq 3 \ \wedge \ 2\neq 3$.

De façon évidente, pour tout ATF $\mathcal{A}$, il existe un ATF $\mathcal{A}'$ à contraintes pleines tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A})'$. Nous ne donnons pas ici la construction qui consiste uniquement à transformer la combinaison booléenne de contraintes égalitaires de chaque règle en une disjonction de contraintes pleines équivalente et de créer une règle par membre de cette disjonction (cette opération augmente le nombre de règles). Lorsque nous nous restreignons aux automates à contraintes pleines les notions de déterminisme et de complétude restent les mêmes.

Remarquons que pour deux contraintes pleines c et c' de EC'_n différentes, la contrainte $c \wedge c'$ est insatisfiable. Nous avons donc :

Remarque 110. *Un ATF $\mathcal{A}$ à contraintes pleines est déterministe si pour toute paire de règles $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q'(f(x_1, \ldots, x_n))$ de $\mathcal{A}$, on a $q = q'$.*

Finalement, les restrictions faites sur les règles des automates pour qu'ils soient normalisés ne portant que sur les contraintes égalitaires, nous définissons la notion suivante. Soient n un entier strictement positif, c une contrainte pleine de EC'_n et $(s_i)_{i\in[n]}$ un n-uplet de symboles. Alors $(s_i)_{i\in[n]}$ *satisfait les égalités* de c si pour toute contrainte égalitaire $\pi = \pi'$ de c, on a $s_\pi = s_{\pi'}$.

Définition 111 (ATF normalisé-complet). *Un ATF $\mathcal{A} = (\Sigma, Q, Q_f, \Delta)$ est* normalisé-complet *si :*

- *Il est déterministe ;*
- *Toutes les contraintes de ses règles sont des contraintes pleines ;*
- *Pour tout symbole de fonction f de Σ d'arité n, pour tout n-uplet $(q_i)_{i\in[n]}$ d'états et toute contrainte pleine c de EC'_n dont les égalités sont satisfaites par $(q_i)_{i\in[n]}$, il existe un état q tel que la règle $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ appartienne à Δ ;*
- *Pour tout symbole de fonction f de Σ d'arité n, pour tout n-uplet $(q_i)_{i\in[n]}$ d'états et toute contrainte pleine c de EC'_n dont les égalités ne sont pas satisfaites par $(q_i)_{i\in[n]}$, il n'existe pas d'état q tel que la règle $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ appartienne à Δ.*

Exemple 112. *Nous considérons l'ATF $\mathcal{A} = (\Sigma, Q, Q_f, \Delta)$ défini par $\Sigma = \{f/2, h/1, a/0\}$, $Q = \{q, q_f\}$, $Q_f = \{q_f\}$ et les règles suivantes :*

$$
\begin{array}{rcl@{\qquad}rcl}
a & \to & q(a) & h(q(x_1)) & \to & q(h(x_1)) \\
h(q_f(x_1)) & \to & q_f(h(x_1)) & f(q(x_1), q(x_2)) & \xrightarrow{[1=2]} & q_f(f(x_1, x_2)) \\
f(q(x_1), q(x_2)) & \xrightarrow{[1\neq 2]} & q(f(x_1, x_2)) & f(q(x_1), q_f(x_2)) & \xrightarrow{[1\neq 2]} & q(f(x_1, x_2)) \\
f(q_f(x_1), q(x_2)) & \xrightarrow{[1\neq 2]} & q(f(x_1, x_2)) & f(q_f(x_1), q_f(x_2)) & \xrightarrow{[1=2]} & q_f(f(x_1, x_2)) \\
f(q_f(x_1), q_f(x_2)) & \xrightarrow{[1\neq 2]} & q(f(x_1, x_2)) & & &
\end{array}
$$

$\mathcal{A}$ est de façon évidente normalisé-complet.

Les *ATF* possèdent la propriété de normalisation-complétion (propriété 113) ce qui permet de simplifier un certain nombre de preuves sur cette classe d'automates. En particulier, les preuves de clôture par opérations booléennes de la classe des langages reconnus par la classe des *ATF* sont faites en considérant des automates normalisés-complets (section 2.5.4).

Propriété 113 (Normalisation-complétion, B. Bogaert et S. Tison [11]). *Pour tout ATF $\mathcal{A}$, il existe un ATF $\mathcal{A}'$ normalisé-complet tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

2.5.4 Propriétés de clôture

La classe *RATF* a les mêmes propriétés de clôture booléennes que la classe *RAC*.

Propriété 114 (Opérations booléennes, B. Bogaert et S. Tison [11]). *La classe RATF est close par opérations booléennes (union, intersection et complément).*

B. Bogaert et S. Tison ont également étudié dans leur article [11], les propriétés de clôture par morphismes d'arbres de la classe *RATF* et ont obtenus les résultats suivants.

Propriété 115 (Morphismes symbole-symbole, B. Bogaert et S. Tison [11]). *La classe RATF est close par morphismes d'arbres symbole-symbole.*

Propriété 116 (Morphismes inverse, B. Bogaert et S. Tison [11]). *La classe RATF n'est pas close par morphismes d'arbres inverses.*

2.5.5 Problèmes de décision

L'indécidabilité du problème du vide pour les *AC* (propriété 88) vient de la possibilité de superposer des tests d'égalité dans un calcul de l'automate. D'autre part, la preuve de décidabilité du problème du vide pour les *ACD* (propriété 100) est rendue difficile par la possibilité de superposer (croiser) des tests de différence. Pour les *ATF*, on n'a jamais de telles superpositions, ce qui d'une part rend le problème du vide décidable et d'autre part simplifie la preuve de décidabilité.

Propriété 117 (Problème du vide, B. Bogaert et S. Tison [11]). *Le problème du vide est décidable pour les ATF.*

Donc les *ATF* ont l'avantage de conserver les propriétés des *AC* tout en ayant un problème du vide décidable. B. Bogaert et S. Tison ont également étudié le problème de finitude pour les *ATF*.

Propriété 118 (Problème de finitude, B. Bogaert et S. Tison [11]). *Le problème de finitude est décidable pour les ATF.*

2.5.6 Applications

La restriction sur les contraintes des *ATF* étant assez forte, leur puissance d'expression s'en trouve forcément limitée par rapport aux automates d'arbres à contraintes. Néanmoins, ces automates permettent d'améliorer dans des cas intéressants certains résultats obtenus avec les automates d'arbres standards (section **Applications** du chapitre **Automates d'arbres finis**).

Pour la suite, nous considérons une signature Σ et un ensemble de variables $\mathcal{X}$.

Tout d'abord, la décidabilité du problème de réductibilité inductive pour les systèmes de réécriture linéaire à gauche (théorème 65) s'étend ainsi :

Théorème 119 (Problème de réductibilité inductive, B. Bogaert et S. Tison [11]). *Le problème de réductibilité inductive est décidable dans le cas d'un terme t semi-linéaire de $T(\Sigma, \mathcal{X})$ et d'un système de réécriture $\mathcal{R}$ semi-linéaire à gauche.*

En effet, les semi-linéarités permettent de montrer que l'ensemble des instances closes de t et l'ensemble des formes normales closes de $\mathcal{R}$ sont reconnaissables par ATF (propriétés 120 et 121). Donc l'intersection de ces deux ensembles l'est aussi puisque la classe $RATF$ est close par intersection (propriété 114).

Propriété 120 (ATF reconnaissabilité des instances closes). *Si t est un terme semi-linéaire de $T(\Sigma, \mathcal{X})$, alors l'ensemble des instances closes de t appartient à $RATF$.*

Propriété 121 (ATF reconnaissabilité des formes normales). *Soit $\mathcal{R}$ un système de réécriture semi-linéaire à gauche. Alors l'ensemble des termes clos en forme normale pour $\mathcal{R}$ appartient à $RATF$.*

De plus, la propriété d'entourage (propriété 57) se généralise facilement au cas des termes linéaires pour lesquels les non linéarités apparaissent entre positions frères.

Propriété 122. *Si t est un terme semi-linéaire de $T(\Sigma, \mathcal{X})$, alors l'ensemble des termes entourant t appartient à $RATF$.*

Exemple 123. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$, le terme $t = f(x, x)$ et l'ATF $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où $\mathcal{Q} = \{q, q_f\}$, $\mathcal{Q}_f = \{q_f\}$ et Δ est constitué des règles suivantes:*

$$
\begin{array}{rcl@{\qquad}rcl}
a & \to & q(a) & g(q(x_1)) & \to & q(g(x_2)) \\
g(q_f(x_1)) & \to & q_f(g(x_1)) & f(q(x_1), q(x_2)) & \to & q(f(x_1, x_2)) \\
f(q(x_1), q(x_2)) & \xrightarrow{[1=2]} & q_f(f(x_1, x_2)) & f(q_f(x_1), q(x_2)) & \to & q_f(f(x_1, x_2)) \\
f(q(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) & & &
\end{array}
$$

L'automate $\mathcal{A}$ reconnaît l'ensemble des termes entourant t.

Vérification de sortes

Dans la section 2.2.5, nous avons vus que les $RATEG$ permettent de reconnaître l'ensemble des termes clos bien sortés suivant une signature avec déclarations de sortes (propriété 79). Ce résultat reste valable dans le cas des ATF à condition que les non linéarités des déclarations ne fassent intervenir que des positions frères.

Propriété 124. *Soit une signature multi-sortée telle que pour toute déclaration de sorte $t \in s$, t est un terme semi-linéaire. Alors pour toute sorte s, il existe un ATF $\mathcal{A}_s$ reconnaissant l'ensemble des termes clos de sorte s.*

La propriété 122 et les bonnes propriétés des ATF (propriétés 114 et 117) permettent d'étendre le résultat de décidabilité de la théorie de la réduction avec sortes obtenue par les AAS (théorème 70) aux déclarations de sortes.

Théorème 125. *Le problème de satisfiabilité du fragment de la théorie de la réduction avec déclaration de sortes dans lequel les prédicats $red_t(.)$ et les déclarations de sorte $t' \in s$ sont tels que t et t' sont semi-linéaires est décidable.*

2.6 Automates à tests d'égalité entre frères

Avant de définir les automates à tests entre frères, B. Bogaert et S. Tison [10] avait défini les automates à tests d'égalité entre frères. La classe de ces automates est une sous-classe de la classe des *ATF* dans laquelle les contraintes des règles se limitent à des contraintes égalitaires entre positions frères. L'ensemble des résultats mentionnés dans cette section est issu de l'article de B. Bogaert et S. Tison [10].

2.6.1 Définition

Définition 126 (*ATEF*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres à tests d'égalité entre frères *(ATEF) est un quadruplet $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini d'états ($\Sigma \cap \mathcal{Q} = \emptyset$);*
3. *$\mathcal{Q}_f \subseteq \mathcal{Q}$ est l'ensemble des états finaux;*
4. *Δ est un système de réécriture dont les règles sont de la forme :*
   (a) *$a \to q(a)$ où a est une constante de Σ et q un état de $\mathcal{Q}$;*
   (b) *$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ où f est un symbole de Σ d'arité n supérieure ou égale à un, $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$ et $x_1, \ldots, x_n$ des variables de $\mathcal{X}$.*

Notons que les règles peuvent être non linéaires. Cette non linéarité permet d'imposer des égalités entre termes frères.

Remarque 127. *Les égalités imposées par les automates à tests d'égalité entre frères sont exprimées par la non linéarité des parties gauches de règles et non par des contraintes égalitaires.*

Nous désignons par *RATEF* la classe des langages d'arbres reconnus par les *ATEF*.

2.6.2 Propriétés de clôture

Les *ATEF* n'ont pas toutes les propriétés de clôture des *ATF*. En particulier l'absence de contraintes diségalitaires dans les règles rend la classe des langages reconnus par la classe des *ATEF* non close par complément.

Propriété 128 (Union, intersection, B. Bogaert et S. Tison [10]). *La classe RATEF est close par union et intersection.*

Propriété 129 (Complément, B. Bogaert et S. Tison [10]). *La classe RATEF n'est pas close par complément.*

2.6.3 Expressivité

Tout *AAS* est de façon évidente un *ATEF*. De plus la classe des réguliers étant close par complément (propriété 45) mais pas la classe *RATEF* (propriété 129), nous avons la propriété suivante.

Propriété 130. *Le pouvoir d'expression des ATEF est strictement supérieur à celui des AAS.*

Par exemple, l'ensemble des arbres binaires équilibrés appartient à la classe *RATEF* (exemple 131) mais n'est pas régulier.

Exemple 131. *Soit la signature* $\Sigma = \{f/2, a/0\}$. *Les termes équilibrés de* $T(\Sigma)$ *sont définis récursivement par :*

- *a est équilibré ;*
- *Si* t_1 *et* t_2 *sont équilibrés, alors* $f(t_1, t_2)$ *est équilibré.*

L'automate à tests d'égalité entre frères $\mathcal{A} = (\Sigma, \{q\}, \{q\}, \Delta)$ *dont les règles sont définies ci-dessous reconnaît les termes équilibrés de* $T(\Sigma)$.

$$\begin{array}{rcl} a & \to & q(a) \\ f(q(x), q(x)) & \to & q(f(x,x)) \end{array}$$

Tout *ATEF* est de façon évidente équivalent à un *ATF*. De plus la classe *RATF* étant close par complément (propriété 114) mais pas la classe *RATEF* (propriété 129), nous avons la propriété suivante.

Propriété 132. *Le pouvoir d'expression des ATF est strictement supérieur à celui des ATEF.*

Par exemple, l'ensemble des arbres binaires non équilibrés est reconnaissable par *ATF* (exemple 133) mais pas par *ATEF*.

Exemple 133. *Soient la signature* $\Sigma = \{f/2, a/0\}$ *et l'automate à tests entre frères* $\mathcal{A} = (\Sigma, \{q, q_f\}, \{q_f\}, \Delta)$ *dont les règles sont définies ci-dessous.*

$$\begin{array}{rcl} a & \to & q(a) \\ f(q(x_1), q(x_2)) & \to & q_f(f(x_1, x_2)) \\ f(q(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) \end{array} \qquad \begin{array}{rcl} f(q(x_1), q(x_2)) & \xrightarrow{[1=2]} & q(f(x_1, x_2)) \\ f(q_f(x_1), q(x_2)) & \to & q_f(f(x_1, x_2)) \\ f(q_f(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) \end{array}$$

$\mathcal{A}$ reconnaît les termes non équilibrés de $T(\Sigma)$.

2.6.4 Normalisation

La notion de normalisation définie pour les *ATF* reste valable pour les *ATEF* puisqu'elle ne fait intervenir que des contraintes égalitaires. Nous adaptons donc sa définition à la forme des règles des *ATEF*.

Définition 134 (*ATEF* normalisé). *Un ATEF est* normalisé *si pour toute règle*

$$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$$

de $\mathcal{A}$, $(x_i)_{i\in[n]}$ est valide *pour $(q_i)_{i\in[n]}$, c'est-à-dire si: $\forall i, j \in [n], x_i = x_j \Rightarrow q_i = q_j$.*

Propriété 135 (Normalisation, B. Bogaert et S. Tison [10]). *Pour tout ATEF $\mathcal{A}$, il existe un ATEF $\mathcal{A}'$ normalisé tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Cette propriété est utilisée pour montrer que le pouvoir d'expression des automates à tests d'égalité entre frères et cousins simples est supérieur à celui des *ATEF* (lemme 187).

2.6.5 Problème du vide

Puisque tout *ATEF* est un *ATF* et le problème est décidable pour les *ATF* (propriété 117), le problème du vide est décidable pour les *ATEF*.

Propriété 136 (Problème du vide). *Le problème du vide est décidable pour les ATEF.*

2.6.6 Application

Nous considérons Σ et Γ deux signatures et $\mathcal{X}$ un ensemble de variables.

Propriété 137. *Soient $\mathcal{L}$ un langage régulier et Φ un morphisme d'arbres de $T(\Sigma)$ dans $T(\Gamma)$ semi-linéaire. Alors $\Phi(\mathcal{L})$ appartient à RATEF.*

Dans la preuve de cette propriété, nous construisons un *ATEF* $\mathcal{A}'$ avec ϵ-transitions, et nous montrons que $\Phi(\mathcal{L}) = \mathcal{L}(\mathcal{A}')$. En procédant de façon similaire au cas des *AAS* (théorème 36), nous pouvons montrer qu'il existe un *ATEF* équivalent à $\mathcal{A}'$ sans ϵ-transitions.

Preuve : Soient $\mathcal{L}$ un langage régulier et Φ un morphisme d'arbres de $T(\Sigma)$ dans $T(\Gamma)$ semi-linéaire déterminé par l'application φ. D'après le théorème 42, il existe un *AAS* réduit $\mathcal{A} = (\Sigma, Q, Q_f, \Delta)$ tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}$. Tout d'abord, nous construisons un automate $\mathcal{A}' = (\Gamma, Q', Q'_f, \Delta')$.

Considérons une règle $r := f(q_1(y_1), \ldots, q_n(y_n)) \to q(f(y_1, \ldots, y_n))$ de Δ, le terme semi-linéaire $t_f = \varphi(f)$ de $T(\Gamma, \mathcal{X}_n)$ et l'ensemble des positions $\mathcal{P}os(t_f)$. Nous définissons l'ensemble d'états $Q^r = \{q_p^r \mid p \in \mathcal{P}os(t_f)\}$ et l'ensemble de règles Δ^r contenant les règles suivantes. Pour toute position p de $\mathcal{P}os(t_f)$,

1. Si $p = \epsilon$ et $t_f(p)$ est un symbole g de Γ_k, alors $g(q_1^r(z_1), \ldots, q_k^r(z_k)) \to q_\epsilon^r(g(z_1, \ldots, z_k))$ avec
$$\forall i, j \in [k], (z_i = z_j) \Leftrightarrow (t_f(i) \in \mathcal{X} \text{et } t_f(i) = t_f(j))$$
est une règle de Δ^r,

2. Si $p \neq \epsilon$ et $t_f(p)$ est un symbole g de Γ_k, alors $g(q^r_{p1}(z_1), \ldots, q^r_{pk}(z_k)) \to q^r_p(g(z_1, \ldots, z_k))$ avec $z_1, \ldots, z_k$ variables distinctes de $\mathcal{X}$ est une règle de Δ^r,

3. Si $t_f(p) = x_i$ variable de $\mathcal{X}_n$, alors $q_i(x) \to q^r_p(x)$ est une règle de Δ^r,

4. $q^r_\epsilon(x) \to q(x)$ est une règle de Δ^r.

Nous procédons de même pour toutes les règles de Δ. Nous supposons que les ensembles d'états $\mathcal{Q}^r$ sont disjoints et qu'ils sont disjoints de $\mathcal{Q}$. Nous définissons maintenant $\mathcal{A}'$ par:

- $\mathcal{Q}' = \mathcal{Q} \bigcup_{r \in \Delta} \mathcal{Q}^r$,
- $\mathcal{Q}'_f = \mathcal{Q}_f$,
- $\Delta' = \bigcup_{r \in \Delta} \Delta^r$.

$\mathcal{A}'$ est de façon évidente un *ATEF*. Montrons que $\Phi(\mathcal{L}) = \mathcal{L}(\mathcal{A}')$.

$\Phi(\mathcal{L}) \subseteq \mathcal{L}(\mathcal{A}')$. Pour toute règle $r := f(q_1(y_1), \ldots, q_n(y_n)) \to q(f(y_1, \ldots, y_n))$ de Δ, tous termes $t_1, \ldots, t_n$ de $T(\Gamma)$ tels que pour tout i de $[n]$, $t_i \xrightarrow{*}_{\mathcal{A}'} q_i(t_i)$ et toute position p de $\mathcal{P}os(t_f)$, nous montrons que

$$t'|_p \xrightarrow{*}_{\mathcal{A}'} q^r_p(t'|_p) \tag{1}$$

avec $t' = t_f\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ par induction sur la longueur de p.

Position feuille différente de ϵ: Soit p une position de $\mathcal{LP}os(t_f)$. Si $t_f(p)$ est une variable x_i, alors $t'|_p = t_i \xrightarrow{*}_{\mathcal{A}'} q_i(t_i) \to_{\mathcal{A}'} q^r_p(t_i)$ puisque la règle $q_i(x) \to q^r_p(x)$ appartient à Δ^r (point 3 de la définition de Δ^r).

Sinon $t_f(p)$ est une constante a de Γ, alors $t'|_p = a \to_{\mathcal{A}'} q^r_p(a)$ puisque la règle $a \to q^r_p(a)$ appartient à Δ^r (point 2 de la définition de Δ^r).

Donc dans tous les cas, $t'|_p \xrightarrow{*}_{\mathcal{A}'} q^r_p(t'|_p)$.

Position non feuille différente de ϵ: Soit p une position de $\mathcal{IP}os(t_f)$ différente de ϵ. Supposons que la propriété (1) soit vraie pour toute position de longueur strictement supérieure à celle de p.

Il existe un symbole g de Γ d'arité k différente de zéro tel que $t'|_p = g(t'|_{p1}, \ldots, t'|_{pk})$. Pour tout entier i de $[k]$, on a par hypothèse d'induction, $t'|_{pi} \xrightarrow{*}_{\mathcal{A}'} q^r_{pi}(t'|_{pi})$ puisque $|pi| > |p|$.

De plus la règle $g(q^r_{p1}(z_1), \ldots, q^r_{pk}(z_k)) \to q^r_p(g(z_1, \ldots, z_k))$ où $z_1, \ldots, z_k$ sont des variables distinctes appartient à Δ^r (point 2 de la définition de Δ^r). Donc

$$t'|_p \xrightarrow{*}_{\mathcal{A}'} g(q^r_{p1}(t'|_{p1}), \ldots, q^r_{pk}(t'|_{pk})) \to_{\mathcal{A}'} q^r_p(t'|_p).$$

La propriété (1) est donc vérifiée pour p.

Position ϵ : Si $t_f(\epsilon)$ est une variable ou une constante, la propriété (1) est vraie (cas des positions feuilles).

Supposons donc qu'il existe un symbole g de Γ d'arité k différente de zéro tel que $t' = g(t'|_1, \ldots, t'|_k)$. Pour tout entier i de $[k]$, la propriété (1) est vraie pour la position i d'après les deux cas précédents donc $t'|_i \xrightarrow{*}_{\mathcal{A}'} q_i^r(t'|_i)$. D'où $t' \xrightarrow{*}_{\mathcal{A}'} g(q_1^r(t'|_1), \ldots, q_k^r(t'|_k))$.

La règle $g(q_1^r(z_1), \ldots, q_k^r(z_k)) \to q_\epsilon^r(g(z_1, \ldots, z_k))$ avec

$$\forall i, j \in [k], (z_i = z_j) \Leftrightarrow (t_f(i) \in \mathcal{X} \text{et } t_f(i) = t_f(j))$$

appartient à Δ^r (point 1 de la définition de Δ^r). Donc pour tous entiers i, j de $[k]$ tels que $i \neq j$, si $z_i = z_j$, on a $t_f(i)$ est une variable et $t_f(i) = t_f(j)$. Nous en déduisons que $t'|_i = t'|_j$. Donc

$$g(q_1^r(t'|_1), \ldots, q_k^r(t'|_k)) \to_{\mathcal{A}'} q_\epsilon^r(t')$$

par la règle ci-dessus. La propriété (1) est donc vérifiée pour la position ϵ.

La propriété (1) est donc vraie. Nous en déduisons que :

$$t_f\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\} \xrightarrow{*}_{\mathcal{A}'} q(t_f\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}) \tag{2}$$

puisque $t'|_\epsilon = t_f\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ et la règle $q_\epsilon^r \to q$ appartient à Δ (point 4 de la définition de Δ^r).

Nous montrons maintenant que :

$$\forall t \in T(\Sigma)\ t \xrightarrow{*}_{\mathcal{A}} q(t) \ \Rightarrow\ \Phi(t) \xrightarrow{*}_{\mathcal{A}'} q(\Phi(t)) \tag{3}$$

par induction sur la longueur de la chaîne de dérivation $t \xrightarrow{*}_{\mathcal{A}} q(t)$.

Base : Soit t un terme de $T(\Sigma)$ tel que $t \to_{\mathcal{A}} q(t)$. Alors t est une constante a de Σ et la règle $a \to q$ appartient à Δ. Donc $\Phi(t) = t_f$ est un terme clos et nous avons $t_f \xrightarrow{*}_{\mathcal{A}'} q(t_f)$ par la propriété (2). Donc la propriété (3) est vraie pour le cas de base.

Induction : Soit k un entier strictement positif. Supposons que la propriété (3) soit vraie pour toute chaîne de dérivation de longueur inférieure ou égale à k.

Soit t un terme de $T(\Sigma)$ tel que $t \xrightarrow{*}_{\mathcal{A}} f(q_1(t_1), \ldots, q_n(t_n)) \xrightarrow{r}_{\mathcal{A}} q(t)$ par une chaîne de dérivation de longueur $k+1$ avec $r := f(q_1(y_1), \ldots, q_n(y_n)) \to q(f(y_1, \ldots, y_n))$. Par hypothèse d'induction, nous avons pour tout i de $[n]$, $\Phi(t_i) \xrightarrow{*}_{\mathcal{A}'} q_i(\Phi(t_i))$ puisque $t_i \xrightarrow{*}_{\mathcal{A}} q_i(t_i)$ par une chaîne de longueur inférieure ou égale à k.

Par définition, $\Phi(t) = t_f\{x_1 \leftarrow \Phi(t_1), \ldots, x_n \leftarrow \Phi(t_n)\}$. Donc par la propriété (2), $\Phi(t) \xrightarrow{*}_{\mathcal{A}'} q(\Phi(t))$ ce qui termine la preuve de la propriété (3).

Nous déduisons de cette propriété et de $\mathcal{Q}'_f = \mathcal{Q}_f$ que $\Phi(\mathcal{L}) \subseteq \mathcal{L}(\mathcal{A}')$.

$\mathcal{L}(\mathcal{A}') \subseteq \Phi(\mathcal{L})$. Nous montrons que :

$$\forall t' \in T(\Gamma),\ t' \xrightarrow{*}_{\mathcal{A}'} q(t'), q \in \mathcal{Q} \ \Rightarrow\ \exists t \in \mathcal{L}(\mathcal{A}, q) \text{tel que } t' = \Phi(t) \qquad (4)$$

par induction sur le nombre d'états de $\mathcal{Q}$ apparaissant dans la réduction de t' en $q(t')$.

Base : Soit t' un terme de $T(\Gamma)$ tel que $t' \xrightarrow{*}_{\mathcal{A}'} q(t')$ avec q état de $\mathcal{Q}$ et tel que $q$soit le seul état de $\mathcal{Q}$ apparaîssant dans la réduction de t' en $q(t')$. Donc puisque les ensembles d'états $\mathcal{Q}^r$ sont disjoints deux à deux, il existe une règle $r := f(q_1(y_1), \ldots, q_n(y_n)) \rightarrow q(f(y_1, \ldots, y_n))$ de Δ telle que seules les règles de Δ^r sont utilisées dans la réduction de t' en $q(t')$. Nous déduisons des règles de Δ^r qu'il existe un symbole f de Σ tel que $t' = \varphi(f)$.

Puisque $\mathcal{A}$ est réduit, il existe des termes $t_1, \ldots, t_n$ de $T(\Sigma)$ tels que pour tout i de $[n]$, $t_i \xrightarrow{*}_{\mathcal{A}} q_i(t_i)$. Donc $t = f(t_1, \ldots, t_n) \xrightarrow{*}_{\mathcal{A}} q(t)$. De plus $\Phi(t) = t'$ donc la propriété (4) est vraie pour le cas de base.

Induction : Soit k un entier non nul. Supposons que la propriété (4) soit vraie pour toute dérivation dans laquelle apparaît au plus k états de $\mathcal{Q}$.

Soit t' un terme de $T(\Gamma)$ tel que $t' \xrightarrow{*}_{\mathcal{A}'} q(t')$ avec q état de $\mathcal{Q}$ et $k+1$ états de $\mathcal{Q}$ apparaissant dans la dérivation de t' en $q(t')$. Il existe alors C contexte de $\mathcal{C}^m(\Gamma)$ avec m entier strictement positif, $u'_1, \ldots, u'_m$ termes de $T(\Gamma)$, $s_1, \ldots, s_m$ états de $\mathcal{Q}$ tels que :

- $t' = C[u'_1, \ldots, u'_m]$,
- $t' \xrightarrow{*}_{\mathcal{A}'} C[s_1(u'_1), \ldots, s_m(u'_m)] \xrightarrow{*}_{\mathcal{A}'} q(t')$,
- q est le seul état de l'ensemble $\mathcal{Q}$ apparaîssant dans la dérivation de $C[s_1(u'_1), \ldots, s_m(u'_m)]$ en $q(t')$.

Par hypothèse d'induction, il existe $u_1, \ldots, u_m$ termes de $T(\Sigma)$ tels que pour tout i de $[m]$, $u'_i = \Phi(u_i)$ avec $u_i \xrightarrow{*}_{\mathcal{A}} s_i(u_i)$.

Puisque les ensembles d'états $\mathcal{Q}^r$ sont disjoints deux à deux, il existe une règle $r := f(q_1(y_1), \ldots, q_n(y_n)) \rightarrow q(f(y_1, \ldots, y_n))$ de Δ telle que seules les règles de Δ^r sont utilisées dans la réduction $C[s_1(u'_1), \ldots, s_m(u'_m)]$ en $q(t')$. Nous en déduisons qu'il existe un terme semi-linéaire t_f et $t'_1, \ldots, t'_n$ termes de $T(\Gamma)$ tels que $C[s_1(u'_1), \ldots, s_m(u'_m)] = t_f\{x_1 \leftarrow q_1(t'_1), \ldots, x'_n \leftarrow q_n(t'_n)\}$. Notons que $t' = C[u'_1, \ldots, u'_m] = t_f\{x_1 \leftarrow t'_1, \ldots, x'_n \leftarrow t'_n\}$.

Soit le terme $t = f(v_1, \ldots, v_n)$ tel que pour tout entier i de $[n]$,

- Si x_i apparaît dans t_f, alors $v_i = u_j$ avec j tel que $q_i(t'_i) = s_j(u'_j)$,
- Sinon v_i est un terme de $T(\Sigma)$ tel que $v_i \xrightarrow{*}_{\mathcal{A}} q_i(v_i)$ (v_i existe puisque $\mathcal{A}$ est réduit).

Soit i entier de $[n]$, tel que x_i apparaît dans t_f. Alors $v_i = u_j$ avec j tel que $q_i(t'_i) = s_j(u'_j)$. Donc $\Phi(v_i) = u'_j = t'_i$ et $q_i = s_j$. Or $u_j \xrightarrow{*}_{\mathcal{A}} s_j(u_j)$. D'où $v_i \xrightarrow{*}_{\mathcal{A}} q_i(v_i)$. Finalement pour tout entier i de $[n]$, $v_i \xrightarrow{*}_{\mathcal{A}} q_i(v_i)$. Nous avons donc $t' = \Phi(t)$ et

$$t \xrightarrow{*}_{\mathcal{A}} f(q_1(v_1), \ldots, q_n(v_n)) \xrightarrow{r}_{\mathcal{A}} q(t).$$

Finalement la propriété (4) est vraie.

Nous déduisons de cette propriété et de $\mathcal{Q}'_f = \mathcal{Q}_f$ que $\mathcal{L}(\mathcal{A}') \subseteq \Phi(\mathcal{L})$. Donc $\mathcal{L}(\mathcal{A}') = \Phi(\mathcal{L})$ d'où $\Phi(\mathcal{L})$ appartient à *RATEF*. □

2.7 Automates de réduction

Nous avons vu dans la section **Applications** du chapitre **Automates d'arbres standards** que le problème de satisfiabilité de la théorie de la réduction est décidable lorsqu'on se restreint à des termes linéaires (théorème 61). Dans [23], M. Dauchet, A.-C. Caron et J.-L. Coquidé montrent que ce problème est décidable dans le cas général en définissant une sous-classe des automates à contraintes, appelée classe des automates de réduction, pour laquelle le problème du vide est décidable.

2.7.1 Définition

Les restrictions sur les contraintes caractérisant les automates de réduction sont moins simples à exprimer que dans le cas des automates définis précédemment (*ACD*, *ATF* et *ATEF*).

En effet, la restriction n'est pas d'ordre syntaxique sur la forme des contraintes égalitaires ($\pi = \pi'$) et diségalitaires ($\pi \neq \pi'$) mais concerne l'ensemble des calculs pouvant être réalisés par l'automate. L'idée est de borner le nombre de superpositions (croisements) d'égalités testées dans les calculs, indépendamment du terme sur lequel l'automate calcule, afin d'obtenir la décidabilité du problème du vide.

Définition 138 (*AR*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate de réduction *(AR) est un automate à contraintes $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ tel qu'il existe un ensemble partiellement ordonné fini S et une fonction ω de $\mathcal{Q}$ vers S telle que pour toute règle de transition $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ de Δ, $\omega(q)$ est un majorant de $\{\omega(q_1), \ldots, \omega(q_n)\}$ dans S et de plus, c'est un majorant strict si c contient des contraintes égalitaires.*

Nous désignons par *RAR* la classe des langages d'arbres reconnus par les *AR*.

Remarque 139. *Soit un calcul r de $\mathcal{A}$ sur un terme t. Si en une position p de t une règle faisant intervenir des tests d'égalité est appliquée, on a $\omega(r(t|_p))$ strictement plus grand $\omega(r(t|_{p'}))$ pour toute position p' de t telle que $p \lhd p'$. Donc puisque S est fini, le nombre de tests d'égalité sur chaque branche est fini.*

Exemple 140. *Soient la signature $\Sigma = \{f/2, a/0\}$ et l'automate $\mathcal{A} = (\Sigma, \{q, q_{f(x,y)}, q_f\}, \{q_f\},$*

Δ) *dont les règles sont:*

$$
\begin{array}{rcl}
a & \to & q(a) \\
f(q(x_1), q(x_2)) & \to & q_{f(x,y)}(f(x_1, x_2)) \\
f(q_{f(x,y)}(x_1), q(x_2)) & \xrightarrow{[11=2]} & q_f(f(x_1, x_2)) \\
f(q_{f(x,y)}(x_1), q(x_2)) & \xrightarrow{[11\neq 2]} & q_{f(x,y)}(f(x_1, x_2)) \\
f(q(x_1), q_{f(x,y)}(x_2)) & \to & q_{f(x,y)}((f(x_1, x_2)) \\
f(q_{f(x,y)}(x_1), q_{f(x,y)}(x_2)) & \xrightarrow{[11=2]} & q_f(f(x_1, x_2)) \\
f(q_{f(x,y)}(x_1), q_{f(x,y)}(x_2)) & \xrightarrow{[11\neq 2]} & q_{f(x,y)}(f(x_1, x_2)) \\
f(q_f(x_1), q(x_2)) & \to & q_f(f(x_1, x_2)) \\
f(q(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) \\
f(q_f(x_1), q_{f(x,y)}(x_2)) & \to & q_f(f(x_1, x_2)) \\
f(q_{f(x,y)}(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2)) \\
f(q_f(x_1), q_f(x_2)) & \to & q_f(f(x_1, x_2))
\end{array}
$$

Soient l'ensemble $\mathcal{S} = \{0, 1\}$ *tel que* $0 < 1$ *et la fonction* ω *de* $\mathcal{S}$ *dans* $\mathcal{Q}$ *telle que* $\omega(q_f) = 1$ *et* $\omega(q) = \omega(q_{f(x,y)}) = 0$. *Alors l'automate* $\mathcal{A}$ *est un AR déterministe et complet reconnaissant l'ensemble des termes entourant le terme* $f(f(x, y), x)$.

2.7.2 Expressivité

Tout automate à contraintes diségalitaires est un *AR* puisque seules des inégalités sont testées.

Propriété 141. *Le pouvoir d'expression des AR est supérieur à celui des ACD.*

2.7.3 Déterminisation et complétion

La classe des automates de réduction est close par complétion mais non par déterminisation.

Propriété 142 (Complétion, M. Dauchet *et al.* **[23]).** *Pour tout AR* $\mathcal{A}$, *il existe un AR* $\mathcal{A}'$ *complet tel que* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Remarque 143. *Dans le cas de l'algorithme de complétion de M. Dauchet et al., si* $\mathcal{A}$ *est déterministe alors l'automate complet* $\mathcal{A}'$ *est déterministe.*

Propriété 144 (Déterminisation, M. Dauchet *et al.* **[23]).** *La classe des AR n'est pas close par déterminisation.*

2.7.4 Propriété de clôture

La classe des langages reconnus par les automates de réduction n'est pas close par complément. En effet, l'automate $\mathcal{A}$ défini dans l'exemple 133 est un AR puisqu'aucune contrainte d'égalité apparaît dans ses règles. Donc cet automate reconnaît l'ensemble des termes binaires non équilibrés. Le complément de ce langage est l'ensemble des termes binaires équilibrés. Or on peut montrer que ce langage n'est pas reconnaissable par AR.

Néanmoins, la sous-classe des langages reconnus par les automates de réduction déterministes est close par complément puisque ces automates peuvent être complétés (propriété 142 et remarque 143).

Propriété 145 (Complément, M. Dauchet *et al.* **[23]).** *La classe des langages reconnus par les automates de réduction déterministes est close par complément.*

Cette restriction sur le déterminisme des automates n'est pas nécessaire pour les autres opérations booléennes et donc nous avons la propriété suivante.

Propriété 146 (Union, intersection, M. Dauchet *et al.* **[23]).** *La classe RAR est close par union et intersection.*

2.7.5 Problème du vide

Comme nous l'avons vu ci-dessus, la finitude de S permet de borner le nombre de tests d'égalité réalisés par un automate de réduction le long de toute branche d'un calcul. Cette borne assure la décidabilité du problème du vide.

Propriété 147 (Problème du vide, M. Dauchet *et al.* **[23]).** *Le problème du vide est décidable pour les AR déterministes.*

L'algorithme de décision du problème du vide de M. Dauchet *et al.* [23] s'inspire de la longue preuve de D. Plaisted [70] dont est déduite la décidabilité de la réductibilité inductive.

2.7.6 Applications

M. Dauchet *et al.* [23] montrent que les automates de réduction permettent de généraliser les résultats sur la théorie de la réduction obtenues avec les AAS (propriété 57 et théorème 61).

Tout d'abord, les règles de ces automates suffisent à reconnaître l'ensemble des termes entourant un terme quelconque.

Propriété 148. *Soient Σ une signature et $\mathcal{X}$ un ensemble de variables. Alors si t est un terme de $T(\Sigma, \mathcal{X})$, l'ensemble des termes entourant t est reconnu par AR déterministes et complets.*

L'exemple 140 illustre cette propriété.

Puisque la classe des AR déterministes et complets possèdent de bonnes propriétés (décidabilité du problème du vide (propriété 147) et clotûre par opérations booléennes (propriétés 145 et 146), on a le théorème suivant :

Théorème 149 (M. Dauchet *et al.* **[23]).** *Le problème de satisfiabilité de la théorie de la réduction est décidable.*

2.8 Automates de réduction généralisés

Dans un article de 1994 [15], A.C. Caron, H. Comon, J.L. Coquidé, M. Dauchet et F. Jacquemard introduisent une nouvelle classe d'automates généralisant à la fois les *ATF* (section 2.5) et les *AR* (section 2.7). Plus exactement, pour un calcul de ces automates, appelés automates de réduction généralisés, le nombre de tests d'égalité entre positions frères est quelconque (*ATF*) et le nombre de tests d'égalilité entre positions non frères est borné sur chaque branche (*AR*).

2.8.1 Définition

Définition 150 (*ARG*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate de réduction généralisé *(ARG) est un automate à contraintes (Σ, Q, Q_f, Δ) tel qu'il existe un ensemble partiellement ordonné fini S et une fonction ω de Q vers S telle que pour toute règle de transition $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ de Δ, $\omega(q)$ est un majorant de $\{\omega(q_1), \ldots, \omega(q_n)\}$ dans S et de plus, c'est un majorant strict si c contient des contraintes égalitaires de la forme $\pi = \pi'$ avec $|\pi| > 1$ ou $|\pi'| > 1$.*

Nous désignons par *RARG* la classe des langages d'arbres reconnus par les *ARG*.

2.8.2 Expressivité

De façon évidente, tout *ATF* est un *ARG* et tout *ARG* est un automate à contraintes. D'où les propriétés suivantes.

Propriété 151. *Le pouvoir d'expression des ARG est supérieur à celui des ATF.*

Propriété 152. *Le pouvoir d'expression des AC est supérieur à celui des ARG.*

Enfin tout *AR* est un *ARG*. De plus l'ensemble des termes binaires équilibrés est reconnaissable par *ARG* mais pas par *AR*.

Propriété 153. *Le pouvoir d'expression des ARG est strictement supérieur à celui des AR.*

2.8.3 Problème du vide

La propriété essentielle des *ARG* est de conserver la décidabilité du problème du vide des *ATF* et *AR*.

Propriété 154 (Problème du vide, A.-C. Caron *et al.* [15]). *Le problème du vide est décidable pour les ARG.*

2.8.4 Applications

Les tests et propriétés des *ARG* permettent de combiner les résultats de décidabilité obtenus pour la théorie de la réduction d'une part avec les *AR* (théorème 149) et d'autre part avec les *ATF* (théorème 125).

Théorème 155. *Le problème de satisfiabilité du fragment de la théorie de la réduction avec déclaration de sortes est décidable lorsque les déclarations de sorte $t' \in s$ sont telles que t' est semi-linéaire.*

2.9 Automates à tests d'égalité entre cousins

Nous avons dans les sections précédentes que le problème du vide est indécidable pour les AC (propriété 88) mais qu'il devient décidable lorsque l'on se limite à des tests d'égalité entre positions frères (propriété 136).

Nous pouvions donc espérer conserver cette décidabilité en limitant les tests d'égalité à des termes de même profondeur. Pourtant, M. Tommasi [86] montre que le problème du vide est indécidable dès que l'on ne borne pas le nombre de tests d'égalité autorisés entre termes cousins (sous-termes à la profondeur 2). Dans le chapitre **Tests d'égalité entre cousins** de la partie **Automates à contraintes** nous établissons une nouvelle preuve de ce résultat.

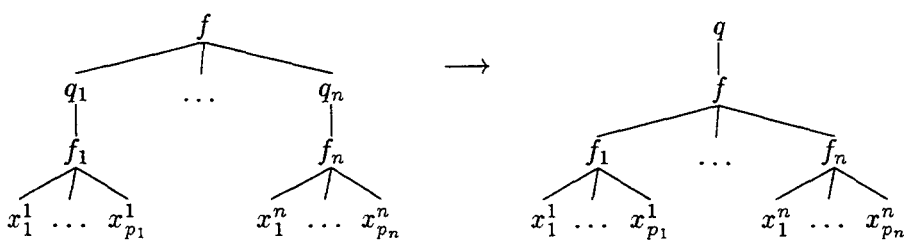
Dans cette section, nous définissons les automates ayant de tels tests et nous les appelons automates à tests d'égalités entre cousins.

2.9.1 Définition

Définition 156 (*ATEC*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres à tests d'égalité entre cousins *(ATEC) est un quadruplet $(\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini d'états ($\Sigma \cap \mathcal{Q} = \emptyset$);*
3. *$\mathcal{Q}_f \subseteq \mathcal{Q}$ est l'ensemble des états finaux;*
4. *Δ est un système de réécriture dont les règles sont de la forme :*
   (a) *$a \to q(a)$ où a est une constante de Σ et q un état de $\mathcal{Q}$;*
   (b) *$f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ où f est un symbole de Σ d'arité n supérieure ou égale à un, $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$ et $x_1, \ldots, x_n$ des variables distinctes de $\mathcal{X}$;*
   (c) *$f(q_1(u_1), \ldots, q_n(u_n)) \to q(f(u_1, \ldots, u_n))$ où pour tout i de $[n]$, $u_i = f_i(x^i_1, \ldots, x^i_{p_i}))$ avec $f, f_1, \ldots, f_n$ symboles de Σ d'arité respective $n, p_1, \ldots, p_n$ ($n \leq 1$), $q_1, \ldots, q_n, q$ des états de $\mathcal{Q}$ et les $x^k_{i_j}$ des variables de $\mathcal{X}$.*

La troisième forme de règles se représente plus facilement sous forme d'arbre :



Remarque 157. *Seules les règles de la troisième forme peuvent être non linéaires ce qui permet d'imposer uniquement des égalités entre termes cousins.*

Nous désignons par *RATEC* la classe des langages d'arbres reconnus par les *ATEC*.

2.9.2 Expressivité

A tout *ATEF*, on peut associer un *ATEC* reconnaissant le même langage. En effet, on remplace toute règle $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ par l'ensemble des règles de la troisième forme pour tout n-uplet $u_1, \ldots, u_n$ tel que :

- Pour tout entier i de $[n]$, u_i soit une constante de Σ ou un terme de la forme $f_i(x^i_1, \ldots, x^i_{p_i})$ avec f_i symbole de Σ d'arité p_i et $x^k_{i_j}$ variables distinctes de $\mathcal{X}$, et
- Pour tous entiers i, j de $[n]$, $u_i = u_j$ si $x_i = x_j$.

Donc le pouvoir d'expression des *ATEC* est supérieur à celui *ATEF*. Nous prouvons ci-dessous que la classe *RATEF* est strictement incluse dans la classe *RATEC*.

Propriété 158. *Le pouvoir d'expression des ATEC est strictement supérieur à celui ATEF.*

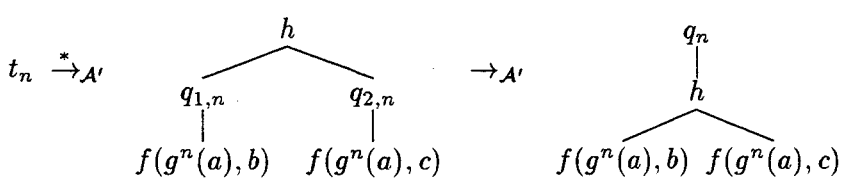
Preuve : Soit l'automate $\mathcal{A} = (\{a/0, b/0, c/0, g/1, f/2, h/2\}, \{q_a, q_b, q_c, q_1, q_2, q_f\}, \{q_f\}, \Delta)$ avec Δ composé des règles de transition suivantes :

$$\begin{aligned}
a &\to q_a(a)\\
b &\to q_b(b)\\
c &\to q_c(c)\\
g(q_a(x)) &\to q_a(g(x))\\
f(q_a(x), q_b(y)) &\to q_1(f(x, y))\\
f(q_a(x), q_c(y)) &\to q_2(f(x, y))\\
h(q_1(f(x, y)), q_2(f(x, z))) &\to q_f(h(f(x, y), f(x, z)))
\end{aligned}$$

Alors $\mathcal{A}$ est un *ATEC* reconnaissant le langage suivant :

$$\mathcal{L} = \{t \in T(\Sigma) \mid t = h(f(t_1, t_2), f(t_1, t_3)) \text{ avec } t_1 \in T(\{g, a\}) \text{ et } t_2, t_3 \in \{b, c\}\}.$$

Supposons que $\mathcal{L}$ soit un langage de la classe *RATEF*. Alors il existe un automate à tests d'égalité entre frères $\mathcal{A}' = (\{a/0, b/0, c/0, g/1, f/2, h/2\}, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ reconnaissant $\mathcal{L}$. Pour tout entier n, nous notons t_n le terme $h(f(g^n(a), b), f(g^n(a), c))$. Alors puisque t_n appartient à $\mathcal{L}$, il existe un état final q_n de $\mathcal{Q}_f$ et deux états $q_{1,n}, q_{2,n}$ de $\mathcal{Q}$ tels que :



Soient m et n deux entier distincts ($m \neq n$). Puisque t_m et t_n appartiennent à $\mathcal{L}$, nous avons $f(g^m(a), b) \xrightarrow{*}_{\mathcal{A}'} q_{1,m}$ et $f(g^n(a), c) \xrightarrow{*}_{\mathcal{A}'} q_{2,n}$. Supposons que $q_{1,m} = q_{1,n}$, alors

$$h(f(g^m(a), b), f(g^n(a), c)) \xrightarrow{*}_{\mathcal{A}'} q_n(h(f(g^m(a), b), f(g^n(a), c))).$$

Donc le terme $h(f(g^m(a), b), f(g^n(a), c))$ appartient à $\mathcal{L}$. On aboutit à une contradiction d'où $q_{1,m} \neq q_{1,n}$. Les états de la famille $(q_{1,n})_{n \in \mathbb{N}}$ sont donc deux à deux distincts ce qui contredit la finitude de l'ensemble des états $\mathcal{Q}$. Nous en déduisons que $\mathcal{L}$ n'est pas un langage de la classe *RATEF*. □

Enfin tout *ATEC* est de façon évidente un *RATEG*.

Propriété 159. *Le pouvoir d'expression des RATEG est supérieur à celui des ATEC.*

2.9.3 Propriétés de clôture

Propriété 160 (Union, intersection). *La classe RATEC est close par union et intersection.*

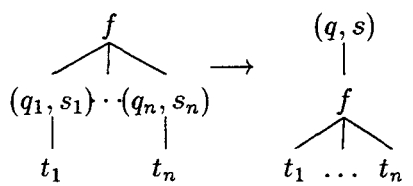
Preuve : Soient Σ une signature, $\mathcal{X}$ un ensemble de variables et, $\mathcal{A}_1 = (\Sigma, \mathcal{Q}_1, \mathcal{Q}_{f_1}, \Delta_1)$ et $\mathcal{A}_2 = (\Sigma, \mathcal{Q}_2, \mathcal{Q}_{f_2}, \Delta_2)$ deux *ATEC*. Sans perte de généralité, nous pouvons supposer que $\mathcal{Q}_1$ et $\mathcal{Q}_2$ sont disjoints.

Soit l'automate $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ avec $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, $\mathcal{Q}_f = \mathcal{Q}_{f_1} \cup \mathcal{Q}_{f_2}$ et $\Delta = \Delta_1 \cup \Delta_2$. $\mathcal{A}$ est un *ATEC* et nous avons de façon évidente $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$. Donc la classe *RATEC* est close par union.

Montrons maintenant que la classe *RATEC* est close par intersection. Nous désignons par π_1 (respectivement π_2) la projection de $\mathcal{Q}_1 \times \mathcal{Q}_2$ sur $\mathcal{Q}_1$ (respectivement $\mathcal{Q}_2$) et nous considérons l'automate $\mathcal{A}' = (\Sigma, \mathcal{Q}', \mathcal{Q}'_f, \Delta')$ défini par :

- $\mathcal{Q}' = \mathcal{Q}_1 \times \mathcal{Q}_2$,
- $\mathcal{Q}'_f = \{q \in \mathcal{Q}' \mid \forall i \in [2], \pi_i(q) \in \mathcal{Q}_{f_i}\}$, et
- Δ' défini par :

  Si $f(q_1(u_1), \ldots, q_n(u_n)) \to q(f(u_1, \ldots, u_n))$ est une règle de Δ_1 et $f(s_1(v_1), \ldots, s_n(v_n)) \to s(f(v_1, \ldots, v_n))$ une règle de Δ_2 alors la règle suivante appartient à Δ :



avec $f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n)\sigma$ où σ est l'un des unificateurs les plus généraux de $f(u_1, \ldots, u_n)$ et $f(v_1, \ldots, v_n)$.

Alors nous pouvons montrer que pour tout terme t de $T(\Sigma)$,

$$t \to_{\mathcal{A}'} q(t) \Leftrightarrow \forall i \in [2], t \to_{\mathcal{A}_i} \pi_i(q)(t).$$

Nous en déduisons que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$ d'où la clôture par intersection de la classe *RATEC*. □

2.9.4 Problème du vide

Propriété 161 (Problème du vide, M. Tommasi [86]). *Le problème du vide est indécidable pour les ATEC.*

Chapitre 3

Méthode des Arbres-Grilles

L'utilisation des grilles pour coder les calculs des machines de Turing afin de prouver des résultats d'indécidabilité est classique [50, 95, 39, 58, 93]. Nous adaptons cette méthode aux arbres en codant toute grille finie en un arbre. Nous nommons la méthode ainsi obtenue la méthode des Arbres-Grilles.

Tout d'abord, nous faisons un rappel sur les machines de Turing (section 3.1) puis nous définissons la méthode des Arbres-Grilles (section 3.2).

3.1 Machine de Turing

Les machines de Turing sont un outil classique pour la démonstration de l'indécidabilité de problèmes. Elles ont été introduites par A.M. Turing [91] en 1936 et constituent l'un des modèles les plus standards de calculateurs.

Dans un premier temps, nous rappelons les principales définitions relatives aux machines de Turing (section 3.1.1). Nous suivons pour cela la présentation des machines de Turing de J.E. Hopcroft et J.D. Ullman [43]. Ensuite nous rappelons certains résultats d'indécidabilité liés au problème de l'arrêt de ces machines (section 3.1.2).

3.1.1 Définitions

Plusieurs variantes des spécifications des machines de Turing existent dans la littérature, le modèle que nous considérons [43] (voir illustration en figure 1) est déterministe et consiste en une unité de contrôle finie contenant un état, une bande divisée en cellules et une tête scrutant une cellule de la bande à un instant donné. La bande est bornée à gauche et infinie à droite. Chaque cellule de la bande contient un symbole d'un alphabet fini dit de bande.

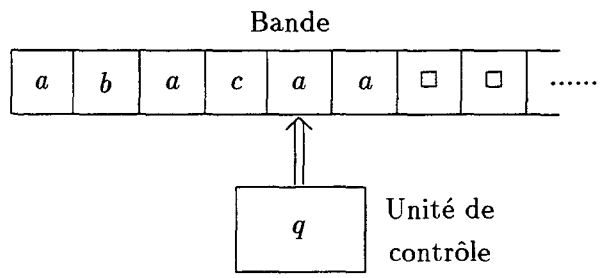
Initialement la tête scrute la cellule de la bande la plus à gauche et l'unité de contrôle est dans un état dit initial. De plus un mot d'entrée constitué de symboles pris dans un sous-ensemble de l'alphabet de bande est stocké sur la bande en partant de la cellule la plus à gauche, toutes les autres cellules contenant un symbole particulier appelé blanc (symbole de l'alphabet de bande n'appartenant pas à l'alphabet d'entrée).

Une machine de Turing exécute des mouvements. Chaque mouvement dépend du contenu de la cellule scrutée par la tête et de l'état contenu par l'unité de contrôle. Dans un mouvement, une machine de Turing

1. change d'état,

2. remplace le symbole de la cellule scrutée par un nouveau symbole, et
3. soit garde la même position pour sa tête, soit la modifie en la déplaçant d'une cellule vers la gauche ou la droite.

FIG. 1 - Une machine de Turing.



Maintenant que nous avons donné une idée du fonctionnement d'une machine de Turing, nous en donnons une définition formelle.

Définition 162. *Une* machine de Turing *déterministe est un septuple* $\mathcal{M} = (\mathcal{Q}, \Sigma, \Gamma, \delta, q_0, \square, \mathcal{F})$ *où :*

- $\mathcal{Q}$ *est un ensemble fini d'états;*
- $\Sigma \subseteq (\Gamma \setminus \{\square\})$ *est l'alphabet d'entrée;*
- Γ *est l'alphabet de bande (ensemble fini de constantes);*
- $\square$ *appartient à* Γ *et est le blanc;*
- q_0 *appartient à* $\mathcal{Q}$ *et est l'état initial;*
- $\mathcal{F} \subseteq \mathcal{Q}$ *est l'ensemble des états finaux;*
- δ *est la fonction de transition (ou ensemble des règles de transition) :* $\delta : \mathcal{Q} \times \Gamma \to \mathcal{Q} \times \Gamma \times \{L, 0, R\}$.

Considérons une machine de Turing déterministe $\mathcal{M} = (\mathcal{Q}, \Sigma, \Gamma, \delta, q_0, \square, \mathcal{F})$. Une règle de transition $\delta(q, x) \to (q', x', d)$ de $\mathcal{M}$ s'interprète de la façon suivante : si $\mathcal{M}$ est dans l'état q et la cellule scrutée par la tête contient le symbole x, alors $\mathcal{M}$ passe dans l'état q', stocke x' à la place de x dans la cellule scrutée et suivant la valeur de d modifie ou pas la position de sa tête. Si $d = 0$ la tête ne bouge pas, si $d = L$ la tête se déplace d'une cellule vers la gauche et si $d = R$ d'une cellule vers la droite.

L'état global de $\mathcal{M}$ à un instant donné est décrit par un mot wqw' appelé *configuration* ou *description instantanée* où :

- q est l'état de la machine,

- ww' est un mot de Γ^* constitué du contenu de la bande c'est-à-dire :
  - Le premier symbole de ww' est le contenu de la cellule la plus à gauche de la bande,
  - ww' contient le symbole non blanc le plus à droite de la bande lorsque celui-ci existe, et
  - Le premier symbole du mot w' est le contenu de la cellule scrutée par la tête.

Remarque 163. *A un instant donné, une infinité de configurations sont possibles, celles-ci ne différant que par le nombre de blancs se trouvant à l'extrême droite du mot w'.*

Une configuration q_0w' est dite *initiale* si w' appartient à $\Sigma^*\{\square\}^*$ et une configuration wqw' est dite *finale* si q est un état final de $\mathcal{F}$.

Un *mouvement* de $\mathcal{M}$ consiste en une paire de configurations (C, C') où C' est obtenue à partir de C par application d'une règle de transition de δ. Un mouvement entre C et C' est noté $C \vdash_{\mathcal{M}} C'$. Nous notons $\overset{*}{\vdash}_{\mathcal{M}}$ la clôture réflexive et transitive de $\vdash_{\mathcal{M}}$, donc $C \overset{*}{\vdash}_{\mathcal{M}} C'$ si $C = C'$ (pas de mouvement) ou si la configuration C' est obtenue à partir de la configuration C par application d'un nombre fini de mouvements.

Une suite finie de configurations $C_1, \ldots, C_k$ telle que pour tout entier i de $[k-1], C_i \vdash_{\mathcal{M}} C_{i+1}$, est appelée

- Un *calcul* de la machine de Turing $\mathcal{M}$ si la configuration C_1 est initiale,
- Un calcul *débutant sur w* de $\mathcal{M}$ si $C_1 = q_0w$, et
- Un *calcul réussi* de $\mathcal{M}$ si C_1 est initiale et C_k finale.

3.1.2 Problème de l'arrêt

L'indécidabilité du problème de l'arrêt des machines de Turing est un résultat qui revient souvent dans les preuves d'indécidabilité. En particulier ce résultat a permis à E.L. Post [73] de prouver l'indécidabilité du problème des mots des systèmes semi-Thueins. En 1953, H.G. Rice [75] montre que toute propriété non triviale des langages récursivement énumérables est indécidable (Théorème de Rice). En 1967, D. Scott [81] établit une nouvelle preuve de l'indécidabilité du problème de correspondance de Post (la preuve initiale étant due à E.L. Post [72]). L'indécidabilité du problème de l'arrêt des machines de Turing a permis également de montrer l'indécidabilité de plusieurs variantes du problème du pavage d'un plan par des dominos.

Avant d'énoncer le problème de l'arrêt des machines de Turing et de l'une de ses variantes appelée problème de l'arrêt sur l'entrée vide, nous définissons la notion d'*arrêt*. Soit $\mathcal{M} = (\mathcal{Q}, \Sigma, \Gamma, \delta, q_0, \square, \mathcal{F})$ une machine de Turing. $\mathcal{M}$ s'arrête sur l'entrée w, w mot non vide de $\Sigma^*\{\square\}^*$, si $q_0w \overset{*}{\vdash}_{\mathcal{M}} w_1q'w_2$ avec q' état final de $\mathcal{F}$ et $\mathcal{M}$ s'arrête sur l'*entrée vide* s'il existe un calcul réussi pour $\mathcal{M}$ à partir d'une bande ne contenant que des blancs.

Problème de l'arrêt
Entrée : *Une machine de Turing $\mathcal{M}$ et un mot d'entrée w.*
Question : *La machine $\mathcal{M}$ s'arrête-t-elle sur l'entrée w ?*

Problème de l'arrêt sur l'entrée vide
Entrée : *Une machine de Turing $\mathcal{M}$.*
Question : *La machine $\mathcal{M}$ s'arrête-t-elle sur l'entrée vide?*

Ces deux problèmes sont indécidables (A.M. Turing [91]). Ces résultats sont conservés si l'on considére la classe des machine de Turing ne possédant qu'un seul état final et dont l'alphabet d'entrée est constitué de deux symboles et l'alphabet de bande de trois symboles (donc l'alphabet de bande est l'union de l'alphabet d'entrée et du singleton contenant le symbole blanc).

Cette remarque nous conduit à la définition d'une classe dite restreinte de machine de Turing (définition 164) utilisée dans la section suivante et dans laquelle nous pouvons montrer que les deux problèmes d'arrêt précédents restent indécidables (théorèmes 165 et 166).

Définition 164. *Une machine de Turing $\mathcal{M}$ déterministe est dite* restreinte *si:*

- *$\mathcal{M}$ est de la forme $(Q, \{a, b\}, \{a, b, \square\}, \delta, q_0, \square, \{q_f\})$;*
- *Aucun blanc n'est écrit sur la bande par application d'une règle de transition (donc le symbole blanc n'apparaît jamais à gauche de la tête de lecture);*
- *q_0 n'apparaît pas en partie droite des règles.*

Théorème 165. *Le problème de l'arrêt pour les machines de Turing déterministes restreintes est indécidable.*

Théorème 166. *Le problème de l'arrêt sur l'entrée vide pour les machines de Turing déterministes restreintes est indécidable.*

3.2 Méthode de la grille

La méthode de la grille est un outil classique dans les preuves d'indécidabilité puisqu'elle fournit un moyen pratique pour coder les calculs des machines de Turing. Dans le codage d'un calcul, chaque ligne de la grille contient une configuration de la machine de Turing considérée et deux lignes successives de la grille correspondent à un mouvement. En fait, une cellule (m, n) de la grille contient la $n^{\text{ième}}$ valeur de la $m^{\text{ième}}$ configuration du calcul considéré. L'avantage de cette structure réside en la possibilité de vérifier uniquement par des tests locaux que les lignes successives d'une grille correspondent aux configurations successives d'un calcul réussi de la machine de Turing.

Ce codage permet de déduire de l'indécidabilité de l'arrêt de la machine de Turing d'autres résultats importants d'indécidabilité. Par exemple, ce codage est utilisé dans la preuve de l'indécidabilité de plusieurs variantes du problème de pavage d'un plan par des dominos. Par la suite, ces résultats ont permis de prouver l'indécidabilité de plusieurs sous-classes du calcul des prédicats [39, 50, 58, 95] et l'indécidabilité de la solvabilité des équations diophantiennes exponentielles [93]. W. Thomas [85] utilise ce même codage de la machine du Turing pour démontrer l'indécidabilité de la théorie monadique du second ordre faible de la grille. De plus amples informations sur la méthode de la grille se trouvent dans le livre de E. Börger, E. Grädel et Y. Gurevich [12].

Dans la suite de cette section, nous adaptons la méthode de la grille aux arbres. Nous obtenons ainsi un outil de preuves d'indécidabilité, appelé méthode des Arbres-Grilles, que

nous utilisons dans la suite de ce document. En fait nous associons à chaque grillee fini un arbre. Donc nous pouvons coder chaque calcul d'une machine de Turing en un arbre. En fait la méthode que nous utilisons lorsque nous faisons appel à la méthode des Arbres-Grilles est la suivante.

Méthode 167. *Nous exprimons deux propriétés: d'une part qu'un arbre est le codage d'une grille et d'autre part que la grille associée à cet arbre code un calcul réussi d'une machine de Turing.*

L'étape consistant à vérifier si une grille code un calcul réussi d'une machine de Turing peut s'effectuer par des tests locaux sur la grille (appartenance et exclusion de motifs particuliers). Ces tests sont réalisables par des outils tels que les automates d'arbres, la réécriture et les contraintes ensemblistes, ce qui nous permet de montrer que les théories suivantes sont indécidables :

- Le problème du vide est indécidable pour les automates à tests d'égalité entre cousins (chapitre **Tests d'égalités entre cousins** de la partie **Automates d'arbres à contraintes**);
- Le fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas pour la classe des systèmes de réécriture linéaires, minces et convergents (chapitre **Indécidabilité** de la partie **Réécriture**);
- Le fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules contenant pour seul opérateur ensembliste l'union (chapitre **Indécidabilité** de la partie **Contraintes ensemblistes**).

Dans un premier temps, nous présentons les grilles finies (section 3.2.1). Puis nous associons à chaque calcul d'une machine de Turing une grille (section 3.2.2) et finalement à chaque grille un arbre particulier appelé G-Arbre (section 3.2.3). Nous en déduisons la proposition 177 à laquelle nous faisons appel pour démontrer les trois résultats d'indécidabilité cités ci-dessus.

3.2.1 Grilles finies

Définition 168. *Une* grille finie *sur un ensemble de symboles Σ est une fonction $g : D_g \to \Sigma$ dont le* domaine D_g *est un sous-ensemble de $\mathbb{N}_+ \times \mathbb{N}_+$ satisfaisant les propriétés suivantes:*

1. $\forall (u+1, v) \in D_g$, $(u, v) \in D_g$;
2. $\forall (u, v+1) \in D_g$, $(u, v) \in D_g$;
3. $\forall (u, v) \in D_g$, $(u+1, v+1) \in D_g$ *s'il existe* w *tel que* $(u+1, w) \in D_g$.

Remarque 169. *Toutes les grilles que nous considérons dans la suite de ce document sont finies donc nous omettons à partir de maintenant l'adjectif "finie".*

Toute grille g se représente par un quadrillage bidimentionnel q dont les cellules contiennent les différentes valeurs prises par g. En fait, la cellule de q se trouvant au croisement de la $m^{\text{ième}}$

ligne et de la $n^{\text{ième}}$ colonne est désignée par le couple (m, n) et contient la valeur $g(m, n)$ de la grille, l'origine $(1, 1)$ du quadrillage étant situé en son coin inférieur gauche.

Par définition du domaine des grilles, chaque ligne (respectivement chaque colonne) du quadrillage q commence au bord gauche (respectivement au bord inférieur) du plan et est constitué de cellules pleines (conditions 1 et 2 de la définition 168). De plus par la condition 3 de la définition 168, la longueur des lignes croit strictement lorsque l'on parcourt le quadrillage de bas en haut (exemple figure 2).

FIG. 2 - Une grille sur $\Sigma = \{a, b, c\}$ et son quadrillage.

g(1,1)	=	g(2,4)	=	a
g(2,1)	=	g(2,2)	=	b
g(1,2)	=	g(2,3)	=	c

b	*b*	*c*	*a*
a	*c*		

Dans la suite, nous ne faisons plus la distinction entre une grille et sa représentation en quadrillage, et nous désignons par $lig_g(m)$ (respectivement $col_g(n)$) la $m^{\text{ième}}$ ligne (respectivement la $n^{\text{ième}}$ colonne) d'une grille g :

$$\begin{aligned} lig_g(m) &= \{(m, l) \mid (m, l) \in D_g\} \\ col_g(n) &= \{(k, n) \mid (k, n) \in D_g\} \end{aligned}$$

3.2.2 Machine de Turing et grille

Les grilles étant définies, nous montrons dans cette section que l'on peut coder tout calcul d'une machine de Turing en une grille. Nous considérons une instance $\mathcal{M}$ de la classe restreinte des machines de Turing déterministe (définition 164). Supposons que $\mathcal{M} = (\mathcal{Q}, \{a, b\}, \{a, b, \square\}, \delta, q_s, \square, q_f)$ avec $\mathcal{Q} = \{q_1, \ldots, q_k\}$.

La signature Σ ainsi que plusieurs autres entités construites par la suite dépendent de la machine de Turing $\mathcal{M}$ mais par souci de clarté, nous allons omettre dans les notations l'indice $\mathcal{M}$. Pour des raisons techniques, les symboles de la bande situés à gauche de la tête seront indexés par la lettre g pour gauche et les autres symboles de la bande par d pour droite. Nous obtenons ainsi pour alphabet de bande l'ensemble de constantes *ConstantesMT* défini ci-dessous.

Définition 170. *(Constantes) Soit l'ensemble ConstantesMT de constantes suivant:*

$$\begin{aligned} SymbGauches &:= \{a_l, b_l\} \\ SymbDroits &:= \{a_r, b_r, \square_r\} \\ Etats &:= \{q_1, \ldots, q_k\} \\ ConstantesMT &:= SymbGauches \cup SymbDroits \cup Etats \end{aligned}$$

Remarquons que le symbole $\square_l$ n'existe pas puisqu'aucun blanc ne peut apparaître à gauche de la tête de la machine (restriction de la définition 164).

Chaque configuration CF de la machine de Turing $\mathcal{M}$ est un mot sur l'alphabet *ConstantesMT*, et plus exactement un mot de l'expression régulière suivante :

$$CF := SymbGauches^* \; Etats \; SymbDroits^+.$$

Et chaque calcul de la machine $\mathcal{M}$ est une suite finie de mots de l'expression régulière précédente.

Nous considérons maintenant l'ensemble des grilles construites sur l'alphabet *ConstantesMT*. Toute grille g représente une suite finie de mots de $ConstantesMT^*$ où $g(m,n)$ est la $n^{\text{ième}}$ lettre du $m^{\text{ième}}$ mot de la suite. Bien évidemment, toute grille g ne code pas un calcul de la machine de Turing $\mathcal{M}$ et nous allons donc définir dans quelles conditions cela est le cas. Par la suite, nous ne faisons plus la distinction entre les configurations qui ne différent que par le nombre de symboles blanc se trouvant à leur extrême droite (exemple 171).

Exemple 171. *Les configurations* $a_l a_l q b_r a_r \square_r$ *et* $a_l a_l q b_r a_r \square_r \square_r \square_r \square_r$ *seront utilisées indifféremment.*

Nous définissons maintenant un ensemble P_T de motifs bidimentionnels qui nous permet de détailler les conditions que doit remplir une grille pour représenter un calcul de la machine $\mathcal{M}$.

Définition 172 (P_T). *L'ensemble P_T est l'ensemble des motifs décrit en figure 3 où x, y, z sont des éléments quelconques de ConstantesMT, c_l, d_l des éléments quelconques de SymbGauches, c_r, d_r des éléments quelconques de SymbDroits et _ un élément quelconque de ConstantesMT.*

Remarquons que P_T est un ensemble fini puisque l'ensemble *ConstantesMT* est fini.

Dans la proposition suivante, nous établissons une relation entre grille et calcul.

Proposition 173. *Soit une grille g dont la première ligne est la configuration initiale $q_s \square_r \square_r$. Alors g représente un calcul de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide si et seulement si aucun des motifs de P_T apparaît dans g.*

Preuve : Soit une grille g dont la première ligne est la configuration initiale $q_s \square_r \square_r$. Les deux blancs assurent que si aucun des motifs de P_T apparaît dans g, chaque ligne de g est une configuration de la machine de Turing $\mathcal{M}$.

Supposons dans un premier temps qu'au moins un des motifs de P_T apparaisse dans g. Si le premier motif de la figure 3 apparaît dans g, un symbole autre que le blanc apparaît à droite d'un blanc ce qui traduit le fait qu'un blanc est écrit sur la bande par la tête de la machine $\mathcal{M}$ ce qui contredit la définition des machines de Turing restreintes (définition 164). Si l'un des deux motifs suivants de la figure 3 apparaît dans g, un caractère gauche (*SymbGauches*), respectivement un caractère droit (*SymbDroits*), est modifié alors que la cellule de droite, respectivement de gauche, ne contient pas un état (*Etats*). Si l'un des autres motifs de la figure 3 apparaît dans g, les symboles de la bande sont modifiés sans application d'une règle de la machine de Turing $\mathcal{M}$. Donc dans tous les cas, g n'est pas le codage d'un calcul de la machine $\mathcal{M}$. Donc si g représente un calcul de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide alors aucun des motifs de P_T apparaît dans g.

FIG. 3 - Ensemble P_T de motifs.

$\begin{array}{|c|}\hline \Box_r \\ \hline x \\ \hline \end{array}$ où $x \neq \Box_r$

$\begin{array}{|c|c|}\hline x & - \\ \hline c_l & d_l \\ \hline \end{array}$ où $x \neq c_l$

$\begin{array}{|c|c|}\hline - & x \\ \hline c_r & d_r \\ \hline \end{array}$ où $x \neq d_r$

Si $(q, d) \mapsto (p, e, L) \in \delta$:

$\begin{array}{|c|c|c|}\hline x & y & z \\ \hline c_l & q & d_r \\ \hline \end{array}$ où $(x \neq p$ ou $y \neq c_r$ ou $z \neq e_r)$

Si $(q, d) \mapsto (p, e, 0) \in \delta$:

$\begin{array}{|c|c|c|}\hline x & - & - \\ \hline c_l & q & d_r \\ \hline \end{array}$ où $x \neq c_l$ $\qquad$ $\begin{array}{|c|c|}\hline x & y \\ \hline q & d_r \\ \hline \end{array}$ où $(x \neq p$ ou $y \neq e_r)$

Si $(q, d) \mapsto (p, e, R) \in \delta$:

$\begin{array}{|c|c|c|}\hline x & - & - \\ \hline c_l & q & d_r \\ \hline \end{array}$ où $x \neq c_l$ $\qquad$ $\begin{array}{|c|c|}\hline x & y \\ \hline q & d_r \\ \hline \end{array}$ où $(x \neq e_l$ ou $y \neq p)$

Supposons maintenant qu'aucun des motifs de P_T apparaisse dans g. Montrons par induction sur le nombre m de lignes de g que la ligne $lig_g(m)$ se termine par deux blancs et que g représente un calcul de la machine $\mathcal{M}$ débutant sur l'entrée vide.

Base ($m = 1$). Soit g une grille dont la seule ligne est la configuration initiale $q_s\Box_r\Box_r$. De façon évidente, la ligne $lig_g(m)$ se termine par deux blancs et g représente un calcul de la machine $\mathcal{M}$ débutant sur l'entrée vide.

Induction. Soit m un entier strictement positif. Supposons que l'hypothèse d'induction soit vraie pour toute grille dont le nombre de lignes est inférieur ou égal à m. Soit g une grille constituée de $m+1$ lignes dont la première ligne est la configuration initiale $q_s\Box_r\Box_r$ et dans laquelle aucun des motifs de P_T apparaît. Considérons la grille g' définie par :

- $D_{g'} = D_g \setminus \{(m+1, v) \mid (m+1, v) \in D_g\}$,
- $\forall (u, v) \in D_{g'}$, $g'(u, v) = g(u, v)$.

Alors g' est une grille de m lignes dont la première ligne est la configuration initiale $q_s\Box_r\Box_r$ et dans laquelle aucun des motifs de P_T apparaît. Donc par hypothèse d'induction, la ligne $lig_{g'}(m)$ se termine par deux blancs et g' représente un calcul de la machine $\mathcal{M}$ débutant sur l'entrée vide.

Montrons maintenant que le passage de la ligne $lig_g(m)$ à la ligne $lig_g(m+1)$ constitue un mouvement de la machine $\mathcal{M}$.

Soit n le nombre de colonnes de la grille g'. Alors $g(m,n)$ est le dernier symbole de la ligne $lig_g(m)$. L'exclusion du troisième motif de P_T nous permet d'affirmer que $g(m+1,n) = \square_r$ puisque $g(m,n-1) = g(m,n) = \square_r$. Par définition de la structure de la grille (condition 3 de la définition 168), $\{l \in \mathbb{N} \mid l > n \text{ et } (m+1,l) \in D_g\}$ est non vide. Nous déduisons de l'exclusion du premier motif de P_T que pour tout entier l de l'ensemble précédent, on a $g(m+1,l) = \square_r$. Finalement la ligne $lig_g(m+1)$ se termine par deux blancs.

De plus la ligne $lig_g(m)$ représente une configuration de la machine $\mathcal{M}$ car g' représente un calcul de cette machine. Nous pouvons alors montrer que la ligne $lig_g(m+1)$ de g représente une configuration obtenue à partir de la configuration représentée par la ligne $lig_g(m)$ par application d'une règle de transition de $\mathcal{M}$ puisqu'aucun motif de P_T apparaît dans g.

Finalement g représente bien un calcul de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide. □

Nous venons de définir une bijection entre l'ensemble des calculs de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide et un ensemble de grilles particulières. Dans la section suivante, nous définissons un ensemble d'arbres particuliers pour représenter les grilles et nous en déduisons un codage des calculs réussis de la machine $\mathcal{M}$ dans ces arbres.

3.2.3 Grille et Arbre

Pour représenter une grille, on peut utiliser un ensemble de symboles de fonction binaires constitué des symboles de la machine de Turing $\mathcal{M}$. Mais pour la simplification des preuves, nous utilisons un symbole ternaire, l'ensemble des constantes constituant les symboles de $\mathcal{M}$ et une nouvelle constante $\perp_0$. Nous notons Γ la nouvelle signature ainsi définie.

Définition 174 (Signature Γ). *La signature Γ contient les symboles de fonction suivants:*

$$\begin{aligned} f &: \textit{ternaire} \\ \perp_0 &: \textit{constante} \\ a_l, b_l, a_r, b_r, \square_r, q_1, \ldots, q_k &: \textit{constante} \end{aligned}$$

Sur cette signature, nous définissons la notion de G-Arbres, arbres représentant les grilles.

Définition 175 (G-Arbre). *Un terme clos t d'une signature contenant Γ est un* G-Arbre sur Γ si:

$$t \in T(\Gamma) \qquad \text{(PURETÉ)}$$

Pour tout sous-arbre $f(x,y,z)$ de t:

$$x = \perp_0 \text{ ou } tête(x) = f \qquad \text{(FILS-1)}$$
$$y = \perp_0 \text{ ou } tête(y) = f \qquad \text{(FILS-2)}$$
$$z \in ConstantesMT \qquad \text{(CONSTANTES)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), f(x_2, y_2, z_2), z_3)$ de t :

$$y_1 = x_2 \qquad \text{(EGALITÉ)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), y_2, z_2)$ de t :

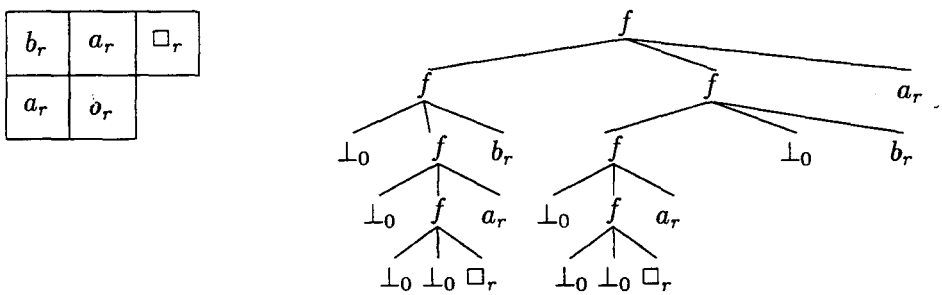
$$tête(y_1) = f \qquad \text{(CORPS-1)}$$

Pour tout sous-arbre $f(x_1, f(f(x_2, y_2, z_2), y_3, z_3), z_1)$ de t :

$$tête(x_1) = f \qquad \text{(CORPS-2)}$$

Tout G-Arbre sur Γ peut être représenté par une grille et inversement. En fait, dans un G-Arbre t, le dernier fils de chaque symbole f est le contenu d'une cellule de la grille, le premier fils le voisin du dessus de cette cellule et le second fils le voisin de droite. Et la "fin" d'une grille (sur une ligne ou une colonne) correspond à une feuille $\perp_0$ de t (exemple figure 4).

FIG. 4 - Une grille de deux lignes et son G-Arbre.



La condition EGALITÉ traduit le fait qu'en faisant à partir d'une cellule de la grille un pas vers le haut puis un pas vers la droite, on aboutit à la même cellule qu'en faisant un pas vers la droite puis un pas vers le haut.

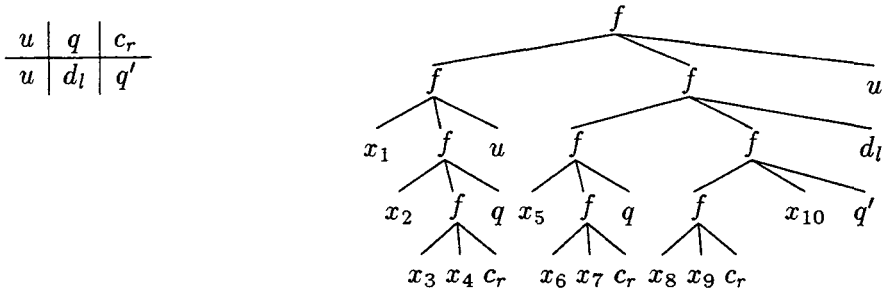
La condition CORPS-1 traduit le fait que toute cellule ayant un voisin du dessus a aussi un voisin supérieur droit. En d'autres termes, pour tout entier m, la $m+1^{\text{ième}}$ ligne de la grille est strictement plus longue que la $m^{\text{ième}}$ ligne (conditions 3 de la définition 168).

Finalement la condition CORPS-2 traduit le fait que toute cellule ayant un voisin supérieur droit à aussi un voisin du dessus. Par conséquent toutes les lignes commencent à la même position (conditions 2 de la définition 168).

Nous pouvons donc associer facilement à toute grille un arbre appelé G-Arbre et vice-versa. Nous pouvons également associer à tout motif d'une grille un contexte de $T(\Gamma, \mathcal{X})$ (contexte) ayant la structure d'un G-Arbre. La construction est la même que pour obtenir un G-Arbre à partir d'une grille, excepté que les symboles $\perp_0$ de fin de ligne et de colonne sont remplacés par des variables de façon à obtenir un arbre linéaire (exemple figure 5 et définition 176).

Définition 176 (P'_T). *Nous associons à tout motif de l'ensemble P_T un contexte de $T(\Gamma, \mathcal{X})$. L'ensemble obtenu est noté P'_T.*

FIG. 5 - Un motif et son contexte associé.



Nous venons de définir une bijection entre l'ensemble des grilles finis et un ensemble d'arbres appelés G-Arbres. Les G-Arbres peuvent donc être utilisés pour coder les calculs de la machine de Turing $\mathcal{M}$ comme le font les grilles. Nous allons maintenant caractériser l'ensemble des G-Arbres codant les calculs réussis de la machine $\mathcal{M}$ débutant sur l'entrée vide.

Proposition 177. *Un G-Arbre t sur Γ code un calcul réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide si et seulement si :*

- *t entoure l'arbre $f(x_1, f(x_2, f(x_3, \perp_0, \square_r), \square_r), q_s)$* (INIT1)
- *t n'entoure pas l'arbre $f(f(x_1, x_2, q_s), x_3, x_4)$* (INIT2)
- *t n'entoure pas l'arbre $f(x_1, f(x_2, x_3, q_s), x_4)$* (INIT3)
- *t n'entoure aucun des motifs de P'_T* (CALCUL)
- *t entoure l'arbre $f(\perp_0, x_1, q_f)$* (FINAL)

Preuve : Soit t un G-Arbre sur Γ. Supposons que t code un calcul r réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide. La configuration initiale de la bande est alors n'importe quel mot débutant par q_s et se prolongeant par un nombre quelconque de blancs $\square_r$. Puisque nous ne faisons pas la distinction entre les configurations qui ne différent que par le nombre de symboles blanc se trouvant à leur extrême droite (exemple 171), nous prenons pour configuration initiale de la bande $q_s\square_r\square_r$. La première ligne de la grille g associée au calcul r est donc $q_s\square_r\square_r$. L'arbre linéaire associé à cette configuration est l'arbre $f(x_1, f(x_2, f(x_3, \perp_0, \square_r), \square_r), q_s)$. Donc la condition INIT1 est satisfaite. De plus comme q_s n'apparaît pas en partie droite des règles de la machine $\mathcal{M}$ (restriction de la définition 164), cet état n'apparaît que dans la cellule se trouvant en bas à gauche de la grille. Or si la condition INIT2 (respectivement la condition INIT3) n'est pas respectée, q_s apparaît sur une ligne de g autre que la première (respectivement dans une colonne de g autre que la première). Donc ces deux conditions sont également satisfaites.

Comme r est un calcul de la machine de Turing $\mathcal{M}$, nous déduisons de la proposition 173 qu'aucun motifs de P_T apparaît dans g d'où la condition CALCUL puisque P'_T est l'ensemble des contextes associés à P_T (définition 176). De plus le calcul r est réussi, donc l'état final q_f apparaît en ligne $lig_g(m)$ de la grille g où m est le nombre de lignes de g. Nous en déduisons la condition FINAL.

Finalement si t est un G-Arbre sur Γ codant un calcul réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide, t vérifie les cinq conditions énoncées dans la proposition 177.

Considérons maintenant un G-Arbre t sur Γ satisfaisant les conditions INIT1, INIT2, INIT3, 173 et FINAL. Soit g la grille associée à t. Les conditions INIT1, INIT2 et INIT3 imposent que la première ligne de g est la configuration initiale $q_s\Box_r\Box_r$.

De plus, nous déduisons de la condition CALCUL qu'aucun motifs de P_T apparaît dans g. Donc d'après la proposition 173, g code un calcul de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide. Il en est donc de même pour t.

Finalement la condition FINAL impose que q_f apparaît en ligne $lig_g(m)$ de la grille g où m est le nombre de lignes de g, et donc le calcul est réussi.

Donc si t est un G-Arbre sur Γ satisfaisant les cinq conditions énoncées dans la proposition 177, t code un calcul réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide. □

Cette proposition constitue le cœur de la méthode des Arbres-Grilles que nous appliquons dans les domaines des automates à contraintes, de la réécriture et des contraintes ensemblistes.

Troisième partie

Automates d'arbres à contraintes

Cette partie, composée de trois chapitres, est consacrée à l'étude des automates d'arbres à contraintes.

Dans le premier chapitre, nous montrons en utilisant la méthode des Arbres-Grilles que le problème du vide est indécidable pour les automates à tests d'égalité entre cousins. Dans le second, nous définissons une classe d'automates à tests d'égalité entre frères et cousins pour laquelle le vide est décidable ce qui nous permet de réduire la frontière entre la décidabilité et l'indécidabilité du problème du vide pour les automates d'arbres à contraintes.

Finalement, dans le dernier chapitre, nous montrons que le problème de reconnaissabilité pour les ATF est décidable c'est-à-dire que pour tout langage $\mathcal{L}$ reconnu par un ATF, on peut décider si $\mathcal{L}$ est régulier.

Chapitre 1

Tests d'égalité entre cousins

Dans le chapitre **Automates d'arbres à contraintes** de la partie **Outils de décidabilité et d'indécidabilité**, nous avons vu que le problème du vide est décidable pour les automates à tests entre frères (*ATF*) ainsi que pour les automates d'entourage généralisés (*ARG*). Ces résultats permettaient d'espérer que le problème du vide restait décidable en effectuant des tests d'égalité entre sous-arbres situés à la même profondeur. Pourtant, M. Tommasi [86] montre que le problème du vide est indécidable dans la classe des automates à tests d'égalité entre cousins (*ATEC*), c'est-à-dire dès que l'on autorise des tests d'égalité entre sous-arbres à la profondeur deux. Dans ce chapitre nous établissons une nouvelle preuve de ce résultat en utilisant la méthode des Arbres-Grilles.

Théorème 178. *Le problème du vide est indécidable pour les automates à tests d'égalité entre cousins.*

Nous rappelons tout d'abord les différents éléments intervenant dans la méthode des Arbres-Grilles que nous avons introduite dans la partie **Outils de décidabilité et d'indécidabilité.**

Soit $\mathcal{M} = (Q, \{a, b\}, \{a, b, \Box\}, \delta, q_s, \Box, q_f)$, avec $Q = \{q_1, \ldots, q_k\}$, une instance de la classe restreinte des machines de Turing déterministes (définition 164). Nous définissons une signature Γ composée des symboles de fonction suivants :

$$\begin{aligned} f &: \textit{ternaire} \\ \bot_0 &: \textit{constante} \\ a_l, b_l, a_r, b_r, \Box_r, q_1, \ldots, q_k &: \textit{constante} \end{aligned}$$

sur laquelle on définit un ensemble d'arbres particuliers, appelés G-Arbres (définition 175), représentant les grilles finies.

Définition 175 (G-Arbre). *Un terme clos t d'une signature contenant Γ est un* G-Arbre sur Γ si :

$$t \in T(\Gamma) \qquad \text{(PURETÉ)}$$

Pour tout sous-arbre $f(x, y, z)$ de t :

$$x = \bot_0 \text{ ou } \textit{tête}(x) = f \qquad \text{(FILS-1)}$$

$$y = \bot_0 \text{ ou } \textit{tête}(y) = f \qquad \text{(FILS-2)}$$

$$z \in \{a_l, b_l, a_r, b_r, \Box_r, q_1, \ldots, q_k\} \qquad \text{(CONSTANTES)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), f(x_2, y_2, z_2), z_3)$ de t :

$$y_1 = x_2 \qquad \text{(EGALITÉ)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), y_2, z_2)$ de t :

$$tête(y_1) = f \qquad \text{(CORPS-1)}$$

Pour tout sous-arbre $f(x_1, f(f(x_2, y_2, z_2), y_3, z_3), z_1)$ de t :

$$tête(x_1) = f \qquad \text{(CORPS-2)}$$

Nous définissons ensuite un ensemble P'_T de contextes de $T(\Gamma, \mathcal{X})$ décelant les arbres qui ne codent pas un calcul de la machine $\mathcal{M}$. Finalement nous montrons qu'on peut caractériser l'ensemble des G-Arbres codant les calculs réussis de la machine $\mathcal{M}$ débutant sur l'entrée vide (proposition 177).

Proposition 177. *Un G-Arbre t sur Γ code un calcul réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide si et seulement si :*

- *t entoure l'arbre $f(x_1, f(x_2, f(x_3, \perp_0, \square_r), \square_r), q_s)$* (INIT1)
- *t n'entoure pas l'arbre $f(f(x_1, x_2, q_s), x_3, x_4)$* (INIT2)
- *t n'entoure pas l'arbre $f(x_1, f(x_2, x_3, q_s), x_4)$* (INIT3)
- *t n'entoure aucun des motifs de P'_T* (CALCUL)
- *t entoure l'arbre $f(\perp_0, x_1, q_f)$* (FINAL)

Ces rappels faits nous donnons l'idée de la preuve du théorème 178. Nous réduisons le problème de l'arrêt de la machine $\mathcal{M}$ dans le problème du vide pour les automates à tests d'égalité entre cousins en utilisant la méthode des Arbres-Grilles. Nous construisons un *ATEC* $\mathcal{A}_{stop}$ reconnaissant l'ensemble des G-Arbres codant un calcul réussi de la machine $\mathcal{M}$ débutant sur l'entrée vide. Le résultat d'indécidabilité de l'arrêt de la machine de Turing sur l'entrée vide (théorème 166) nous permet alors d'énoncer le résultat du théorème 178.

Le langage reconnu par l'automate $\mathcal{A}_{stop}$ est l'intersection de plusieurs langages :

$$\mathcal{L}(\mathcal{A}_{stop}) = \mathcal{L}(\mathcal{A}_{grille}) \cap \mathcal{L}(\mathcal{A}_{init}) \cap \mathcal{L}(\mathcal{A}_{calcul}) \cap \mathcal{L}(\mathcal{A}_{final})$$

où $\mathcal{A}_{grille}$, $\mathcal{A}_{init}$, $\mathcal{A}_{calcul}$ et $\mathcal{A}_{final}$ sont des automates permettant de reconnaître les arbres satisfaisant les conditions de la proposition 177 :

- $\mathcal{A}_{grille}$ reconnaît les arbres qui sont des G-Arbres sur Γ (définition 175);
- $\mathcal{A}_{init}$ reconnaît les arbres satisfaisant les conditions relatives à la configuration initiale (conditions INIT1, INIT2 et INIT3 de la proposition 177);
- $\mathcal{A}_{calcul}$ reconnaît les arbres n'entourant aucun des contextes de P'_T (condition CALCUL de la proposition 177);

- $\mathcal{A}_{final}$ reconnaît les arbres satisfaisant la condition sur l'état final (condition FINAL de la proposition 177).

En fait, $\mathcal{A}_{grille}$ est un automate à tests d'égalité entre cousins dont les tests permettent de reconnaître les arbres satisfaisant la condition EGALITÉ de la définition 175. Les autres automates sont des automates d'arbres standards puisque les tests nécessaires pour vérifier que la grille associée à un G-arbre code un calcul réussi de la machine $\mathcal{M}$ sont locaux et font intervenir des arbres linéaires (arbres associés aux motifs à exclure ou à retrouver dans la grille).

L'idée de la preuve étant donnée, nous montrons dans un premier temps qu'il existe un automate à tests d'égalité entre cousins $\mathcal{A}_{grille}$ reconnaissant l'ensemble des G-Arbres (proposition 179).

Proposition 179. *Il existe un automate à tests d'égalité entre cousins $\mathcal{A}_{grille}$ tel que :*

$$\mathcal{L}(\mathcal{A}_{grille}) = \{t \in T(\Gamma) \mid t \text{ est un G-Arbre sur } \Gamma\}.$$

Preuve : Nous allons construire un *AR* $\mathcal{A}_{gl}$ reconnaissant l'ensemble des termes de $T(\Gamma)$ satisfaisant les conditions PURETÉ, FILS-1, FILS-2, CONSTANTES, CORPS-1 et CORPS-2 de la définition 175 et un *ATEC* $\mathcal{A}_g$ reconnaissant l'ensemble des termes de $T(\Gamma)$ satisfaisant la condition EGALITÉ de la définition 175 tels que

$$\mathcal{L}(\mathcal{A}_{grille}) = \mathcal{L}(\mathcal{A}_g) \cap \mathcal{L}(\mathcal{A}_{gl}).$$

Les conditions PURETÉ, FILS-1, FILS-2, CONSTANTES, CORPS-1 et CORPS-2 de la définition 175 correspondent à des tests locaux sur les sous-arbres d'un arbre. Nous définissons pour chacun de ces tests, un ensemble de contextes qu'un arbre ne doit pas entourer pour que celui-ci satisfasse la condition associée au test.

- Condition FILS-1 : $Fi_1 = \{f(h, x_1, x_2) \mid h \in \Gamma \setminus \{f, \perp_0\}\}$;
- Condition FILS-2 : $Fi_2 = \{f(x_1, h, x_2) \mid h \in \Gamma \setminus \{f, \perp_0\}\}$;
- Condition CONSTANTES : $Co = \{f(x_1, x_2, \perp_0), f(x_1, x_2, f(x_3, x_4, x_5))\}$;
- Condition CORPS-1 : $Co_1 = \{f(f(x_1, h, x_2), x_3, x_4) \mid h \in \Gamma_0\}$;
- Condition CORPS-2 : $Co_2 = \{f(h, f(f(x_1, x_2, x_3), x_4, x_5), x_6) \mid h \in \Gamma_0\}$.

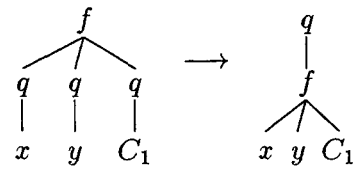
Soit C un contexte de $Fi_1 \cup Fi_2 \cup Co \cup Co_1 \cup Co_2$. Alors l'ensemble des termes de $T(\Gamma)$ entourant C est régulier (propriété 57). Or la classe des réguliers est close par complément (propriété 45), donc l'ensemble des termes de $T(\Gamma)$ n'entourant pas C est régulier. Donc pour toute condition ci-dessus, l'ensemble des termes de $T(\Gamma)$ vérifiant celle-ci est régulier car la classe des réguliers est close par intersection (propriété 45).

De cette même propriété, nous déduisons qu'il existe un automate d'arbres standard $\mathcal{A}_{gl}$ reconnaissant l'ensemble des termes de $T(\Gamma)$ satisfaisant les conditions PURETÉ, FILS-1, FILS-2, CONSTANTES, CORPS-1 et CORPS-2 de la définition 175 (la condition PURETÉ étant satisfaite par le fait que tout arbre de $\mathcal{L}(\mathcal{A}_{gl})$ appartient à $T(\Gamma)$). Notons que $\mathcal{A}_{gl}$ est un *ATEC* puisque le pouvoir d'expression des *ATEC* est strictement supérieur à celui des *ATEF* (propriété 158), lui-même strictement supérieur à celui des *AAS* (propriété 130).

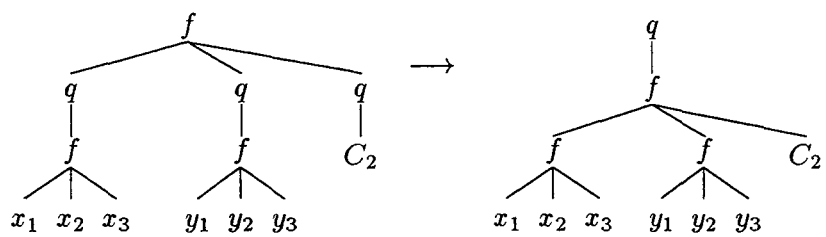
Montrons maintenant que le langage

$$\mathcal{L} = \{t \in T(\Gamma) \mid t \text{ satisfait la condition EGALITÉ de la définition 175}\}$$

est reconnu par l'automate à tests d'égalité entre cousins $\mathcal{A}_g = (\Gamma, \{q\}, \{q\}, \Delta)$ dont l'ensemble des règles contient toutes les règles que l'on peut construire à partir de l'alphabet Γ et du seul état q exceptées les règles suivantes :



où x, y sont des variables distinctes, et C_1 est une constante ou une variable z,



où $x_2 \neq y_1$, et C_2 est une constante ou l'arbre $f(z_1, z_2, z_3)$.

Considérons un terme t de $T(\Gamma)$ de la forme $f(f(t_1, t_2, t_3), f(t_4, t_5, t_6), t_7)$ avec $t_2 \neq t_4$ et supposons que pour tout entier i de $[7], t_i \xrightarrow{*}_{\mathcal{A}_g} q(t_i)$. S'il existe une règle r de l'automate $\mathcal{A}_g$ et un terme t' tel que $t \xrightarrow{r} t'$, alors r est :

- Soit de la première forme définie ci-dessus avec x, y variables et C_1 une constante ou une variable;
- Soit de la seconde forme avec $x_2 \neq y_1$ et C_2 une constante ou l'arbre $f(z_1, z_2, z_3)$.

Nous en déduisons qu'aucune règle de l'automate $\mathcal{A}_g$ n'est applicable à t est donc que t n'appartient pas à $\mathcal{L}(\mathcal{A}_g)$. Nous montrons alors facilement que $\mathcal{L}(\mathcal{A}_g) = \mathcal{L}$.

Finalement puisque la classe *RATEC* est close par intersection (propriété 160), il existe un automate d'arbres à tests d'égalité entre cousins $\mathcal{A}_{grille}$ tel que $\mathcal{L}(\mathcal{A}_{grille}) = \mathcal{L}(\mathcal{A}_g) \cap \mathcal{L}(\mathcal{A}_{gl}) = \{t \in T(\Gamma) \mid t \text{ est un G-Arbre sur } \Gamma\}$. □

Maintenant que nous savons reconnaître à l'aide d'un automates d'arbres à tests d'égalité entre cousins l'ensemble des G-Arbres sur Γ, nous allons utiliser la proposition 177 pour reconnaître les G-Arbres codant les calculs réussis de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide. Nous montrons pour cela qu'il existe un automate d'arbres à tests d'égalité entre cousins $\mathcal{A}_{stop}$ reconnaissant cet ensemble de G-Arbres (proposition 180). Dans un premier temps, nous montrons qu'il existe des automates d'arbres standards $\mathcal{A}_{init}$, $\mathcal{A}_{calcul}$ et $\mathcal{A}_{final}$ reconnaissant les arbres satisfaisant les différentes conditions énoncées par la proposition 177.

Ensuite nous utilisons les propriétés de clôture des automates pour en déduire l'existence de l'automate $\mathcal{A}_{stop}$.

Proposition 180. *Il existe un automates d'arbres à tests d'égalité entre cousins $\mathcal{A}_{stop}$ reconnaissant le langage $\mathcal{L}$ suivant:*

$$\mathcal{L} = \{t \mid t \text{ est un G-Arbre sur } \Gamma \text{ codant un calcul réussi de la machine } \mathcal{M} \text{ débutant sur l'entrée vide}\}.$$

Preuve : Nous allons tout d'abord montrer qu'il existe un *AAS* $\mathcal{A}_{init}$ reconnaissant le language constitué des termes de $T(\Gamma)$ satisfaisant les conditions INIT1, INIT2 et INIT3 de la proposition 177. Puisque l'ensemble des termes de $T(\Gamma)$ entourant un contexte de $T(\Gamma, \mathcal{X})$ est régulier (propriété 57) et la classe des réguliers est close par complément (propriété 45), il existe trois *AAS* $\mathcal{A}_{init1}$, $\mathcal{A}_{init2'}$ et $\mathcal{A}_{init3'}$ tels que:

$$\mathcal{L}(\mathcal{A}_{init1}) = \{t \in T(\Gamma) \mid t \text{ entoure l'arbre } f(x_1, f(x_2, f(x_3, \bot_0, \Box_r), \Box_r), q_s)\}$$
$$\mathcal{L}(\mathcal{A}_{init2}) = \{t \in T(\Gamma) \mid t \text{ n'entoure pas l'arbre } f(f(x_1, x_2, q_s), x_3, x_4)\}$$
$$\mathcal{L}(\mathcal{A}_{init3}) = \{t \in T(\Gamma) \mid t \text{ n'entoure pas l'arbre } f(x_1, f(x_2, x_3, q_s), x_4)\}$$

Nous déduisons alors de la clôture par intersection de la classe des réguliers (propriété 45) qu'il existe un *AAS* $\mathcal{A}_{init}$ tel que:

$$\begin{aligned}\mathcal{L}(\mathcal{A}_{init}) &= \mathcal{L}(\mathcal{A}_{init1}) \cap \mathcal{L}(\mathcal{A}_{init2}) \cup \mathcal{L}(\mathcal{A}_{init3}) \\ &= \{t \in T(\Gamma) \mid t \text{ satisfait les conditions INIT1, INIT2 et INIT3 de la proposition 177}\}\end{aligned}$$

Puisque les motifs de P'_T et l'arbre $f(\bot_0, x_1, q_f)$ sont des contextes de $T(\Gamma, \mathcal{X})$, nous montrons, en procédant de façon similaire à ci-dessus, qu'il existe $\mathcal{A}_{calcul}$ et $\mathcal{A}_{final}$ automates d'arbres standards tels que:

$$\mathcal{L}(\mathcal{A}_{calcul}) = \{t \in T(\Gamma) \mid t \text{ n'entoure aucun des motifs de } P'_T\}$$
$$\mathcal{L}(\mathcal{A}_{final}) = \{t \in T(\Gamma) \mid t \text{ entoure l'arbre } f(\bot_0, x_1, q_f)\}$$

Finalement nous déduisons du fait que la classe des réguliers est incluse dans la classe *ATEC* (propriétés 158 et 130) et de la propriété de clôture par intersection de la classe *RATEC* (propriété 128) qu'il existe un automate d'arbres à tests d'égalité entre cousins $\mathcal{A}_{stop}$ tel que:

$$\mathcal{L}(\mathcal{A}_{stop}) = \mathcal{L}(\mathcal{A}_{grille}) \cap \mathcal{L}(\mathcal{A}_{init}) \cap \mathcal{L}(\mathcal{A}_{calcul}) \cap \mathcal{L}(\mathcal{A}_{final}).$$

De façon évidente, $\mathcal{L}(\mathcal{A}_{stop})$ est l'ensemble des G-Arbres sur Γ satisfaisant les cinq conditions énoncées dans la proposition 177. Nous déduisons alors de la proposition 177 que $\mathcal{L}(\mathcal{A}_{stop}) = \mathcal{L}$. □

Preuve : théorème 178. D'après la proposition 180, si nous pouvons décider si le langage reconnu par l'automate $\mathcal{A}_{stop}$ est vide ou non, nous pouvons décider si la machine de Turing $\mathcal{M}$ s'arrête sur l'entrée vide ce qui contredit le théorème 166. Nous en déduisons que le problème du vide est indécidable pour les automates à tests d'égalité entre cousins. □

Le théorème 178 permet d'affiner la frontière entre la décidabilité et l'indécidabilité du problème du vide pour les automates d'arbres à contraintes. En effet ce problème est décidable lorsqu'on effectue des tests à profondeur un (ATF) mais devient indécidable dès que l'on effectue uniquement des tests d'égalité à profondeur deux ($ATEC$).

Chapitre 2

Tests d'égalités entre frères et cousins

Dans le chapitre précédent, nous avons affiné la frontière entre la décidabilité et l'indécidabilité du problème du vide pour les automates d'arbres à contraintes en montrant que ce problème est indécidable lorsqu'on effectue des tests d'égalité à profondeur deux (automates à tests d'égalité entre cousins). Nous montrons dans ce chapitre que l'on peut encore affiner cette frontière en définissant une classe d'automates à contraintes autorisant des tests d'égalité entre frères et cousins contenant la classe des automates à tests d'égalité entre frères et dans laquelle le vide est décidable. Cette classe d'automates est obtenue en effectuant des restrictions sur la forme des règles des automates.

2.1 Définition

Dans la preuve de l'indécidabilité du problème du vide pour les *ATEC* du chapitre précédent, nous construisons un automate permettant de vérifier si un arbre code bien une grille, c'est-à-dire qu'en faisant à partir d'une cellule de la grille un pas vers le haut puis un pas vers la droite, on obtient la même description qu'en faisant un pas vers la droite puis un pas vers le haut (point EGALITÉ de la définition 175). Un tel automate est en fait un *ATEF* (automate $\mathcal{A}_g$ de la preuve du lemme 179) dont les règles testent si pour tout sous-terme de motif $(f(f(x_1, x_2, x_3), f(x_4, x_5, x_6), x_7))$ d'un terme, on a $x_2 = x_4$. Les positions de x_2 et x_4 ne sont pas les mêmes relativement à leur père respectif ce qui permet de coder la structure de la grille. Ce codage ne semblant plus possible en imposant que dans les règles entre cousins les positions de deux variables égales soient les mêmes relativement à leur père respectif, nous nous sommes intéressés à cette restriction afin de voir si celle-ci nous permettait de définir des automates à tests d'égalité entre cousins particuliers pour lesquels le problème du vide serait décidable. Cette étude a abouti à la définition des automates à tests d'égalités entre frères et cousins se trouvant ci-dessous. Dans ce chapitre, nous étudions les différentes propriétés de cette nouvelle sous-classe d'automates à contraintes.

Définition 181 (*ATEFCS*). *Soit $\mathcal{X}$ un ensemble dénombrables de variables. Un* automate d'arbres à tests d'égalité entre frères et cousins simple *(ATEFCS) est un automate à contraintes (Σ, Q, Q_f, Δ) dont les règles sont de la forme :*

1. *$a \to q(a)$ où a est une constante de Σ et q un état de Q;*

2. $f(q_1(x_1),\ldots,q_n(x_n)) \to q(f(x_1,\ldots,x_n))$ *où* f *est un symbole de* Σ *d'arité* n *supérieure ou égale à un,* $q_1,\ldots,q_n,q$ *des états de* $\mathcal{Q}$ *et* $x_1,\ldots,x_n$ *des variables de* $\mathcal{X}$ *telles que* $(x_i)_{i\in[n]}$ *est valide pour* $(q_i)_{i\in[n]}$.

3. $f(q_1(u_1),\ldots,q_n(u_n)) \to q(f(u_1,\ldots,u_n))$ *avec*

   (a) f *un symbole de* Σ *d'arité* n *supérieure ou égale à un,* $q_1,\ldots,q_n,q$ *des états de* $\mathcal{Q}$;

   (b) *Pour tout entier* i *de* $[n]$, u_i *est un terme linéaire* $f_i(x^i_1,\ldots,x^i_{p_i})$ *où* f_i *est un symbole de* Σ *d'arité* p_i *et* $x^i_1,\ldots,x^i_{p_i}$ *des variables de* $\mathcal{X}$;

   (c) *Pour tous entiers* u,v,w,z, *si* $x^u_w = x^v_z$ *alors* $w = z$, $f_u = f_v$ *et* $q_u = q_v$, *c'est-à-dire si deux variables sont égales, elles sont situées à la même position sous des symboles de fonction et des états égaux.*

Nous désignons par *RATEFCS* la classe des langages reconnus par les *ATEFCS.*

Exemple 182. *Soit l'automate* $\mathcal{A} = (\{a/0, b/0, c/0, g/1, f/2, h/2\}, \{q_a, q_b, q_c, q, q_f\}, \{q_f\}, \Delta)$ *avec* Δ *composé des règles de transition suivantes:*

$$\begin{aligned} a &\to q_a(a)\\ b &\to q_b(b)\\ c &\to q_c(c)\\ g(q_a(x)) &\to q_a(g(x))\\ f(q_a(x),q_b(y)) &\to q(f(x,y))\\ f(q_a(x),q_c(y)) &\to q(f(x,y))\\ h(q(f(x,y)),q(f(x,z))) &\to q_f(h(f(x,y),f(x,z))) \end{aligned}$$

Alors $\mathcal{A}$ *est un ATEFCS reconnaissant le langage suivant:*

$$\mathcal{L} = \{t \in T(\Sigma) \mid t = h(f(t_1,t_2),f(t_1,t_3)) \text{ avec } t_1 \in T(\{g,a\}) \text{ et } t_2,t_3 \in \{b,c\}\}.$$

2.2 Propriétés de clôtures booléennes

2.2.1 Union

Proposition 183. *La classe RATEFCS est close par union.*

Preuve : Soient $\mathcal{L}$ et $\mathcal{L}'$ deux langages de la classe *RATEFCS* reconnus respectivement par les *ATEFCS* $\mathcal{A} = (\Sigma,\mathcal{Q},\mathcal{Q}_f,\Delta)$ et $\mathcal{A}' = (\Sigma,\mathcal{Q}',\mathcal{Q}'_f,\Delta')$. Nous pouvons supposer sans perte de généralité que les ensemble $\mathcal{Q}$ et $\mathcal{Q}'$ sont disjoints. Alors de façon évidente l'automate $\mathcal{B} = (\Sigma,\mathcal{Q}\cup\mathcal{Q}',\mathcal{Q}_f\cup\mathcal{Q}'_f,\Delta\cup\Delta')$ est un *ATEFCS* reconnaissant $\mathcal{L}\cup\mathcal{L}'$. □

2.2.2 Intersection

Proposition 184. *La classe RATEFCS n'est pas close par intersection avec la classe des réguliers, c'est-à-dire si $\mathcal{L}$ est un langage de RATEFCS et $\mathcal{L}_r$ un langage régulier, alors $\mathcal{L} \cap \mathcal{L}_r$ n'appartient pas forcément à RATEFCS.*

Preuve : Soit $\mathcal{L}$ le langage de la classe *RATEFCS* défini dans l'exemple 182 et $\mathcal{L}_r$ le langage régulier reconnu par l'automate :

$$\mathcal{A}_r = (\{a/0, b/0, c/0, g/1, f/2, h/2\}, \{q_a, q_b, q_c, q_1, q_2, q_f\}, \{q_f\}, \Delta_r)$$

avec Δ_r constitué des règles suivantes :

$$\begin{aligned} a &\to q_a(a) \\ b &\to q_b(b) \\ c &\to q_c(c) \\ g(q_a(x)) &\to q_a(g(x)) \\ f(q_a(x), q_b(y)) &\to q_1(f(x,y)) \\ f(q_a(x), q_c(y)) &\to q_2(f(x,y)) \\ h(q_1(x), q_2(y)) &\to q_f(h(x,y)) \end{aligned}$$

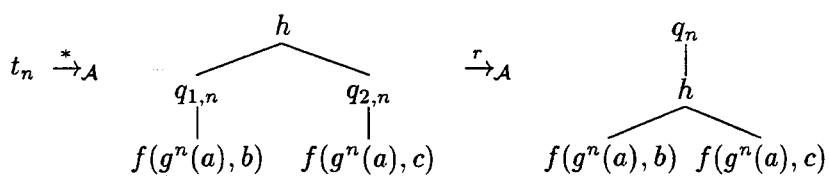
Alors $\mathcal{L}_r = \{h(f(g^m(a), b), f(g^{m'}(a), c)) \mid m, m' \in \mathbb{N}\}$. Donc

$$\mathcal{L} \cap \mathcal{L}_r = \{h(f(g^n(a), b), f(g^n(a), c)) \mid n \in \mathbb{N}\}.$$

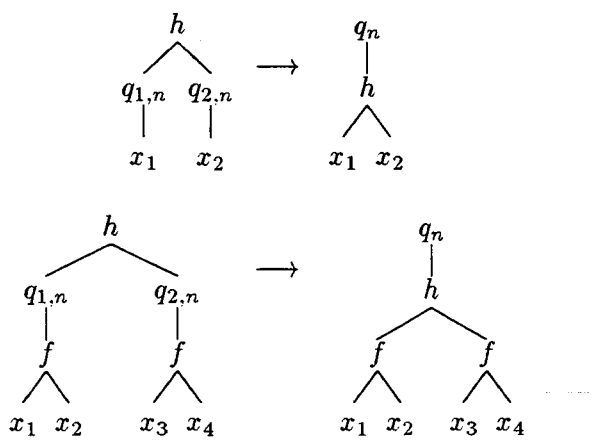
Supposons que le langage $\mathcal{L} \cap \mathcal{L}_r$ soit reconnu par un *ATEFCS* $\mathcal{A}$:

$$\mathcal{A} = (\{a/0, b/0, c/0, g/1, f/2, h/2\}, \mathcal{Q}, \mathcal{Q}_f, \Delta_s).$$

Pour tout entier n, nous notons t_n le terme $h(f(g^n(a), b), f(g^n(a), c))$. Alors puisque t_n appartient à $\mathcal{L} \cap \mathcal{L}_r$, il existe un état final q_n de $\mathcal{Q}_f$, deux états $q_{1,n}, q_{2,n}$ de $\mathcal{Q}$ et une règle r de Δ_s tels que :



avec r de l'une des formes suivantes :



Dans la première forme de règle, les variables x_1 et x_2 sont distinctes d'après la point (2) de la définition des *ATEFCS*).

Dans la deuxième forme de règle, les variables sont deux à deux distinctes (points (3b) et (3c) de la définition des *ATEFCS*). En particulier, si les variables x_1 et x_3 étaient égales, on aurait $q_{1,n} = q_{2,n}$ et donc $h(f(g^n(a), b), f(g^n(a), b))$ appartiendrait à $\mathcal{L} \cap \mathcal{L}_r$.

Soient m et n deux entiers distincts ($m \neq n$). Puisque t_m et t_n appartiennent à $\mathcal{L}$, nous avons $f(g^m(a), b) \xrightarrow{*}_{\mathcal{A}} q_{1,m}$ et $f(g^n(a), c) \xrightarrow{*}_{\mathcal{A}} q_{2,n}$. Supposons que $q_{1,m} = q_{1,n}$, alors

$$h(g(g^m(a), b), f(g^n(a), c)) \xrightarrow{*}_{\mathcal{A}} q_n(h(g(g^m(a), b), f(g^n(a), c))).$$

Donc le terme $h(g(g^m(a), b), f(g^n(a), c))$ appartient à $\mathcal{L} \cap \mathcal{L}_r$. On aboutit à une contradiction d'où $q_{1,m} \neq q_{1,n}$. Les états de la famille $(q_{1,n})_{n \in \mathbb{N}}$ sont donc deux à deux distincts ce qui contredit la finitude de l'ensemble des états Q. Nous en déduisons que $\mathcal{L} \cap \mathcal{L}_r$ n'est pas un langage de la classe *RATEFCS*. □

Nous déduisons facilement de cette proposition et de la proposition 187, le fait suivant :

Corollaire 185. *La classe RATEFCS n'est pas close par intersection.*

2.2.3 Complément

Proposition 186. *La classe RATEFCS n'est pas close par complément.*

Preuve : Supposons que la classe *RATEFCS* soit close par complément. Alors soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages de la classe *RATEFCS*. Nous avons $\mathcal{L}_1 \cap \mathcal{L}_2 = \complement(\complement\mathcal{L}_1 \cup \complement\mathcal{L}_2)$. Nous en déduisons que la classe *RATEFCS* est close par intersection puisqu'elle est close par union (proposition 183) et par complément (par hypothèse). Ceci contredit le corollaire 185. Finalement la classe *RATEFCS* n'est pas close par complément. □

2.3 Expressivité

Proposition 187. *Le pouvoir d'expression des ATEFCS est strictement supérieur à celui des ATEF.*

Preuve : Soit $\mathcal{L}$ un langage de la classe *ATEF*. D'après la propriété 135, il existe un *ATEF* $\mathcal{A}$ normalisé reconnaissant $\mathcal{L}$. Nous déduisons de la définition des règles des automates à tests d'égalité entre frères normalisés que $\mathcal{A}$ est un *ATEFCS*. Donc le pouvoir d'expression des *ATEFCS* est supérieur à celui des *ATEF*.

Finalement le langage $\mathcal{L}$ défini dans l'exemple 182 n'est pas un langage de la classe *RATEF* d'après la preuve de la propriété 158. □

De façon évidente, tout *ATEFCS* est un *RATEG* donc le pouvoir d'expression des *RATEG* est supérieur à celui des *ATEFCS*. De plus la classe des langages reconnus par *RATEG* étant close par intersection (propriété 73) et la classe *RATEFCS* ne l'étant pas (corollaire 185), nous avons la propriété suivante.

Propriété 188. *Le pouvoir d'expression des RATEG est strictement supérieur à celui des ATEFCS.*

2.4 Morphismes d'arbres

Proposition 189. *La classe RATEFCS n'est pas close par réétiquetages inverses.*

Preuve : Soient les signatures $\Sigma = \{a/0, g/1, h/1, f/2\}$ et $\Gamma = \{a/0, g/1, f/2\}$, et le réétiquetage Φ défini par l'application φ de Σ dans $T(\Gamma, \mathcal{X})$ suivante :

$$\begin{array}{lcllcl} \varphi(a) & = & a & \varphi(g) & = & g(x_1) \\ \varphi(f) & = & f(x_1, x_2) & \varphi(h) & = & g(x_1) \end{array}$$

L'ensemble $\{f(g^n(a), g^n(a) \mid n \in \mathbb{N}\}$ est un langage d'arbres sur Γ de la classe *RATEFCS* et son image inverse par Φ est l'ensemble des arbres dont toutes les branches sont de même longueur et sont constituées des symboles g et h. On peut montrer que ce dernier langage n'appartient pas à la classe *RATEFCS* puisque les règles des *ATEFCS* ne permettent pas de tester l'égalité de la hauteur de deux termes différents. Nous en déduisons que la classe *ATEFCS* n'est pas close par réétiquetages inverses. □

2.5 Problème du vide

Nous montrons dans cette section, le théorème suivant :

Théorème 190. *Le problème du vide est décidable pour les ATEFCS.*

Nous considérons dans toute cette section un *ATEFCS* $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$. Nous allons calculer pour tout entier v, tout état q de $\mathcal{Q}$ et tout symbole de fonction f de Σ, l'ensemble des arbres t tels que $Haut(t) \leq v$, $t \xrightarrow{*} q(t)$ et $tête(t) = f$ (section 2.5.1).

Nous montrons alors qu'il existe un entier k ne dépendant que des données de l'automate $\mathcal{A}$ tel que pour tout entier v et pour tout ensemble défini ci-dessus de hauteur v non vide, l'ensemble correspondant de hauteur $k+1$ est non vide (section 2.5.2). Nous en déduisons que le problème du vide est décidable pour les *ATEFCS*.

2.5.1 Algorithme

Nous définissons dans cette section l'algorithme qui va nous permettre de calculer l'entier k. Tout d'abord, nous définissons les ensembles cités ci-dessus. Pour tout entier v, tout état q de $\mathcal{Q}$ et tout symbole de fonction f de Σ, soient les ensembles $\langle q, f\rangle_{\leq v}$ et $\langle q, f\rangle_{=v}$ définis par :

$$\begin{array}{lcl} \langle q, f\rangle_{\leq v} & = & \{t \in T(\Sigma) \mid Haut(t) \leq v, t \xrightarrow{*} q(t) \text{ et } tête(t) = f\}; \\ \langle q, f\rangle_{=v} & = & \langle q, f\rangle_{\leq v} \cap \{t \in T(\Sigma) \mid Haut(t) = v\}. \end{array}$$

Les différents ensembles $\langle q, f\rangle_{\leq v}$ sont calculables et placés dans une table à deux entrées comme suit (figure 1) :

Colonnes : Pour tout état q de $\mathcal{Q}$ et tout symbole f de Σ, le couple $\langle q, f\rangle$ est la tête d'une colonne (remarquons que le nombre de colonnes est fini puisque les nombres d'états et de symboles de fonction sont finis) ;

Lignes : Pour tout entier v, v est la tête d'une ligne ;

FIG. 1 - Table

	$\langle q_1, f_1\rangle$	...	$\langle q_1, f_m\rangle$	$\langle q_2, f_1\rangle$	...	$\langle q_n, f_m\rangle$
0	$\langle q_1, f_1\rangle_{\leq 0}$	...	$\langle q_1, f_m\rangle_{\leq 0}$	$\langle q_2, f_1\rangle_{\leq 0}$	...	$\langle q_n, f_m\rangle_{\leq 0}$
1	$\langle q_1, f_1\rangle_{\leq 1}$	...	$\langle q_1, f_m\rangle_{\leq 1}$	$\langle q_2, f_1\rangle_{\leq 1}$	...	$\langle q_n, f_m\rangle_{\leq 1}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
v	$\langle q_1, f_1\rangle_{\leq v}$	...	$\langle q_1, f_m\rangle_{\leq v}$	$\langle q_2, f_1\rangle_{\leq v}$	...	$\langle q_n, f_m\rangle_{\leq v}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Cellules : Pour tout entier v, tout état q de $\mathcal{Q}$ et tout symbole f de Σ, la cellule se trouvant à l'intersection de la ligne dont la tête est v et de la colonne dont la tête est $\langle q, f\rangle$ contient l'ensemble $\langle q, f\rangle_{\leq v}$.

Soient q un état de $\mathcal{Q}$ et f symbole de Σ. Alors dans la colonne de tête $\langle q, f\rangle$, les ensembles $\langle q, f\rangle_{\leq v}$ sont croissants, c'est-à-dire pour tout entier v :

$$\langle q, f\rangle_{\leq v} \subseteq \langle q, f\rangle_{\leq v+1} \tag{1}$$

Nous regardons si les ensembles de la table sont vides ou non et nous calculons celle-ci ligne par ligne jusqu'à obtention de deux lignes successives telles que tout ensemble non vide le soit précédemment. En fait, la table est construite par un algorithme calculant les lignes $0, 1, \ldots$ et ainsi de suite. Et l'algorithme s'arrête après avoir calculé la ligne $k+1$ si et seulement si la condition d'arrêt suivante est satisfaite :

$$\forall q \in \mathcal{Q}, \forall f \in \Sigma \ \left(\langle q, f\rangle_{\leq k} = \emptyset \ \Rightarrow \ \langle q, f\rangle_{\leq k+1} = \emptyset\right).$$

Proposition 191. *L'algorithme termine.*

Preuve : La table est constituée de $card(\mathcal{Q}) \times card(\Sigma)$ colonnes. De plus dans une colonne donnée, les ensembles sont croissants (relation (1)). Donc il existe un entier $k \leq card(\mathcal{Q}) \times card(\Sigma)$ tel que la condition d'arrêt de l'algorithme soit satisfaite. □

Dans la suite de la preuve, $k+1$ désigne l'indice de la ligne pour lequel l'algorithme s'arrête c'est-à-dire le plus petit entier pour lequel on a :

$$\forall q \in \mathcal{Q}, \forall f \in \Sigma \ \left(\langle q, f\rangle_{\leq k} = \emptyset \ \Rightarrow \ \langle q, f\rangle_{\leq k+1} = \emptyset\right).$$

2.5.2 Preuve du vide

Cette section dédiée à la preuve de la décidabilité du problème du vide pour les *ATEFCS* est constituée de trois étapes. Dans un premier temps, nous montrons que s'il existe un terme de hauteur supérieure ou égale à $k+1$ de symbole de tête f et appartenant à $\mathcal{L}(\mathcal{A}, q)$, alors il existe un terme de hauteur inférieure ou égale à $k+1$ de symbole de tête f et appartenant à $\mathcal{L}(\mathcal{A}, q)$ (lemme 192). Nous en déduisons que s'il existe un terme de symbole de tête f

et appartenant à $\mathcal{L}(\mathcal{A}, q)$, alors il existe un terme de hauteur inférieure ou égale à $k+1$ de symbole de tête f et appartenant à $\mathcal{L}(\mathcal{A}, q)$ (corollaire 193). Finalement nous prouvons la décidabilité du problème du vide pour les *ATEFCS* (théorème 190).

Lemme 192. $\forall v \in \mathbb{N}, v \geq k+1, \forall q \in \mathcal{Q}, \forall f \in \Sigma, \quad (\langle q, f\rangle_{=v} \neq \emptyset \Rightarrow \langle q, f\rangle_{\leq k+1} \neq \emptyset)$.

Preuve : La preuve se fait par induction sur la valeur de v. La propriété énoncée est satisfaite de façon évidente dans le cas où $v=k+1$.

Soit v un entier supérieur ou égal à $k+1$. Supposons que la propriété énoncée par le lemme soit satisfaite pour tout entier v' tel que $k+1 \leq v' \leq v$. Supposons de plus qu'il existe un état q de $\mathcal{Q}$ et un symbole f de Σ tels que $\langle q, f\rangle_{=v+1}$ soit non vide. Alors il existe $t, t_1, \ldots, t_n$ termes de $T(\Sigma)$, t' terme de $T(\Sigma \cup \mathcal{Q})$ et r règle de Δ tels que :

- $Haut(t) = v+1$,
- $t = f(t_1, \ldots, t_n)$, et
- $t \xrightarrow{*} t' \xrightarrow{r} q(t)$.

Nous montrons que $\langle q, f\rangle_{\leq k+1}$ est non vide en construisant un terme $s = f(s_1, \ldots, s_n)$ de $T(\Sigma)$ de hauteur inférieure ou égale à $k+1$ et un terme s' de $T(\Sigma \cup \mathcal{Q})$ tels que $s \xrightarrow{*} s' \xrightarrow{r} q(s)$. Nous distinguons deux cas suivant la forme de la règle r.

Premier cas : r est de la forme $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$.

Montrons tout d'abord que pour tout entier i de $[n]$, on a $\langle q_i, tête(t_i)\rangle_{\leq k}$ non vide. Soit i un entier de $[n]$. De façon évidente $t_i \xrightarrow{*} q_i(t_i)$ et $Haut(t_i) \leq v$. Donc par définition, t_i appartient à $\langle q_i, tête(t_i)\rangle_{\leq v}$. Nous en déduisons qu'il existe un entier v_i inférieur ou égal à v tel que t_i appartienne à $\langle q_i, tête(t_i)\rangle_{=v_i}$. Donc $\langle q_i, tête(t_i)\rangle_{\leq k+1}$ est non vide puisque :

- Si $v_i \leq k+1$, $\langle q_i, tête(t_i)\rangle_{=v_i} \subseteq \langle q_i, tête(t_i)\rangle_{\leq k+1}$ et $\langle q_i, tête(t_i)\rangle_{=v_i} \neq \emptyset$;
- Si $v_i > k+1$, $v_i \leq v$ donc par hypothèse d'induction $\langle q_i, tête(t_i)\rangle_{\leq k+1} \neq \emptyset$.

Nous en déduisons par la condition d'arrêt de l'algorithme que pour tout i de $[n]$, $\langle q_i, tête(t_i)\rangle_{\leq k}$ est non vide. De plus, par définition des règles (point (2) de la définition des *ATEFCS*), on a $(x_i)_{i\in[n]}$ valide pour $(q_i)_{i\in[n]}$. D'où pour tous entiers i, j de $[n]$, $(x_i = x_j) \Rightarrow (q_i = q_j)$. Donc il existe une famille $(s_i)_{i\in[n]}$ de termes de $T(\Sigma)$ telle que :

- $\forall i \in [n], s_i \in \langle q_i, tête(t_i)\rangle_{\leq k}$;
- $\forall i, j \in [n], (x_i = x_j) \Rightarrow (s_i = s_j)$.

Nous en déduisons que $(x_i)_{i\in[n]}$ est valide pour $(s_i)_{i\in[n]}$ et donc que

$$s \xrightarrow{*} f(q_1(s_1), \ldots, q_n(s_n)) \xrightarrow{r} q(s).$$

Donc s appartient à $\langle q, f\rangle_{\leq k+1}$.

Second cas : r est de la forme $f(q_1(u_1), \ldots, q_n(u_n)) \to q(f(u_1, \ldots, u_n))$ avec pour tout entier i de $[n]$, $u_i = f_i(x_1^i, \ldots, x_{p_i}^i)$.

Pour tout entier i de $[n]$, nous avons $tête(t_i) = f_i$ donc nous montrons comme dans le premier cas que pour tout i de $[n]$, $\langle q_i, f_i \rangle_{\leq k}$ est non vide.

Soit $(E_l)_{l \in I}$, I ensemble d'indices, la partition de $[n]$ telle que :

- $\forall l \in I, \forall u, v \in E_k, f_u = f_v$ et $q_u = q_v$;
- $\forall l, l' \in I, l \neq l', \forall u \in E_l, \forall v \in E_{l'}, f_u \neq f_v$ ou $q_u \neq q_v$.

Il existe une famille $(s_i)_{i \in [n]}$ de termes de $T(\Sigma)$ telle que :

- $\forall l \in I, \forall u \in E_l, s_u \in \langle q_u, f_u \rangle_{\leq k}$, et
- $\forall l \in I, \forall u, v \in E_l, s_u = s_v$.

Pour tout i de $[n]$, $f_i(x_1^i, \ldots, x_{p_i}^i)$ est linéaire (point (3b) de la définition des *ATEFCS*), donc $(x_j)_{j \in [p_i]}$ est valide pour $(s_i|_j)_{j \in [p_i]}$.

De plus, soient u, v, w, z des entiers tels que $x_w^u = x_z^v$, alors $f_u = f_v$, $q_u = q_v$ et $w = z$ (point (3c) de la définition des *ATEFCS*). Donc par définition de la partition $(E_k)_{k \in I}$, on a $s_u = s_v$. D'où $s_u|_w = s_v|_z$. Nous en déduisons que $(x_j^i)_{i \in [n], j \in [p_i]}$ est valide pour $(s_i|_j)_{i \in [n], j \in [p_i]}$.

Donc $s \xrightarrow{*} f(q_1(s_1), \ldots, q_n(s_n)) \xrightarrow{r} q(s)$. D'où s appartient à $\langle q, f \rangle_{\leq k+1}$.

Finalement $\langle q, f \rangle_{\leq k+1}$ est non vide dans tous les cas. □

Corollaire 193. $\forall v \in \mathbb{N}, \forall q \in \mathcal{Q}, \forall f \in \Sigma, \ (\langle q, f \rangle_{=v} \neq \emptyset \ \Rightarrow \ \langle q, f \rangle_{\leq k+1} \neq \emptyset)$.

Preuve : Soient v un entier, q un état de $\mathcal{Q}$ et f un symbole de Σ tels que $\langle q, f \rangle_{=v}$ soit non vide.

Dans le cas où v est supérieur ou égal à $k+1$, nous déduisons du lemme 192 que $\langle q, f \rangle_{\leq k+1}$ est non vide. Dans le cas où v est strictement inférieur à $k+1$, nous déduisons des définitions que $\langle q, f \rangle_{=v}$ est inclus dans $\langle q, f \rangle_{\leq k+1}$. Donc $\langle q, f \rangle_{\leq k+1}$ est non vide. □

Nous montrons maintenant que le problème du vide est décidable pour les *ATEFCS* (théorème 190).

Preuve : théorème 190. Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ un *ATEFCS*.

$$\begin{aligned} \mathcal{L}(\mathcal{A}) \neq \emptyset &\Leftrightarrow \exists t \in T(\Sigma), \exists q \in \mathcal{Q}_f \text{ tels que } t \xrightarrow{*} q(t) \\ &\Leftrightarrow \exists v \in \mathbb{N}, \exists q \in \mathcal{Q}_f, \exists f \in \Sigma \text{ tels que } \langle q, f \rangle_{=v} \neq \emptyset \\ &\Rightarrow \exists q \in \mathcal{Q}_f, \exists f \in \Sigma \text{ tels que } \langle q, f \rangle_{\leq k+1} \neq \emptyset \quad \text{(Corollaire 193)} \end{aligned}$$

De plus pour tout état q de $\mathcal{Q}_f$ et tout symbole f de Σ, nous avons :

$$\langle q, f \rangle_{\leq k+1} \neq \emptyset \ \Rightarrow \ \exists v \in \mathbb{N} \text{ tel que } \langle q, f \rangle_{=v} \neq \emptyset.$$

Donc $\mathcal{L}(\mathcal{A}) \neq \emptyset \quad \Leftrightarrow \quad \exists q \in \mathcal{Q}_f, \exists f \in \Sigma$ tels que $\langle q, f \rangle_{\leq k+1} \neq \emptyset$.

Donc puisque notre algorithme termine (proposition 191), on peut décider s'il existe un arbre dans le langage $\mathcal{L}(\mathcal{A})$. Donc le problème du vide est décidable pour les *ATEFCS*. □

2.6 Perspectives

Dans ce chapitre, nous avons, comme dans le chapitre précédent, affiné la frontière entre la décidabilité et l'indécidabilité du problème du vide pour les automates d'arbres à contraintes, mais cette fois-ci du côté décidable.

Cette frontière est très fine puisque le problème du vide est indécidable pour les automates d'arbres à tests d'égalité entre cousins (*ATEC*) et décidable pour les automates d'arbres à tests d'égalité entre frères et cousins simples (*ATEFCS*). Cependant, nous espérons encore pouvoir affiner celle-ci du côté décidable en allégeant les restrictions sur les règles des *ATEFCS*. En particulier, nous pensons que seule la restriction sur les positions des variables égales pour les tests d'égalités entre cousins est indispensable pour obtenir la décidabilité du problème du vide.

Chapitre 3

Reconnaissabilité

Dans le chapitre sur les automates à contraintes, nous avons parcouru l'ensemble des classes de ces automates et leurs propriétés et applications. Une question naturelle se pose lorsqu'on considère un langage reconnu par un tel automate: est-ce que le langage est régulier, c'est-à-dire les contraintes sont-elles réellement nécessaires pour définir le langage? Cette question n'est pas facile en général mais lorsqu'elle peut être résolue, elle permet par exemple d'utiliser les algorithmes classiques des réguliers. Pour les AR, ce problème contient strictement la décidabilité de la reconnaissabilité de l'ensemble des formes normales d'un système de réécriture, problème résolu mais dont les preuves sont très techniques [92, 54]. En effet, l'ensemble des formes normales d'un système de réécriture est reconnaissable par automates à contraintes diségalitaires (propriété 102) et le pouvoir d'expression des automates de réduction est strictement supérieur à celui des automates à contraintes diségalitaires (propriété 141).

Nous montrons dans ce chapitre que ce problème est décidable pour la classe $RATF$ (classe des langages reconnaissables par automates à tests entre frères) et indécidable pour la classe $RATEC$ (classe des langages reconnaissables par automates à tests d'égalité entre cousins).

3.1 Problème de reconnaissabilité

Le problème qui nous intéresse dans ce chapitre est le suivant :

Problème de reconnaissabilité
Entrées : *Un langage $\mathcal{L}$ d'une classe C sur une signature Σ.*
Question : *Est-ce que $\mathcal{L}$ est régulier?*

Nous allons prouver que ce problème est décidable pour la classe $RATF$. De plus, quand le langage d'entrée est régulier, nous pouvons calculer un AAS le reconnaissant.

Nous donnons tout d'abord l'idée de la preuve. Nous considérons un langage $\mathcal{L}$ de la classe $RATF$ et un ATF, $\mathcal{A}$, le reconnaissant. Nous définissons dans un premier temps, un type de minimisation similaire à la minimisation classique (théorème de Myhill-Nerode pour les réguliers [20], théorème 43). Nous en déduisons un ATF, $\mathcal{A}_m$, dit minimisé, reconnaissant $\mathcal{L}$. Notre minimisation nous permet alors de montrer que $\mathcal{A}$ est régulier si et seulement si $\mathcal{A}_m$ n'est pas "sensible aux contraintes", c'est-à-dire que deux règles dont les membres gauches différent uniquement par leurs contraintes ont même membre droit. Les contraintes sont alors inutiles. Par exemple, supposons que nous avons les deux règles suivantes $f(q(x_1), q(x_2)) \xrightarrow{[1=2]}$

$q_1(f(x_1,x_2))$ et $f(q(x_1),q(x_2)) \xrightarrow{[1\neq 2]} q_2(f(x_1,x_2))$ dans l'automate minimisé. Alors si $\mathcal{L}$ est régulier, nous avons $q_1 = q_2$ et donc les contraintes peuvent être supprimées.

Avant d'aller plus loin, notons quelques détails techniques. Tout d'abord, notre minimisation tient compte de la notion de contraintes. En fait, deux états seront équivalents lorsqu'ils auront le même comportement pour le même terme avec contraintes. Ceci nous amène à introduire des termes dont les nœuds sont étiquetés par un symbole de fonction et une contrainte. Ces termes sont appelés termes contraints et sont définis de sorte que tout terme contraint possède au moins une instance close pour laquelle la contrainte de tout nœud est satisfaite.

De plus, notre raisonnement ne fonctionne pas lorsque l'automate $\mathcal{A}$ possède un état dont le langage associé est fini. Par exemple, l'automate $\mathcal{A}$ dont les règles sont $a \to q(a), b \to q(b)$ et $f(q(x_1),q(x_2)) \xrightarrow{[1=2]} q_f(f(x_1,x_2))$ reconnaît le langage $\{f(a,a), f(b,b)\}$. Ce langage est fini et donc régulier (propriété 34). Or $\mathcal{A}$ est minimisé mais nous ne pouvons supprimer la contrainte $1 = 2$.

Ce chapitre est organisé de la façon suivante. Dans un premier temps, nous éliminons les états q tels que $\mathcal{L}(\mathcal{A},q)$ est fini (section 3.2). Ensuite nous définissons la notion de termes contraints (section 3.3). Puis nous définissons et calculons l'automate "minimisé" (section 3.4). Ensuite, nous montrons que le langage est régulier si et seulement si l'automate minimisé n'est pas "sensible aux contraintes" (section 3.5). Nous en déduisons la décidabilité du problème de reconnaissabilité pour les *RATF* (théorème 194) et obtenons une construction de l'automate correspondant lorsque le langage est régulier. Nous étudions ensuite quelques applications de ce résultat (section 3.6). Enfin nous montrons que le problème de reconnaissabilité est indécidable pour la classe *RATEC* (section 3.7).

Théorème 194. *Le problème de reconnaissabilité est décidable pour la classe RATF.*

Nous considérons pour la suite de ce chapitre, une signature Σ et un ensemble de variables $\mathcal{X}$.

3.2 Réduction au cas "infini"

Soit $\mathcal{L}$ un langage de la classe *RATF*. D'après la propriété 113, il existe un *ATF*, $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$, normalisé-complet reconnaissant $\mathcal{L}$. Supposons que $\mathcal{A}$ possède au moins un état q tel que le langage associé à q ($\mathcal{L}(\mathcal{A},q)$) est fini. Nous montrons dans cette section, qu'il existe un langage $\mathcal{L}_2$ reconnu par un *ATF* dont tous les états ont un langage associé infini tel que $\mathcal{L}$ est régulier si et seulement si $\mathcal{L}_2$ est régulier. Finalement nous en déduisons que dans le reste de la preuve du théorème 194, nous pouvons considérer que pour état q de l'automate $\mathcal{A}$, $\mathcal{L}_{\mathcal{A}}(q)$ est infini. Soient

$$\mathcal{L}_1 = \bigcup_{q\in\mathcal{Q}_f, \mathcal{L}(\mathcal{A},q) \text{ fini}} \mathcal{L}(\mathcal{A},q) \quad \text{et} \quad \mathcal{L}_2 = \bigcup_{q\in\mathcal{Q}_f, \mathcal{L}(\mathcal{A},q) \text{ infini}} \mathcal{L}(\mathcal{A},q).$$

Nous avons $\mathcal{L}(\mathcal{A}) = \mathcal{L}_1 \cup \mathcal{L}_2$. Or $\mathcal{L}_1$ est fini donc régulier (propriété 34). On en déduit que $\mathcal{L}(\mathcal{A})$ est régulier si $\mathcal{L}_2$ est régulier puisque la classe des réguliers est close par union (propriété 45). De plus nous pouvons montrer que $\mathcal{L}_2$ est régulier si $\mathcal{L}(\mathcal{A})$ est régulier. En effet, à partir d'un automate d'arbres standard déterministe et complet reconnaissant $\mathcal{L}(\mathcal{A})$, on construit un automate reconnaissant $\mathcal{L}_2$ en supprimant de l'ensemble des états finaux de

l'automate de départ les états dans lesquels les termes de $\mathcal{L}_2$ se dérivent. Finalement $\mathcal{L}(\mathcal{A})$ est régulier si et seulement si $\mathcal{L}_2$ est régulier.

Le langage $\mathcal{L}_2$ est reconnu par l'ATF, $\mathcal{B} = (\Sigma, \mathcal{Q}, \mathcal{Q}'_f, \Delta)$, normalisé-complet où $\mathcal{Q}'_f = \{q \in \mathcal{Q}_f \mid \mathcal{L}(\mathcal{A}, q) \text{ infini}\}$. La suite de la preuve consiste à construire une nouvelle signature Γ en encodant pour tout état q tel que $\mathcal{L}(\mathcal{B}, q)$ est fini, les termes de $\mathcal{L}(\mathcal{B}, q)$ dans les symboles de Γ. Nous définissons ensuite un ATF, $\mathcal{B}' = (\Gamma, \mathcal{Q}', \mathcal{Q}'_f, \Delta')$, normalisé-complet tel que pour tout état q de $\mathcal{Q}'$, $\mathcal{L}(\mathcal{B}', q)$ est infini, ainsi qu'un morphisme d'arbres Φ linéaire de $T(\Gamma)$ dans $T(\Sigma)$ défini sur la définition de Γ tel que $\Phi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B})$ et $\Phi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$. Nous en déduisons que $\mathcal{L}_2$ est régulier si et seulement si $\mathcal{L}(\mathcal{B}')$ est régulier puisque Φ est linéaire (propriété 47).

Avant d'aller plus loin dans la preuve, nous donnons un exemple de construction de Γ, $\mathcal{B}'$ et Φ.

Exemple 195. *Soient* $\Sigma = \{a/0, f/2\}$ *et* $\mathcal{B} = (\Sigma, \mathcal{Q}, \mathcal{Q}'_f, \Delta)$ *avec* $\mathcal{Q} = \{q, q_p, q_f\}$, $\mathcal{Q}'_f = \{q_f\}$ *et* Δ *composé des règles suivantes:*

$$\begin{aligned}
a &\to q(a)\\
f(q(x), q(y)) &\xrightarrow{[1=2]} q_f(f(x,y))\\
f(q_f(x), q_f(y)) &\xrightarrow{[1=2]} q_f(f(x,y))\\
f(q_p(x), q_p(y)) &\xrightarrow{[1=2]} q_p(f(x,y))\\
f(q_1(x), q_2(y)) &\xrightarrow{[1\neq 2]} q_p(f(x,y)) \ \forall (q_1, q_2) \in (\mathcal{Q} \times \mathcal{Q})
\end{aligned}$$

$\mathcal{B}$ est un ATF normalisé-complet. De façon évidente $\mathcal{L}(\mathcal{B}, q) = \{a\}$, et $\mathcal{L}(\mathcal{B}, q_p)$ et $\mathcal{L}(\mathcal{B}, q_f)$ sont infinis. Soient $\square$ un symbole n'appartenant pas à Σ et la signature:

$$\Gamma = \{f_{(\square,\square)}/2, f_{(\square,a)}/1, f_{(a,\square)}/1, f_{(a,a)}/0\}.$$

Alors l'ATF $\mathcal{B}' = (\Gamma, \mathcal{Q}', \mathcal{Q}'_f, \Delta')$ est défini par $\mathcal{Q}' = \{q_p, q_f\}$ et les règles suivantes:

$$\begin{aligned}
f_{(a,a)} &\to q_f(f_{(a,a)})\\
f_{(\square,a)}(q_1(x)) &\to q_p(f_{(\square,a)}(x)) \ \forall q_1 \in \mathcal{Q}'\\
f_{(a,\square)}(q_2(x)) &\to q_p(f_{(\square,a)}(x)) \ \forall q_2 \in \mathcal{Q}'\\
f_{(\square,\square)}(q_f(x), q_f(y)) &\xrightarrow{[1=2]} q_f(f_{(\square,\square)}(x,y))\\
f_{(\square,\square)}(q_p(x), q_p(y)) &\xrightarrow{[1=2]} q_p(f_{(\square,\square)}(x,y))\\
f_{(\square,\square)}(q_1(x), q_2(y)) &\xrightarrow{[1\neq 2]} q_p(f_{(\square,\square)}(x,y)) \ \forall (q_1, q_2) \in (\mathcal{Q}' \times \mathcal{Q}')
\end{aligned}$$

Et $\Phi : T(\Gamma) \to T(\Sigma)$ est le morphisme d'arbres linéaire déterminé par l'application φ de Γ dans $T(\Sigma, \mathcal{X})$ définie par:

$$\begin{aligned}
\varphi(f_{(a,a)}) &= f(a,a) & \varphi(f_{(\square,a)})(x_1) &= f(x_1, a)\\
\varphi(f_{(\square,\square)})(x_1, x_2) &= f(x_1, x_2) & \varphi(f_{(a,\square)})(x_1) &= f(a, x_1)
\end{aligned}$$

Nous revenons maintenant à la preuve et donnons tout d'abord un ensemble de définitions.

Définition 196. *$\mathcal{B}$ est normalisé-complet donc pour tout terme t de $T(\Sigma)$, il existe un unique état q de $\mathcal{Q}$ tel que t appartienne à $\mathcal{L}(\mathcal{B}, q)$. Nous notons q_t cet état, et T_{fini} et T_{infini} les ensembles suivants :*

$$\begin{aligned} T_{fini} &= \{t \in T(\Sigma) \mid \mathcal{L}(\mathcal{B}, q_t) \text{ est fini}\}, \\ T_{infini} &= \{t \in T(\Sigma) \mid \mathcal{L}(\mathcal{B}, q_t) \text{ est infini}\}. \end{aligned}$$

Remarquons que $T(\Sigma) = T_{fini} \cup T_{infini}$. Nous construisons maintenant une nouvelle signature Γ en encodant les termes de T_{fini} dans les symboles de Γ. Soient $\Box$ un symbole n'appartenant pas à Σ et Γ la signature suivante :

$$\begin{aligned} \Gamma = \quad & \{f_{(u_1,\ldots,u_n)} \mid f \in \Sigma_n \text{ et } \forall i \in [n]\ u_i \in T_{fini} \cup \{\Box\}\} \\ & \setminus \{f_{(u_1,\ldots,u_n)} \mid f(u_1, \ldots, u_n) \in T_{fini}\}. \end{aligned}$$

L'arité d'un symbole de fonction $f_{(u_1,\ldots,u_n)}$ de Γ est la cardinalité de l'ensemble $\{i \in [n] \mid u_i = \Box\}$. Remarquons que pour toute constante a de Σ, a appartient à Γ si et seulement si a appartient à T_{infini}.

Soit l'automate $\mathcal{B}' = (\Gamma, \mathcal{Q}', \mathcal{Q}'_f, \Delta')$ défini par :

- $\mathcal{Q}' = \mathcal{Q} \setminus \{q \mid \mathcal{L}(\mathcal{B}, q) \text{ est fini}\}$;
- Définition de Δ' :

  Nous construisons les règles de Δ' en conservant dans une règle de Δ les états q du membre gauche pour lesquels $\mathcal{L}(\mathcal{B}, q)$ est infini et en remplaçant chaque autre état q par un terme de $\mathcal{L}(\mathcal{B}, q)$.

  Soient

  - $m, n \in \mathbb{N}$,
  - $f \in \Sigma_n$, $f_{(u_1,\ldots,u_n)} \in \Gamma_m$,
  - $q_1, \ldots, q_n \in \mathcal{Q}$, $s_1, \ldots, s_m \in \mathcal{Q}'$,
  - $q \in \mathcal{Q}'$,
  - $c \in EC'_n$, $c' \in EC'_m$,

  tels que

  - $\{k \in [n] \mid \mathcal{L}(\mathcal{B}, q_k) \text{ est infini}\} = \{k \in [n] \mid u_k = \Box\}$;
    Par la suite cet ensemble est notée I (de façon évidente $m = card(I)$) et nous définissons un ensemble d'indices $(j_k)_{k \in [m]}$ par quelquesoit k entier de $[m]$, j_k est le $k^{\text{ième}}$ élément de I.
  - $\forall k \in [n] \setminus I, q_k = q_{u_k}$;
  - $\forall k \in [m], q_{j_k} = s_k$;
  - $\forall k, l \in [m], (j_k \approx_c j_l) \Leftrightarrow (k \approx_{c'} l)$;
  - $\forall k, l \in [n] \setminus I, (k \approx_c l) \Leftrightarrow (u_k = u_l)$;

- $\forall k \in [m], \forall l \in [n] \setminus I, j_k \not\approx_c l$.

Alors

$$\Big(f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n)) \in \Delta\Big)$$

si et seulement si

$$\Big(f_{(u_1,\ldots,u_n)}(s_1(x_1), \ldots, s_m(x_m)) \xrightarrow{[c']} q(f_{(u_1,\ldots,u_n)}(x_1, \ldots, x_m)) \in \Delta'\Big).$$

Nous montrons que $\mathcal{B}'$ est un ATF normalisé-complet (proposition 197), tel que pour tout état q, le langage $\mathcal{L}(\mathcal{B}', q)$ est infini (proposition 198).

Proposition 197. *$\mathcal{B}'$ est un ATF normalisé-complet.*

Preuve : Par définition, $\mathcal{B}'$ est un ATF dont toutes les contraintes des règles sont pleines. Montrons maintenant que $\mathcal{B}'$ est déterministe. Soient

$$f_{(u_1,\ldots,u_n)}(s_1(x_1), \ldots, s_m(x_m)) \xrightarrow{[c']} q(f_{(u_1,\ldots,u_n)}(x_1, \ldots, x_m))$$
$$f_{(u_1,\ldots,u_n)}(s_1(x_1), \ldots, s_m(x_m)) \xrightarrow{[c']} q'(f_{(u_1,\ldots,u_n)}(x_1, \ldots, x_m))$$

deux règles de Δ'. En reprenant les éléments de la définition de Δ', les règles

$$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$$
$$f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q'(f(x_1, \ldots, x_n))$$

appartiennent à Δ. Puisque $\mathcal{B}$ est déterministe, nous avons $q = q'$, d'où $\mathcal{B}'$ est déterministe (remarque 110).

Nous considérons maintenant deux entiers m et n, un symbole de fonction $f_{(u_1,\ldots,u_n)}$ d'arité m, m états $s_1, \ldots, s_m$ de $\mathcal{Q}'$ et une contrainte pleine c' de EC'_m. Soient l'ensemble I, l'ensemble d'indices $(j_k)_{k\in[m]}$, les états $q_1, \ldots, q_n$ de $\mathcal{Q}$ et la contrainte pleine c de EC'_n définis dans la définition de Δ'.

Supposons que les égalités de c' sont satisfaites par la famille $(s_i)_{i\in[m]}$. Nous montrons qu'il existe un état q de $\mathcal{Q}'$ tel que la règle

$$f_{(u_1,\ldots,u_n)}(s_1(x_1), \ldots, s_m(x_m)) \xrightarrow{[c']} q(f_{(u_1,\ldots,u_n)}(x_1, \ldots, x_m))$$

appartienne à Δ'.

Nous montrons tout d'abord que les égalités de c sont satisfaites par la famille $(q_i)_{i\in[n]}$. Considérons deux entiers i et j de $[n]$ tels que $i \approx_c j$. De façon évidente, i et j appartiennent soit à I, soit à $[n] \setminus I$. Nous distinguons donc deux cas.

$i, j \in I$: Soient k et l appartenant à $[m]$ tels que $i = j_k$ et $j = j_l$. Nous avons $j_k \approx_c j_l$ donc $k \approx_{c'} l$. Puisque les égalités de c' sont satisfaites par la famille $(s_i)_{i\in[m]}$, nous avons $s_k = s_l$. D'où $q_{j_k} = q_{j_l}$ puisque $s_k = q_{j_k}$ et $s_l = q_{j_l}$. Finalement $q_i = q_j$.

$i, j \in [n] \setminus I$: Puisque $i \approx_c j$, nous avons $u_i = u_j$. D'où $q_i = q_j$ puisque $q_i = q_{u_i}$ et $q_j = q_{u_j}$.

Donc les égalités de c sont satisfaites par la famille $(q_i)_{i\in[n]}$. Puisque $\mathcal{B}$ est normalisé-complet, nous en déduisons qu'il existe un état q de $\mathcal{Q}$ tel que la règle $f(q_1(x_1),\ldots,q_n(x_n)) \xrightarrow{[c]} q(f(x_1,\ldots,x_n))$ appartienne à Δ. Nous montrons que q appartient à $\mathcal{Q}'$ en distinguant deux cas.

$m = 0$: Il existe une constante a de Γ telle que $f_{(u_1,\ldots,u_n)} = a$. Nous avons alors $q = q_a$ avec q_a appartenant à $\mathcal{Q}'$ puisque a appartient à T_{infini}.

$m > 0$: Il existe un entier i de $[m]$ tel que $u_i = \Box$. Nous avons alors de façon évidente $\mathcal{L}(\mathcal{B}, q)$ est infini puisque $\mathcal{L}(\mathcal{B}, q_i)$ est infini. Donc q appartient à $\mathcal{Q}'$.

Finalement q appartient à $\mathcal{Q}'$. Donc par définition de Δ', la règle

$$f_{(u_1,\ldots,u_n)}(s_1(x_1),\ldots,s_m(x_m)) \xrightarrow{[c']} q(f_{(u_1,\ldots,u_n)}(x_1,\ldots,x_m))$$

appartient à Δ'.

Finalement nous montrons que pour toute règle

$$f_{(u_1,\ldots,u_n)}(s_1(x_1),\ldots,s_m(x_m)) \xrightarrow{[c']} q(f_{(u_1,\ldots,u_n)}(x_1,\ldots,x_m))$$

de Δ', les égalités de c' sont satisfaites par la famille $(s_i)_{i\in[m]}$. Soit la règle

$$f(q_1(x_1),\ldots,q_n(x_n)) \xrightarrow{[c]} q(f(x_1,\ldots,x_n))$$

de Δ dont la règle précédente est issue (définition de Δ'). Alors

$$\begin{aligned} \forall k,l \in [m], (k \approx_{c'} l) &\Rightarrow (j_k \approx_c j_l) \\ &\Rightarrow q_{j_k} = q_{j_l} \text{ car } \mathcal{B} \text{ normalisé-complet} \\ &\Rightarrow s_k = s_l. \end{aligned}$$

Donc les égalités de c' sont satisfaites par la famille $(s_i)_{i\in[m]}$. Finalement $\mathcal{B}'$ est un ATF normalisé-complet. □

Proposition 198. *Pour tout état q de $\mathcal{Q}'$, le langage $\mathcal{L}(\mathcal{B}', q)$ est infini.*

Preuve : Soit un état q de $\mathcal{Q}'$. Alors $\mathcal{L}(\mathcal{B}, q)$ est infini, donc il existe une infinité de termes t de hauteur non nulle telle que t appartienne à $\mathcal{L}(\mathcal{B}, q)$. Soit t un tel terme. Il existe alors un symbole f de Σ d'arité n non nulle et $t_1,\ldots,t_n$ termes de $T(\Sigma)$ tels que $t = f(t_1,\ldots,t_n$.

Nous notons $I = \{k \in [n] \mid \mathcal{L}(\mathcal{B}, q_{t_k}) \text{ est infini}\}$. I est non vide. Nous définissons un ensemble d'indices $(j_i)_{i\in[card(I)]}$ par quelquesoit i entier de $[card(I)]$, j_i est le $i^{\text{ième}}$ élément de I.] Soit $(u_k)_{k\in[n]}$ le n-uplet défini par :

- $\forall k \in [n] \setminus I, u_k = t_k$, et
- $\forall k \in I, u_k = \Box$.

Nous pouvons montrer que le terme $f_{(u_1,\ldots,u_n)}(t_{j_1},\ldots,t_{j_{card(I)}})$ appartient à $\mathcal{L}(\mathcal{B}', q)$. Nous déduisons du fait qu'il existe une infinité de termes t que $\mathcal{L}(\mathcal{B}', q)$ est infini. □

Nous construisons maintenant un morphisme d'arbres linéaire Φ tel que $\Phi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B})$ (proposition 199) et $\Phi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$ (proposition 201).

Soit $\Phi : T(\Gamma) \to T(\Sigma)$ le morphisme d'arbres déterminé par l'application φ de Γ dans $T(\Sigma, \mathcal{X})$ définie par:

$$\forall f_{(u_1,\ldots,u_n)} \in \Gamma_m,\ \varphi(f_{(u_1,\ldots,u_n)})(x_1,\ldots,x_m) = f(t_1,\ldots,t_n)$$

avec $(t_k)_{k\in[n]}$ un n-uplet de termes de $T(\Gamma, \mathcal{X})$ tel que:

- Pour tout entier k de $[n]$, si u_k est un terme de $T(\Sigma)$ alors $t_k = u_k$, et
- Pour tout entier k de $[m]$, $t_{j_k} = x_k$ avec j_k le $k^{\text{ième}}$ élément de $\{i \in [n] \mid u_i = \Box\}$.

Remarquons que Φ est de façon évidente un morphisme linéaire.

Lemme 199. $\Phi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B})$.

Preuve : Nous déduisons des définitions de Γ et Φ que:

$$\Phi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B}) \setminus T_{fini}.$$

De plus pour tout terme t de $\mathcal{L}(\mathcal{B})$, q_t appartient à $\mathcal{Q}'_f$ donc $\mathcal{L}(\mathcal{B}, q_t)$ est infini. D'où t n'appartient pas à T_{fini}. Finalement $\Phi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B})$. □

Afin de montrer que nous avons $\Phi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$ (proposition 201), nous montrons dans un premier temps le lemme 200 caractérisant l'application Φ^{-1}.

Lemme 200. *Pour tout terme t de $T(\Sigma)$, nous avons:*

- $\Phi^{-1}(t) = \emptyset$ *si t est un terme de T_{fini}, et*
- $\Phi^{-1}(t)$ *est un singleton sinon.*

Preuve : La preuve se fait par induction sur la hauteur du terme t. Soit t un terme de $T(\Sigma)$.

t de hauteur 0 : Il existe une constante a de Σ telle que $t = a$.

Supposons que a appartienne à T_{fini}. Alors $\mathcal{L}(\mathcal{B}, q_a)$ est fini. D'où a n'appartient pas à Γ. Donc par définition de Φ, nous avons $\Phi^{-1}(a) = \emptyset$.

Supposons maintenant que a appartienne à T_{infini}. Alors $\mathcal{L}(\mathcal{B}, q_a)$ est infini. D'où a appartient à Γ. Donc par définition de Φ, nous avons $\Phi^{-1}(a) = \{a\}$.

Induction : Soit k un entier. Supposons que pour tout terme t de $T(\Sigma)$ de hauteur inférieure ou égale à k, nous avons $\Phi^{-1}(t) = \emptyset$ si t appartient à T_{fini}, sinon $\Phi^{-1}(t)$ est un singleton.

Soit t un terme de $T(\Sigma)$ de hauteur $k+1$. Il existe un symbole f de Σ d'arité n non nulle et des termes $t_1,\ldots,t_n$ de $T(\Sigma)$ tels que $t = f(t_1,\ldots,t_n)$. Nous notons $I = \{k \in [n] \mid \mathcal{L}(\mathcal{B}, q_{t_k})$ est infini$\}$.

Soit t' un terme de $\Phi^{-1}(\{t\})$. Il existe un symbole $f_{(u_1,\ldots,u_n)}$ de Γ d'arité m et des termes $t'_1,\ldots,t'_m$ de $T(\Gamma)$ tels que $t' = f_{(u_1,\ldots,u_n)}(t'_1,\ldots,t'_m)$ avec:

- Pour tout entier k de $[n]$, si u_k est un terme de $T(\Sigma)$ alors $u_k = t_k$, et

- Pour tout entier k de $[m]$, t'_k appartient à $\Phi^{-1}(t_{j_k})$ avec j_k le $k^{\text{ième}}$ élément de l'ensemble $\{i \in [n] \mid u_i = \Box\}$.

Montrons tout d'abord que $I = \{k \in [n] \mid u_k = \Box\}$. Par définitions de Γ et Φ, nous avons $I \subseteq \{k \in [n] \mid u_k = \Box\}$. Supposons qu'il existe un indice k de $[n] \setminus I$ tel que $u_k = \Box$. Alors $\mathcal{L}(\mathcal{B}, q_{t_k})$ est fini, d'où $\Phi^{-1}(t_k) = \emptyset$ par hypothèse d'induction. Donc pour l'indice i de $[m]$ tel que $j_i = k$, t'_i est indéfini. Il y a donc contradiction. Nous en déduisons que $\{k \in [n] \mid u_k = \Box\} \subseteq I$. Finalement $\{k \in [n] \mid u_k = \Box\} = I$. Donc $m = card(I)$.

Supposons maintenant que t appartienne à T_{fini}, c'est-à-dire que $\mathcal{L}(\mathcal{B}, q_t)$ est fini. Supposons de plus que m soit non nul. Soit k un entier de $[m]$. Alors $\mathcal{L}(\mathcal{B}, q_{t_{j_k}})$ est infini car j_k est un indice de I. Nous en déduisons que $\mathcal{L}(\mathcal{B}, q_t)$ est infini. Nous obtenons une contradiction d'où m est nul. Donc pour tout entier i de $[n]$, u_i appartient à T_{fini}. Donc $f_{(u_1,\ldots,u_n)}$ n'appartient pas à Γ puisque $t = f(t_1, \ldots, t_n)$ appartient à T_{fini}. Finalement t' n'existe pas et $\Phi^{-1}(t) = \emptyset$.

Supposons maintenant que t appartienne pas à T_{fini}, c'est-à-dire que t appartient à T_{infini}. Puisque $I = \{k \in [n] \mid u_k = \Box\}$, nous avons pour tout entier i de $[m]$, $\mathcal{L}(\mathcal{B}, q_{t_{j_i}})$ est infini, c'est-à-dire t_{j_i} appartient à T_{infini}. Donc pour tout entier i de $[m]$, t'_i est unique par hypothèse d'induction. Finalement $\Phi^{-1}(t)$ est un singleton.

□

Proposition 201. $\Phi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$.

Preuve : Nous déduisons tout d'abord que $\mathcal{L}(\mathcal{B}') \subseteq \Phi^{-1}(\mathcal{L}(\mathcal{B}))$ du lemme 199 et de la relation $\mathcal{L}(\mathcal{B}') \subseteq \Phi^{-1}(\Phi(\mathcal{L}(\mathcal{B}')))$.

Nous montrons maintenant que $\Phi^{-1}(\mathcal{L}(\mathcal{B})) \subseteq \mathcal{L}(\mathcal{B}')$. Pour cela, nous montrons la propriété suivante

$$\forall t \in T_{infini}, \forall t' \in \Phi^{-1}(t),\ t' \xrightarrow{*}_{\mathcal{B}'} q_t(t') \tag{1}$$

par induction sur la longueur de la dérivation $t \xrightarrow{*}_{\mathcal{B}} q_t(t)$.

Base : Soit t un terme de T_{infini} tel que $t \to_{\mathcal{B}} q_t(t)$. Il existe une constante a de Σ telle que $t = a$ et $a \to q_a$ appartient à Δ. Puisque $\mathcal{L}(\mathcal{B}, a)$ est infini, nous avons a appartient à Γ, $a \to q_a$ à Δ' et $\Phi^{-1}(t) = \{a\}$. Donc $a \to_{\mathcal{B}'} q_a(a)$.

Induction : Soit k un entier. Supposons que la propriété (1) soit vraie pour toute dérivation de longueur inférieure ou égale à k.

Soit t un terme de T_{infini} tel que $t \xrightarrow{*}_{\mathcal{B}} f(q_{t_1}(t_1), \ldots, q_{t_n}(t_n)) \xrightarrow{r}_{\mathcal{B}} q_t(t)$ par une dérivation de longueur $k+1$ avec $r := f(q_{t_1}(x_1), \ldots, q_{t_n}(x_n)) \xrightarrow{[c]} q_t(f(x_1, \ldots, x_n))$.

$\mathcal{L}(\mathcal{B}, q_t)$ est infini puisque t appartient à T_{infini}. Donc d'après la preuve du lemme 200, il existe un terme t' de $T(\Gamma)$ tel que $\Phi^{-1}(t) = \{t'\}$. De plus $t' = f_{(u_1,\ldots,u_n)}(t'_1, \ldots, t'_{card(I)})$ avec

- $I = \{k \in [n] \mid \mathcal{L}(\mathcal{B}, q_{t_k})$ est infini$\}$;
- $\forall i \in I, u_i = \Box$,
- $\forall i \in [n] \setminus I, u_i = t_i$;

- $\forall i \in [card(I)], u_{j_i} = \Box$ avec j_i le $i^{\text{ième}}$ élément de I;
- $\forall i \in [card(I)], \Phi^{-1}(t_{j_i}) = \{t'_i\}$.

Par hypothèse d'induction, nous avons pour tout entier i de $[card(I)]$, $t'_i \xrightarrow{*}_{\mathcal{B}'} q_{t_{j_i}}(t'_i)$, puisque t_{j_i} appartient à T_{infini}.

Par définition de l'automate $\mathcal{B}'$, la règle

$$r' := f_{(u_1,\ldots,u_n)}(q_{t_{j_1}}(x_{j_1}), \ldots, q_{t_{j_{card(I)}}}(x_{j_{card(I)}})) \xrightarrow{[c']} q_t(f_{(u_1,\ldots,u_n)}(x_{j_1}, \ldots, x_{j_{card(I)}}))$$

appartient à Δ' avec c' contrainte de $EC'_{card(I)}$ définie par tous entiers k et l de $[card(I)]$, $(j_k \approx_c j_l) \Leftrightarrow (k \approx_{c'} l)$. De plus :

$$\begin{aligned} \forall k, l \in [card(I)]\ j_k \approx_c j_l &\Rightarrow t_{j_k} = t_{j_l} \\ &\Rightarrow t'_k = t'_l \end{aligned}$$

On en déduit que $f_{(u_1,\ldots,u_n)}(t'_1, \ldots, t'_{card(I)}) \Vdash c'$. Donc

$$t' \xrightarrow{*}_{\mathcal{B}'} f_{(u_1,\ldots,u_n)}(q_{j_1}(t'_1), \ldots, q_{j_{card(I)}}(t'_{card(I)})) \xrightarrow{r'}_{\mathcal{B}'} q_t(t')$$

ce qui termine la preuve de la propriété (1).

Montrons maintenant que $\Phi^{-1}(\mathcal{L}(\mathcal{B})) \subseteq \mathcal{L}(\mathcal{B}')$. Soit t un terme de $\mathcal{L}(\mathcal{B})$. Alors q_t appartient à $\mathcal{Q}'_f$ et donc t est un terme de T_{infini}. Donc d'après la propriété (1), nous avons pour tout terme t' de $\Phi^{-1}(t)$, $t' \xrightarrow{*}_{\mathcal{B}'} q_t(t')$. Donc t' est un terme de $\mathcal{L}(\mathcal{B}')$ puisque l'ensemble des états finaux de $\mathcal{B}'$ est $\mathcal{Q}'_f$. D'où $\Phi^{-1}(\mathcal{L}(\mathcal{B})) \subseteq \mathcal{L}(\mathcal{B}')$.

Finalement $\Phi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$. □

Finalement nous montrons la proposition suivante :

Proposition 202. *$\mathcal{L}(\mathcal{B})$ est régulier si et seulement si $\mathcal{L}(\mathcal{B}')$ est régulier.*

Preuve : Puisque la classe des réguliers est close par morphismes d'arbres inverses (propriété 47) et que $\Phi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$ (proposition 201), nous avons $\mathcal{L}(\mathcal{B}')$ est régulier si $\mathcal{L}(\mathcal{B})$ est régulier.

De plus puisque la classe des réguliers est close par morphismes d'arbres linéaires (propriété 47) et que $\Phi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B})$ (proposition 199) avec Φ linéaire, nous avons $\mathcal{L}(\mathcal{B})$ est régulier si $\mathcal{L}(\mathcal{B}')$ est régulier. Finalement $\mathcal{L}(\mathcal{B})$ est régulier si et seulement si $\mathcal{L}(\mathcal{B}')$ est régulier. □

Nous avons vu au début de la section que notre langage de départ $\mathcal{L}$ est régulier si et seulement si $\mathcal{L}_2$ est régulier. Nous venons de prouver que $\mathcal{L}_2 = \mathcal{L}(\mathcal{B})$ est régulier si et seulement si $\mathcal{L}(\mathcal{B}')$ est régulier. Or $\mathcal{B}'$ est un *ATF* normalisé-complet tel que pour tout état q de $\mathcal{Q}'$, le langage $\mathcal{L}(\mathcal{B}', q)$ est infini (proposition 198).

Nous déduisons de tout cela que l'on peut toujours se ramener au cas "infini" c'est-à-dire que pour résoudre le problème de reconnaissabilité pour la classe *RATF*, nous pouvons supposer que le langage considéré est reconnu par un *ATF* normalisé-complet tel que pour

tout état, le langage associé à celui-ci soit infini. Nous allons donc nous placer dans ce cas-là pour le reste de la preuve du théorème 194.

Nous considérons à partir de maintenant un langage $\mathcal{L}$ reconnu par un automate à tests entre frères, $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$, normalisé-complet tel que pour tout état q de $\mathcal{Q}$, le langage $\mathcal{L}(\mathcal{A}, q)$ est infini.

3.3 Termes contraints

Dans le chapitre sur les automates d'arbres finis, nous citons le théorème de Myhill-Nerode (théorème 43) permettant de minimiser les automates. La méthode développée utilise une relation d'équivalence définie à partir de contextes. Nous effectuons ici une transformation similaire pour les *ATF*. Puisque les règles de ces automates contiennent des contraintes d'égalité et de différence entre frères, nous définissons des termes imposant des égalités et des différences entre termes frères (tests exprimés par des contraintes pleines). De tels termes sont appelés termes contraints. Une étiquette d'un terme contraint à une position p est la combinaison d'un symbole de fonction et d'une contrainte pleine c de sorte que les égalités de c soient satisfaites par les fils de l'étiquette. Les feuilles peuvent être des constantes, des états ou des occurrences d'une variable unique. Un terme contraint est défini de sorte qu'il posséde au moins une instance close pour laquelle la contrainte de tout nœud soit satisfaite. Les contraintes ne doivent donc pas imposer des égalités entre fils clos égaux.

Définition 203. *Soient x une variable de $\mathcal{X}$ et Σ' la signature définie par tout entier n, $\Sigma'_n = \{f_c \mid f \in \Sigma_n, c \in EC'_n\}$. Nous définissons également deux nouveaux ensembles de constantes en associant à tout état q de $\mathcal{A}$ une constante $q^{@}$: $\mathcal{Q}^{@}$ est l'ensemble de constantes $\{q^{@} \mid q \in \mathcal{Q}\}$ et $\mathcal{Q}_f^{@} = \{q^{@} \mid q \in \mathcal{Q}_f\}$.*

Un terme contraint *C sur $\Sigma \cup \mathcal{Q}$ est un terme de $T(\Sigma', \mathcal{Q}^{@} \cup \{x\})$ avec pour toute position p non feuille de C, $C(p) = f_c$ tel que :*

- *La famille $(C|_{pi})_{i \in [n]}$ satisfait les égalités de c et,*
- *c n'impose pas de différence entre termes égaux de $T(\Sigma')$ c'est-à-dire :*

$$\forall i, j \in [n],\ (C|_{pi} \in T(\Sigma') \text{ et } C|_{pi} = C|_{pj}) \Rightarrow (i \approx_c j).$$

Exemple 204. *Considérons f et g deux symboles de fonction d'arité deux, et q_1 et q_2 deux états. Le terme $f_c(g_{c'}(q_1^{@}, x), g_{c'}(q_2^{@}, x))$ avec $c := (1 = 2)$ et $c' := (1 \neq 2)$ n'est pas un terme contraint puisque $g_{c'}(q_1^{@}, x) \neq g_{c'}(q_2^{@}, x)$.*

Par contre, le terme $f_c(g_{c'}(q_1^{@}, x), g_{c'}(q_1^{@}, x))$ est un terme contraint.

Nous opérons deux types d'instanciation sur les termes contraints (exemple 205) :

Instanciation de x. Nous remplaçons toute occurrence de x dans un terme contraint C par un terme contraint C'. Le terme obtenu, noté $C[C']$, est de façon évidente un terme contraint si C' n'est pas un terme de $T(\Sigma')$.

Instanciation d'état. Pour un terme contraint C et un état q, nous remplaçons chaque occurrence de q par un terme contraint issu de $\mathcal{L}(\mathcal{A}, q)$ de sorte d'obtenir un nouveau terme contraint.

Notons que les occurences d'un même état q peuvent être remplaçées par des termes différents.

Exemple 205. *Nous reprenons les éléments de l'exemple 204 et notons C le terme contraint $f_c(g_{c'}(q_1^{@}, x), g_{c'}(q_1^{@}, x))$.*

Alors $C[q_1^{@}] = f_c(g_{c'}(q_1^{@}, q_1^{@}), g_{c'}(q_1^{@}, q_1^{@}))$ est une instance de C. Et si le langage reconnu par q_1 contient deux constantes a et b, le terme contraint $f_c(g_{c'}(a, b), g_{c'}(a, b))$ est une instance de $C[q_1^{@}]$.

Nous étendons maintenant la relation de réécriture $\rightarrow_{\mathcal{A}}$ aux termes contraints. Soit C un terme contraint sur Σ tel que $C = f_c(q_1^{@}, \ldots, q_n^{@})$ avec $q_1^{@}, \ldots, q_n^{@}$ constantes de $\mathcal{Q}^{@}$. Alors $C \rightarrow_{\mathcal{A}} s^{@}$ si et seulement si $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} s(f(x_1, \ldots, x_n))$ est une règle de $\mathcal{A}$.

La relation $\xrightarrow{*}_{\mathcal{A}}$ s'étend alors aux termes contraints (exemple 206). Nous avons en particulier $C[q^{@}] \xrightarrow{*}_{\mathcal{A}} s^{@}$ si $C = s^{@}$ ou $(C = x$ et $q^{@} = s^{@})$.

Exemple 206. *Soient la signature $\Sigma = \{f/2, g/1, a/0\}$ et l'ATF $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ avec $\mathcal{Q} = \{q_0, q\}$, $\mathcal{Q}_f = \{q_0\}$ et Δ constitué des règles suivantes:*

$$\begin{array}{rcl} a & \rightarrow & q_0(a) \\ g(q_0(x)) & \rightarrow & q_0(g(x)) \\ f(q_0(x), q_0(y)) & \xrightarrow{[1\neq 2]} & q(f(x, y)) \end{array}$$

Nous avons la dérivation suivante:

$$\begin{array}{rcl} f_{1\neq 2}(g(a), g(q_0^{@})) & \rightarrow_{\mathcal{A}} & f_{1\neq 2}(g(q_0^{@}), g(q_0^{@})) \\ & \xrightarrow{*}_{\mathcal{A}} & f_{1\neq 2}(q_0^{@}, q_0^{@}) \\ & \rightarrow_{\mathcal{A}} & q^{@} \end{array}$$

La définition des termes contraints et le fait que $\mathcal{A}$ est un automate normalisé-complet permet de montrer la proposition suivante :

Proposition 207. *Pour tout terme contraint C et tout état q, il existe un unique état s tel que $C[q^{@}] \xrightarrow{*}_{\mathcal{A}} s^{@}$.*

3.4 Minimisation

Dans cette section, nous utilisons les termes contraints afin d'étendre la transformation du théorème de Myhill-Nerode (théorème 43) aux *ATF*.

Nous définissons tout d'abord une relation d'équivalence $\equiv_{\mathcal{A}}$ sur l'ensemble des états de $\mathcal{A}$ (définition 208). Cette relation est calculable et nous donnons un algorithme pour sa construction (algorithme EQUIV). Nous montrons la correction de l'algorithme (proposition 212). Ensuite nous définissons l'automate $\mathcal{A}_m$ basé sur les classes d'équivalence de la relation $\equiv_{\mathcal{A}}$. Nous montrons que cet automate est un *ATF* normalisé-complet (proposition 214) équivalent à $\mathcal{A}$ (proposition 215).

Définition 208. *Soit $\equiv_{\mathcal{A}}$ la relation binaire définie sur $\mathcal{Q}$ par quelquesoit q_1 et q_2 états de $\mathcal{Q}$, $q_1 \equiv_{\mathcal{A}} q_2$ si pour tout terme contraint C sur $\Sigma \cup \mathcal{Q}$,*

$$\left(C[q_1^{@}] \xrightarrow{*}_{\mathcal{A}} s_1^{@} \in \mathcal{Q}_f^{@}\right) \Leftrightarrow \left(C[q_2^{@}] \xrightarrow{*}_{\mathcal{A}} s_2^{@} \in \mathcal{Q}_f^{@}\right).$$

La proposition 207 nous permet de montrer facilement que $\equiv_{\mathcal{A}}$ est une relation d'équivalence.

Algorithme EQUIV : Classes d'équivalence
Entrée : Un *ATF* normalisé-complet $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$
Début
  Soit $P := \{\mathcal{Q}_f, \mathcal{Q} \setminus \mathcal{Q}_f\}$ /* P est la relation d'équivalence de départ */
  Faire
    $P' = P$
    /* Affiner l'équivalence P' dans P */
    Si $q_1 P' q_2$ et pour tout terme contraint C de hauteur un,
      $C[q_1^{@}] \rightarrow_{\mathcal{A}} s_1^{@}$ et $C[q_2^{@}] \rightarrow_{\mathcal{A}} s_2^{@}$ avec $s_1 P' s_2$
    Alors $q_1 P q_2$
  Jusqu'à $P' = P$
Sortie : P l'ensemble des classes d'équivalence de $\equiv_{\mathcal{A}}$
Fin

Nous notons $\tilde{q}$ la classe d'équivalence d'un état q suivant P, l'ensemble calculé par l'algorithme EQUIV. Nous allons prouver que cet algorithme est correct c'est-à-dire que P est bien l'ensemble des classes d'équivalence de $\equiv_{\mathcal{A}}$ (proposition 212). Premièrement nous montrons que les classes d'équivalences suivant P sont compatibles avec les règles de l'automate $\mathcal{A}$ (corollaire 211). Pour cela, nous montrons que si l'on remplace, dans un membre gauche de règle, toutes les occurrences d'un état liées par des contraintes d'égalité par un état équivalent suivant P, on ne modifie pas la classe d'équivalence de l'état en membre droit (lemme 209).

Lemme 209. *Soient $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $f(s_1(x_1), \ldots, s_n(x_n)) \xrightarrow{[c]} s(f(x_1, \ldots, x_n))$ deux règles de $\mathcal{A}$ telles qu'il existe un entier j de $[n]$ tel que :*

- $q_j \in \tilde{s_j}$, *et*
- $\forall i \in [n]$, $(i \not\approx_c j) \Rightarrow (s_i = q_i)$.

Alors $\tilde{q} = \tilde{s}$.

Remarque 210. *$\mathcal{A}$ est normalisé-complet donc la famille $(q_i)_{i\in[n]}$ satisfait les égalités de c. D'où pour tout entier i de $[n]$, $(i \approx_c j) \Rightarrow (s_i = s_j)$.*

Preuve : Soient

$$f(q_1(x_1),\ldots,q_n(x_n)) \xrightarrow{[c]} q(f(x_1,\ldots,x_n))$$
$$f(s_1(x_1),\ldots,s_n(x_n)) \xrightarrow{[c]} s(f(x_1,\ldots,x_n))$$

deux règles de $\mathcal{A}$ telles qu'il existe un entier j de $[n]$ tel que q_j appartienne à $\tilde{s_j}$, et pour tout i de $[n]$, $(i \not\approx_c j) \Rightarrow (s_i = q_i)$.

Considérons le terme contraint C sur $\Sigma \cup \mathcal{Q}$ défini par :

- $tête(C) = f_c$;
- Pour tout entier i de $[n]$ si $i \approx_c j$ alors $C(i) = x$ sinon $C(i) = q_i^{@}$.

De façon évidente, pour tout i de $[n]$, $C[q_j^{@}]|_i = f_c(q_1^{@},\ldots,q_n^{@})|_i$ donc :

$$C[q_j^{@}] = f_c(q_1^{@},\ldots,q_n^{@}) \rightarrow_{\mathcal{A}} q^{@}.$$

Montrons maintenant que pour tout i de $[n]$, $C[s_j^{@}]|_i = f_c(s_1^{@},\ldots,s_n^{@})|_i$. Soit un entier i de $[n]$. Si $i \approx_c j$, alors $s_i = s_j$ et $C(i) = x$. Donc $C[s_j^{@}]|_i = s_j^{@} = s_i^{@} = f_c(s_1^{@},\ldots,s_n^{@})|_i$. Si $i \not\approx_c j$ alors $s_i = q_i$ et $C(i) = q_i^{@}$. Donc $C[s_j^{@}]|_i = q_i^{@} = s_i^{@} = f_c(s_1^{@},\ldots,s_n^{@})|_i$. Nous en déduisons que :

$$C[s_j^{@}] = f_c(s_1^{@},\ldots,s_n^{@}) \rightarrow_{\mathcal{A}} s^{@}.$$

De plus C est un terme contraint de hauteur un donc d'après l'algorithme EQUIV, nous avons $\tilde{q} = \tilde{s}$ puisque $\tilde{q_j} = \tilde{s_j}$. □

Puisque pour toute règle, la famille des états satisfait les égalités de la contrainte, nous pouvons montrer le corollaire 211 en utilisant le lemme précédent et en modifiant au fur et à mesure le membre gauche de la première règle pour arriver au membre gauche de la seconde.

Corollaire 211. *Soient $f(q_1(x_1),\ldots,q_n(x_n)) \xrightarrow{[c]} q(f(x_1,\ldots,x_n))$ et $f(s_1(x_1),\ldots,s_n(x_n)) \xrightarrow{[c]} s(f(x_1,\ldots,x_n))$ deux règles de $\mathcal{A}$ telles que pour tout j de $[n]$, $\tilde{q_j} = \tilde{s_j}$. Alors $\tilde{q} = \tilde{s}$.*

Les classes d'équivalences suivant P sont donc compatibles avec les règles de l'automate $\mathcal{A}$. Nous montrons maintenant que l'algorithme EQUIV est correct.

Proposition 212. *L'ensemble P calculé par l'algorithme EQUIV est l'ensemble des classes d'équivalence de $\equiv_{\mathcal{A}}$ c'est-à-dire*

$$\forall q, s \in \mathcal{Q} \quad (q \equiv_{\mathcal{A}} s) \Leftrightarrow (\tilde{q} = \tilde{s}).$$

Preuve : Premièrement, nous montrons que pour tous états q et s de $\mathcal{Q}$, nous avons $(\tilde{q} \neq \tilde{s}) \Rightarrow (q \not\equiv_{\mathcal{A}} s)$ en montrant la propriété suivante :

$$\forall q, s \in \mathcal{Q}\ \tilde{q} \neq \tilde{s} \Rightarrow \exists C \text{ terme contraint tel que } \left(C[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_{\#}^{@} \in \mathcal{Q}_f^{@} \ \Leftrightarrow\ C[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_{\#}^{@} \notin \mathcal{Q}_f^{@}\right) \tag{2}$$

par induction sur le pas de l'algorithme EQUIV où la relation $\tilde{q} \neq \tilde{s}$ apparaît. Soient q et s deux états de $\mathcal{Q}$ tels que $\tilde{q} \neq \tilde{s}$.

Base : La relation $\tilde{q} \neq \tilde{s}$ apparaît au départ de l'exécution de l'algorithme EQUIV. Donc par définition de P, nous avons q appartient à $\mathcal{Q}_f$ si et seulement si s n'appartient pas à $\mathcal{Q}_f$. Donc en considérant le terme contraint $C = x$, nous avons

$$\left(C[q^{@}] \xrightarrow{*}_{\mathcal{A}} q^{@} \in \mathcal{Q}_f^{@} \ \Leftrightarrow\ C[s^{@}] \xrightarrow{*}_{\mathcal{A}} s^{@} \notin \mathcal{Q}_f^{@}\right).$$

Induction : Soit k un entier. Supposons que la propriété (2) soit vrai pour toute relation $\tilde{q} \neq \tilde{s}$ apparaissant à un pas inférieur ou égal à k de l'exécution de l'algorithme EQUIV.

Supposons maintenant qu'une telle relation apparaisse au pas $k+1$. Puisque P' désigne l'ensemble des classes d'équivalence calculé au pas k, il existe un terme contraint C de hauteur un tel que $C[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_1^{@}$ et $C[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_1^{@}$ avec $\tilde{q_1} \neq \tilde{s_1}$ apparaissant à un pas inférieur ou égal à k. Par hypothèse d'induction, il existe un terme contraint C_1 tel que

$$\left(C_1[q_1^{@}] \xrightarrow{*}_{\mathcal{A}} q_{\#}^{@} \in \mathcal{Q}_f^{@} \ \Leftrightarrow\ C_1[s_1^{@}] \xrightarrow{*}_{\mathcal{A}} s_{\#}^{@} \notin \mathcal{Q}_f^{@}\right).$$

Soit le terme contraint $C_2 = C_1[C]$. Nous avons

$$\left(C_2[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_{\#}^{@} \in \mathcal{Q}_f^{@} \ \Leftrightarrow\ C_2[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_{\#}^{@} \notin \mathcal{Q}_f^{@}\right)$$

ce qui termine la preuve de la propriété (2). Nous en déduisons que pour tous états q et s de $\mathcal{Q}$ $(q \equiv_{\mathcal{A}} s) \Rightarrow (\tilde{q} = \tilde{s})$.

Nous montrons maintenant que pour tous états q et s de $\mathcal{Q}$ $(\tilde{q} = \tilde{s}) \Rightarrow (q \equiv_{\mathcal{A}} s)$. Tout d'abord, nous montrons la propriété suivante :

$$\forall q, s \in \mathcal{Q}\ \tilde{q} = \tilde{s} \Rightarrow \left(\forall C \text{ terme contraint } \begin{cases} C[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_{\#}^{@} \\ C[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_{\#}^{@} \end{cases} \Rightarrow \tilde{q_{\#}} = \tilde{s_{\#}})\right) \tag{3}$$

par induction sur la hauteur du terme contraint. Soient q et s deux états de $\mathcal{Q}$ tels que $\tilde{q} = \tilde{s}$ et C un terme contraint tel que $C[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_{\#}^{@}$ et $C[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_{\#}^{@}$.

C de hauteur 0. Nous distinguons trois cas.

Soit $C \in \Sigma'_0$: Il existe une constante a de Σ' telle que $C = a$. Alors $C[q^{@}] = C[s^{@}] = a$. Donc les règles $a \to q_{\#}(a)$ et $a \to s_{\#}(a)$ appartiennent à $\mathcal{A}$. Or $\mathcal{A}$ est déterministe puisque normalisé-complet d'où $q_{\#} = s_{\#}$. Finalement $\tilde{q_{\#}} = \tilde{s_{\#}}$.

Soit $C \in \mathcal{Q}^{@}$: Il existe un état q_1 de $\mathcal{Q}$ tel que $C = q_1^{@}$. Donc $C[q^{@}] = C[s^{@}] = q_1^{@}$ d'où $q_{\#} = s_{\#} = q_1$. Finalement $\tilde{q_{\#}} = \tilde{s_{\#}}$.

Soit $C = x : C[q^{@}] = q^{@}$ et $C[s^{@}] = s^{@}$ donc $q_{\#} = q$ et $s_{\#} = s$. Finalement $\tilde{q_{\#}} = \tilde{s_{\#}}$ puisque $\tilde{q} = \tilde{s}$.

Donc la propriété (3) est vraie pour le cas de base.

Induction. Soit k un entier. Supposons que la propriété (3) soit vraie pour tout terme contraint de hauteur inférieure ou égale à k.

Supposons que C soit de hauteur $k+1$. Il existe un symbole f de Σ d'arité n non nulle, une contrainte pleine c de EC'_n et n termes contraints $(C_i)_{i\in[n]}$ tels que $C = f_c(C_1, \ldots, C_n)$.

Il existe $q_1, \ldots, q_n$ et $s_1, \ldots, s_n$ états de $\mathcal{Q}$ tels que :

$$\begin{array}{lll} C[q^{@}] & \xrightarrow{*}_{\mathcal{A}} & f_c(q_1^{@}, \ldots, q_n^{@}) \rightarrow_{\mathcal{A}} q_{\#}^{@} \\ C[s^{@}] & \xrightarrow{*}_{\mathcal{A}} & f_c(s_1^{@}, \ldots, s_n^{@}) \rightarrow_{\mathcal{A}} s_{\#}^{@} \end{array}$$

Donc les règles

$$\begin{array}{lll} f(q_1(x_1), \ldots, q_n(x_n)) & \xrightarrow{[c]} & q_{\#}(f(x_1, \ldots, x_n)) \\ f(s_1(x_1), \ldots, s_n(x_n)) & \xrightarrow{[c]} & s_{\#}(f(x_1, \ldots, x_n)) \end{array}$$

appartiennent à Δ (l'ensemble des règles de $\mathcal{A}$). Or d'après l'hypothèse d'induction, pour tout entier i de $[n]$, nous avons $\tilde{q_i} = \tilde{s_i}$ puisque $C_i[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_i^{@}$ et $C_i[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_i^{@}$ avec C_i terme contraint de hauteur inférieure ou égale à k. Nous déduisons donc du corollaire 211 que $\tilde{q_{\#}} = \tilde{s_{\#}}$ ce qui termine la preuve de la propriété (3).

Au départ de l'exécution de l'algorithme EQUIV, $P = \{\mathcal{Q}_f, \mathcal{Q} \setminus \mathcal{Q}_f\}$, donc nous pouvons montrer que

$$\forall q \in \mathcal{Q}_f, \forall s \in \mathcal{Q}, (\tilde{q} = \tilde{s}) \Rightarrow (s \in \mathcal{Q}_f) \tag{4}$$

puisqu'à chaque pas de l'algorithme qPs. Nous déduisons de cette propriété et de la propriété (3) que pour tous états q et s de $\mathcal{Q}$ $(q \not\equiv_{\mathcal{A}} s) \Rightarrow (\tilde{q} \neq \tilde{s})$. □

Nous déduisons du lemme précédent que l'algorithme EQUIV calcule les classes d'équivalence de la relation $\equiv_{\mathcal{A}}$. Donc pour tout état q de $\mathcal{Q}$, $\tilde{q}$ est la classe d'équivalence de q suivant $\equiv_{\mathcal{A}}$.

Dans la suite de cette section, nous allons associer à l'automate $\mathcal{A}$, un ATF normalisé-complet $\mathcal{A}_m$, dit "minimisé", tel que :

- Les états de $\mathcal{A}_m$ sont les classes d'équivalence de la relation $\equiv_{\mathcal{A}}$;
- $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$;
- Pour tout état q de $\mathcal{A}_m$, $\mathcal{L}(\mathcal{A}_m, q)$ est infini;
- Pour tous états q et s de $\mathcal{A}_m$, $(q \equiv_{\mathcal{A}_m} s) \Leftrightarrow (q = s)$.

Remarque 213. *Nous parlons d'automate minimisé non en terme d'états mais en terme de contraintes c'est-à-dire que si le langage reconnu est régulier, les contraintes des règles de l'automate minimisé sont inutiles.*

Soit l'automate $\mathcal{A}_m = (\Sigma, \mathcal{Q}_m, \mathcal{Q}_{f_m}, \Delta_m)$ défini par :

- $\mathcal{Q}_m$ est l'ensemble des classes d'équivalence suivant $\equiv_{\mathcal{A}}$;
- $\mathcal{Q}_{f_m} = \{\tilde{q} \mid q \in \mathcal{Q}_f\}$;
- $\Delta_m = \{f(\tilde{q_1}(x_1), \ldots, \tilde{q_n}(x_n)) \xrightarrow{[c]} \tilde{q}(fx_1, \ldots, x_n) \mid \forall i \in [n]\, \exists q_{\#,i} \in \tilde{q_i}, \exists q_\# \in \tilde{q}$ tels que $f(q_{\#,1}(x_1), \ldots, q_{\#,n}(x_n)) \xrightarrow{[c]} q_\#(f(x_1, \ldots, x_n)) \in \Delta\}$.

Nous montrons tout d'abord que $\mathcal{A}_m$ est un ATF normalisé-complet (proposition 214) puis que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$ (proposition 215).

Proposition 214. *$\mathcal{A}_m$ est un ATF normalisé-complet.*

Preuve : Tout d'abord $\mathcal{A}_m$ est un automate à contraintes pleines. Premièrement, nous montrons que $\mathcal{A}_m$ est déterministe. Soient

$$f(\tilde{q_1}(x_1), \ldots, \tilde{q_n}(x_n)) \xrightarrow{[c]} \tilde{q}(f(x_1, \ldots, x_n))$$
$$f(\tilde{q_1}(x_1), \ldots, \tilde{q_n}(x_n)) \xrightarrow{[c]} \tilde{s}(f(x_1, \ldots, x_n))$$

deux règles de Δ_m. D'après la définition de Δ_m :

- $\forall i \in [n]\, \exists q_{\#,i} \in \tilde{q_i}, \exists q_\# \in \tilde{q}$ tels que $f(q_{\#,1}(x_1), \ldots, q_{\#,n}(x_n)) \xrightarrow{[c]} q_\#(f(x_1, \ldots, x_n))$ appartienne à Δ.
- $\forall i \in [n]\, \exists s_{\#,i} \in \tilde{q_i}, \exists s_\# \in \tilde{s}$ tels que $f(s_{\#,1}(x_1), \ldots, s_{\#,n}(x_n)) \xrightarrow{[c]} s_\#(f(x_1, \ldots, x_n))$ appartienne à Δ.

Pour tout entier i de $[n]$, $q_{\tilde{\#},i} = s_{\tilde{\#},i}$ donc d'après le corollaire 211, nous avons $q_{\tilde{\#}} = s_{\tilde{\#}}$. Finalement $\tilde{q} = \tilde{s}$ puisque $\tilde{q} = q_{\tilde{\#}}$ et $\tilde{s} = s_{\tilde{\#}}$. Donc $\mathcal{A}_m$ est déterministe d'après la remarque 110.

Nous considérons maintenant un symbole f de Σ d'arité n, n états $\tilde{q_1}, \ldots, \tilde{q_n}$ de Q_m et c une contrainte pleine de CE'_n tels que $(\tilde{q_i})_{i\in[n]}$ satisfait les égalités de c. Il existe alors une famille $(q_{\#,i})_{i\in[n]}$ d'états de $\mathcal{Q}$ telle que pour tout i de $[n]$, $q_{\#,i}$ appartient à $\tilde{q_i}$ et la famille $(q_{\#,i})_{i\in[n]}$ satisfait les contraintes d'égalité de c. Or $\mathcal{A}$ est normalisé-complet donc il existe une règle $f(q_{\#,1}(x_1), \ldots, q_{\#,n}(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ appartenant à Δ. Donc la règle $f(q_{\tilde{\#},1}(x_1), \ldots, q_{\tilde{\#},n}(x_n)) \xrightarrow{[c]} \tilde{q}(f(x_1, \ldots, x_n))$ appartient à Δ_m par définition de $\mathcal{A}_m$. De plus, pour tout i de $[n]$, nous avons $q_{\tilde{\#},i} = \tilde{q_i}$ donc $f(\tilde{q_1}(x_1), \ldots, \tilde{q_n}(x_n)) \xrightarrow{[c]} \tilde{q}(f(x_1, \ldots, x_n))$ appartient à Δ_m.

Finalement, nous considérons un symbole f de Σ d'arité n, n états $\tilde{q_1}, \ldots, \tilde{q_n}$ de Q_m et c une contrainte pleine de CE'_n tels que $(\tilde{q_i})_{i\in[n]}$ ne satisfait pas les égalités de c. Supposons qu'il existe une règle $f(\tilde{q_1}(x_1), \ldots, \tilde{q_n}(x_n)) \xrightarrow{[c]} \tilde{q}(f(x_1, \ldots, x_n))$ appartenant à Δ_m.

Alors par définition de $\mathcal{A}_m$, il existe un état $q_{\#,i}$ de $\tilde{q}_i$ pour tout i de $[n]$ et un état $q_\#$ de $\tilde{q}$ tels que la règle $f(q_{\#,1}(x_1),\ldots,q_{\#,n}(x_n)) \xrightarrow{[c]} q_\#(f(x_1,\ldots,x_n))$ appartienne à Δ. Puisque deux classes d'équivalence différentes sont disjointes, la famille $(q_{\#,i})_{i\in[n]}$ ne satisfait pas les égalités de c. Donc nous obtenons une contradiction puisque $\mathcal{A}$ est normalisé-complet. Nous en déduisons qu'il n'existe pas de règle $f(\tilde{q}_1(x_1),\ldots,\tilde{q}_n(x_n)) \xrightarrow{[c]} \tilde{q}(f(x_1,\ldots,x_n))$ appartenant à Δ_m.

Finalement $\mathcal{A}_m$ est un ATF normalisé-complet. □

Proposition 215. $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$.

Preuve : Nous montrons tout d'abord que $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}_m)$. Pour cela, nous montrons que :

$$\forall t \in T(\Sigma), \forall q \in Q,\ (t \xrightarrow{*}_{\mathcal{A}} q(t)) \Rightarrow (t \xrightarrow{*}_{\mathcal{A}_m} \tilde{q}(t)) \tag{5}$$

par induction sur la hauteur de t.

Base. Si t est de hauteur 0, il existe une constante a de Σ telle que $t = a$. Alors $a \to q(a)$ est une règle de Δ. Donc par définition, la règle $a \to \tilde{q}(a)$ appartient à Δ_m. Donc $a \to_{\mathcal{A}_m} \tilde{q}(a)$.

Induction. Soit k un entier. Supposons que la propriété (5) soit vraie pour tout terme de hauteur inférieure ou égale à k.

Soient t un terme de $T(\Sigma)$ de hauteur $k+1$ et q un état de $\mathcal{A}$ tel que

$$t \xrightarrow{*}_{\mathcal{A}} f(q_1(t_1),\ldots,q_n(t_n)) \xrightarrow{r}_{\mathcal{A}} q(t)$$

avec $r := f(q_1(x_1),\ldots,q_n(x_n)) \xrightarrow{[c]} q(f(x_1,\ldots,x_n))$.

Pour tout entier i de $[n]$, $t_i \xrightarrow{*}_{\mathcal{A}} q_i(t_i)$ donc par hypothèse d'induction, $t_i \xrightarrow{*}_{\mathcal{A}_m} \tilde{q}_i(t_i)$. Donc $t \xrightarrow{*}_{\mathcal{A}_m} f(\tilde{q}_1(t_1),\ldots,\tilde{q}_n(t_n))$.

De plus $f(\tilde{q}_1(x_1),\ldots,\tilde{q}_n(x_n)) \xrightarrow{[c]} \tilde{q}(f(x_1,\ldots,x_n))$ est une règle de Δ_m et $f(t_1,\ldots,t_n)$ satisfait c donc $f(\tilde{q}_1(t_1),\ldots,\tilde{q}_n(t_n)) \xrightarrow{r}_{\mathcal{A}_m} \tilde{q}(t)$.

Ceci termine la preuve de la propriété (5). Nous en déduisons facilement que $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}_m)$.

Montrons maintenant que $\mathcal{L}(\mathcal{A}_m) \subseteq \mathcal{L}(\mathcal{A})$. Pour cela, nous démontrons que :

$$\forall t \in T(\Sigma), \forall q \in Q,\ (t \xrightarrow{*}_{\mathcal{A}_m} \tilde{q}(t)) \Rightarrow (\exists s \in \tilde{q} \text{ tel que } t \xrightarrow{*}_{\mathcal{A}} s(t)) \tag{6}$$

Soient t un terme de $T(\Sigma)$ et q un état de Q tels que $t \xrightarrow{*}_{\mathcal{A}_m} \tilde{q}(t)$. $\mathcal{A}$ étant normalisé-complet, il existe un état s de Q tel que $t \xrightarrow{*}_{\mathcal{A}} s(t)$. Donc $t \xrightarrow{*}_{\mathcal{A}_m} \tilde{s}(t)$ d'après la propriété (5). Or $\mathcal{A}_m$ est déterministe car normalisé-complet (proposition 214). Donc $\tilde{q} = \tilde{s}$ ce qui termine la preuve de la propriété (6).

Nous déduisons alors de la propriété (4) que $\mathcal{L}(\mathcal{A}_m) = \mathcal{L}(\mathcal{A})$. □

L'automate $\mathcal{A}_m$ que nous avons défini est donc un ATF normalisé complet reconnaissant le même langage que l'automate $\mathcal{A}$ de départ.

Pour terminer cette section, nous montrons que $\mathcal{A}_m$ satisfait les deux propriétés suivantes.

Proposition 216. *Pour tout état q de $\mathcal{A}_m$, $\mathcal{L}(\mathcal{A}_m, q)$ est infini et pour tous états q et s de $\mathcal{A}_m$, $(q \equiv_{\mathcal{A}_m} s) \Leftrightarrow (q = s)$.*

Preuve : Soient q un état de $\mathcal{A}_m$ et $q_\#$ un état de $\mathcal{Q}$ tel que $\tilde{q_\#} = q$. D'après la propriété (5), $\mathcal{L}(\mathcal{A}, q_\#) \subseteq \mathcal{L}(\mathcal{A}_m, q)$. Or le langage $\mathcal{L}(\mathcal{A}, q_\#)$ est infini par définition de $\mathcal{A}$ d'où $\mathcal{L}(\mathcal{A}_m, q)$ infini.

Si nous appliquons l'algorithme EQUIV à l'automate $\mathcal{A}_m$, nous montrons facilement que l'ensemble des classes d'équivalence suivant la relation $\equiv_{\mathcal{A}_m}$ est l'ensemble des états de $\mathcal{A}_m$. Donc pour tous états q et s de $\mathcal{A}_m$, $(q \equiv_{\mathcal{A}_m} s) \Leftrightarrow (q = s)$. □

3.5 Caractérisation

Dans cette section, nous montrons que le langage $\mathcal{L}$ est régulier si et seulement si l'automate minimisé n'est pas "sensible aux contraintes" c'est-à-dire que deux règles dont les membres gauches diffèrent uniquement par leurs contraintes ont même membre droit (théorème 217).

D'après les propositions 214, 215 et 216, nous pouvons supposer que $\mathcal{A}$ normalisé-complet, que pour tout état q de $\mathcal{A}$, $\mathcal{L}(\mathcal{A}, q)$ est infini, et que pour tous états q et s de $\mathcal{A}$, $(q \equiv_{\mathcal{A}} s) \Leftrightarrow (q = s)$.

Théorème 217. *Le langage $\mathcal{L}(\mathcal{A})$ est régulier si et seulement si pour toute paire de règles $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c']} s(f(x_1, \ldots, x_n))$ de Δ, $q = s$.*

Pour la preuve de ce théorème, nous avons besoin de la construction suivante. Nous montrons que l'on peut "instancier" tout terme contraint sur $\Sigma \cup \mathcal{Q}$ de sorte d'obtenir un terme contraint sur Σ en remplaçant chaque état q par le terme contraint associé à un terme d'un sous-ensemble infini de $\mathcal{L}(\mathcal{A}, q)$ (définition 218, lemme 219 et proposition 220).

Définition 218. *Soit C un terme contraint. Nous notons :*

- *$\mathcal{VP}os(C)$ l'ensemble des positions de C où une instance de x apparaît, c'est-à-dire $\mathcal{VP}os(C) = \{p \in \mathcal{P}os(C) \mid C(p) = x\}$,*
- *$\mathcal{SP}os(C)$ l'ensemble des positions de C où une instance d'un état apparaît, c'est-à-dire $\mathcal{SP}os(C) = \{p \in \mathcal{P}os(C) \mid C(p) \in \mathcal{Q}\}$, et*
- *Pour tout état q de $\mathcal{Q}$, $\mathcal{SP}os(C)(q)$ l'ensemble des positions de C où une occurrence de q apparaît, c'est-à-dire $\mathcal{SP}os(C)(q) = \{p \in \mathcal{SP}os(C) \mid C(p) = q\}$.*

Lemme 219. *Soient C un terme contraint sur $\Sigma \cup \mathcal{Q}$, q un état de $\mathcal{Q}$ et T_q un ensemble infini inclus dans $\mathcal{L}(\mathcal{A}, q)$. Alors il existe un terme contraint C' sur $\Sigma \cup \mathcal{Q} \setminus \{q\}$ tel que :*

- *Pour toute position p de $\mathcal{P}os(C) \setminus \mathcal{SP}os(C)(q)$, $C'(p) = C(p)$ et*

- *Chaque occurrence de l'état q est remplacée par le terme contraint associé à un terme de T_q, c'est-à-dire pour toute position p de $\mathcal{SP}os(C)(q)$, il existe un terme t de T_q tel que $C'|_p = lab_t$.*

Où pour tout terme t clos, lab_t désigne le terme contraint sur Σ associé à t, c'est-à-dire :

- *Pour toute position non feuille p de t, si $t(p) = f$ alors $lab_t(p) = f_c$ avec c la contrainte pleine satisfaite par la famille $(t|_{pi})_{i\in[n]}$, et*
- *Pour toute position feuille p de t, $lab_t(p) = t(p)$.*

Preuve : Soient C un terme contraint sur $\Sigma \cup \mathcal{Q}$, q un état de $\mathcal{Q}$ et T_q un ensemble infini inclus dans $\mathcal{L}(\mathcal{A}, q)$. Remarquons que T_q est bien défini puisque $\mathcal{L}(\mathcal{A}, q)$ est infini.

Tout d'abord, nous déduisons des contraintes pleines des positions de C, une conjonction $c_{\mathcal{SP}os(C)(q)}$ de contraintes égalitaires et diségalitaires sur les positions de C étiquetées par q telle que pour toute paire (p, p') de positions de $\mathcal{SP}os(C)(q)$, soit la contrainte $p = p'$ soit la contrainte $p \neq p'$ appartienne à $c_{\mathcal{SP}os(C)(q)}$.

En fait, nous considérons les positions de C étiquetées par q et nous exprimons toutes les égalités entre ces positions imposées par les contraintes de C. Ensuite pour toute paire de positions non liées par une égalité, nous choisissons d'imposer une contrainte diségalitaire.

Formellement pour toute position p de C telle que $C(p) = f_c$ appartenant à Σ', nous notons $cont_C(p)$ la contrainte pleine c, et nous notons $c_{\mathcal{SP}os(C)(q)}$ la conjonction de contraintes définie par :

(i) Nous exprimons toute les égalités imposées par les contraintes de C c'est-à-dire pour toute position p de C, tous entiers i et j tels que $i \approx_{cont_C(p)} j$ et pour tout γ tel que $pi\gamma$ appartiennent à $\mathcal{SP}os(C)(q)$:

$$(pi\gamma = pj\gamma) \in c_{\mathcal{SP}os(C)(q)}.$$

(ii) Nous faisons la clôture transitive de la contrainte trouvée au point précédent afin d'exprimer toutes les égalités :

$$(p_1 = p_2 \wedge p_2 = p_3) \in c_{\mathcal{SP}os(C)(q)} \Rightarrow (p_1 = p_3) \in c_{\mathcal{SP}os(C)(q)}.$$

(iii) Nous imposons des contraintes de différences entre les positions non reliées par des égalités :

$$\forall p, p' \in \mathcal{SP}os(C)(q), p \neq p',\ (p = p') \notin c_{\mathcal{SP}os(C)(q)} \Rightarrow (p \neq p') \in c_{\mathcal{SP}os(C)(q)}.$$

Puisque T_q est infini, nous pouvons montrer qu'il existe une famille $(t_p)_{p\in\mathcal{SP}os(C)(q)}$ de termes de T_q telles que :

1. La famille $(t_p)_{p\in\mathcal{SP}os(C)(q)}$ satisfait la contrainte $c_{\mathcal{SP}os(C)(q)}$.
2. Pour toute position p de $\mathcal{SP}os(C)(q)$, t_p est de hauteur strictement plus grande que la hauteur de C et que la hauteur des termes de l'ensemble $\{t_{p'} \mid p' \in \mathcal{SP}os(C)(q)$ de longueur strictement plus petite que la longueur de $p\}$.

Soit C' le terme de $T(\Sigma')$ défini comme suit :

- Pour toute position p de $\mathcal{P}os(C) \setminus \mathcal{SP}os(C)(q)$, $C'(p) = C(p)$;

- Pour toute position p de $\mathcal{SP}os(C)(q)$, $C'|_p = lab_{t_p}$.

Nous montrons maintenant que C' est un terme contraint. Soit p une position non feuille de C'. Il existe un entier n non nul et une contrainte pleine c de EC'_n tels que $cont_{C'}(p) = c$. Soient i et j deux entier de $[n]$. Nous distinguons deux cas suivant que $i \approx_c j$ ou non.

Si $i \approx_c j$: Nous avons $C|_{pi} = C|_{pj}$ puisque C est un terme contraint et nous devons vérifié que $C'|_{pi} = C'|_{pj}$.

Si $C|_{pi}$ ne contient pas d'occurrence de q, alors $C'|_{pi} = C|_{pi}$ et $C'|_{pj} = C|_{pj}$. D'où $C'|_{pi} = C'|_{pj}$. Sinon, nous avons:

$$\begin{array}{lll} \forall\gamma,\ pi\gamma \in \mathcal{SP}os(C)(q) & \rightarrow & (pi\gamma = pj\gamma) \in c_{\mathcal{SP}os(C)(q)} \quad \text{(point (i))} \\ & \Rightarrow & t_{pi\gamma} = t_{pj\gamma} \quad \text{(point 1)} \\ & \Rightarrow & lab_{t_{pi\gamma}} = lab_{t_{pj\gamma}} \\ & \Rightarrow & C'|_{pi\gamma} = C'|_{pj\gamma} \end{array}$$

Nous en déduisons que $C'|_{pi} = C'|_{pj}$.

Si $i \not\approx_c j$: Nous devons vérifié que $C'|_{pi} \neq C'|_{pj}$ si $C'|_{pi}$ et $C'|_{pj}$ sont des termes de $T(\Sigma')$.

Supposons que $C'|_{pi}$ et $C'|_{pj}$ soient des termes de $T(\Sigma')$. Nous distinguons trois cas.

$C|_{pi}$ et $C|_{pj}$ ne contiennent pas d'occurrences de q : Alors $C|_{pi} = C'|_{pi}$ et $C|_{pj} = C'|_{pj}$ donc $C|_{pi}$ et $C|_{pj}$ sont des termes de $T(\Sigma')$. Nous en déduisons que $C|_{pi} \neq C|_{pj}$ puisque C est un terme contraint. D'où $C'|_{pi} \neq C'|_{pj}$.

Soit $C|_{pi}$, soit $C|_{pj}$ contient au moins une occurrence de q : Supposons que $C|_{pi}$ contienne au moins une occurrence de q et que $C|_{pj}$ n'en contienne aucune (le cas réciproque se traitant de façon similaire). Nous avons donc $C|_{pj} = C'|_{pj}$.

Soit γ une position telle que $pi\gamma$ soit une position de $\mathcal{SP}os(C)(q)$. D'après le point 2, la hauteur de $t_{pi\gamma}$, donc la hauteur de $lab_{t_{pi\gamma}}$, est strictement plus grande que la hauteur de C. Donc la hauteur de $C'|_{pi}$ est strictement plus grande que celle de $C|_{pj}$. Nous en déduisons que $C|_{pi} \neq C|_{pj}$.

$C|_{pi}$ et $C|_{pj}$ contiennent au moins une occurrence de q : Soit γ la plus petite position telle que $pi\gamma$ ou $pj\gamma$ soit une position de $\mathcal{SP}os(C)(q)$.

Supposons que $pi\gamma$ et $pj\gamma$ soient des position de $\mathcal{SP}os(C)(q)$. Alors par construction, la contrainte $c_{\mathcal{SP}os(C)(q)}$ contient la contrainte $(pi\gamma \neq pj\gamma)$. Donc $t_{pi\gamma} \neq t_{pj\gamma}$ par le point 1. D'où $lab_{t_{pi\gamma}} \neq lab_{t_{pj\gamma}}$ et finalement $C|_{pi} \neq C|_{pj}$.

Supposons maintenant que $pi\gamma$ soit une position de $\mathcal{SP}os(C)(q)$ et que $pj\gamma$ ne le soit pas (le cas réciproque se traitant de façon similaire). Si $pj\gamma$ n'est pas une position de C, alors par définitions de C' et γ, $pj\gamma$ n'est pas une position de C'. Donc $C|_{pi} \neq C|_{pj}$.

Si $pj\gamma$ est une position de C et $C'|_{pj\gamma}$ ne contient pas d'occurrence de q, alors $lab_{t_{pi\gamma}}$ est de hauteur strictement plus grande que celle de $C'|_{pj\gamma}$ par le point 2. Donc $C|_{pi} \neq C|_{pj}$.

Enfin si $pj\gamma$ est une position de C et il existe une position γ' non nulle telle que $pj\gamma\gamma'$ soit une position de $\mathcal{SP}os(C)(q)$. Alors la hauteur de $lab_{t_{pj\gamma\gamma'}}$ est strictement plus grande que celle de $lab_{t_{pi\gamma}}$ par le point 2. Donc $C|_{pi} \neq C|_{pj}$.

Nous en déduisons que C' est un terme contraint sur $\Sigma \cup \mathcal{Q} \setminus \{q\}$. □

Nous déduisons facilement par itération du lemme précédent la proposition suivante.

Proposition 220. *Soient C un terme contraint sur $\Sigma \cup \mathcal{Q}$ et pour tout état q de $\mathcal{Q}$, T_q un ensemble infini inclus dans $\mathcal{L}(\mathcal{A}, q)$. Alors il existe un terme contraint C' sur Σ, appelée* instance d'états *de C sur $(T_q)_{q\in\mathcal{Q}}$, tel que :*

- *Pour toute position p de $\mathcal{P}os(C) \setminus \mathcal{SP}os(C)$, $C'(p) = C(p)$ et*
- *Chaque occurrence d'un état q est remplacée par le terme contraint associé à un terme de T_q, c'est-à-dire pour tout état q de $\mathcal{Q}$ et toute position p de $\mathcal{SP}os(C)(q)$, il existe un terme t de T_q tel que $C'|_p = lab_t$.*

Remarquons que si C' est une instance d'états d'un terme contraint C sur $\Sigma \cup \mathcal{Q}$, nous avons

$$\forall q \in \mathcal{Q},\ (C[q^{@}] \xrightarrow{*}_{\mathcal{A}} s^{@} \Rightarrow C'[q^{@}] \xrightarrow{*}_{\mathcal{A}} s^{@}) \tag{7}$$

Montrons maintenant que la condition du théorème 217 est nécessaire.

Proposition 221. *Si Δ contient deux règles $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c']} s(f(x_1, \ldots, x_n))$ telles que $c \neq c'$ et $q \neq s$, alors $\mathcal{L}(\mathcal{A})$ n'est pas régulier.*

Preuve : Supposons qu'il existe deux règles $r := f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $r' := f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c']} s(f(x_1, \ldots, x_n))$ telles que $c \neq c'$ et $q \neq s$. Supposons de plus que $\mathcal{L}(\mathcal{A})$ est régulier. Alors d'après le théorème 40, il existe un AAS, $\mathcal{B} = (\Sigma, \mathcal{Q}_b, \mathcal{Q}_{f,b}, \Delta_b)$, déterministe et complet tel que $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$. Puisque $\mathcal{B}$ est un AAS, nous pouvons montrer la propriété suivante :

$$\begin{gathered}\forall m \in \mathbb{N}_+, \forall C \in \mathcal{C}^m(\Sigma), \forall q \in \mathcal{Q}_b, \forall (t_i)_{i\in[m]} \in \mathcal{L}(\mathcal{B}, q), \forall (t'_i)_{i\in[m]} \in \mathcal{L}(\mathcal{B}, q) \\ C[t_1, \ldots, t_m] \in \mathcal{L}(\mathcal{B}) \Leftrightarrow C[t'_1, \ldots, t'_m] \in \mathcal{L}(\mathcal{B})\end{gathered} \tag{8}$$

Nous donnons tout d'abord, l'idée de la preuve. Nous construisons deux termes, nommés t_1 et t_2, tels que

- $t_1 \in \mathcal{L}(\mathcal{A})$ et $t_2 \notin \mathcal{L}(\mathcal{A})$,
- t_1 et t_2 ne diffèrent qu'à certaines positions p où
  - $t_1(p) = t_2(p) = f$,
  - Les fils de $t_1(p)$ satisfont la contrainte c, et
  - Les fils de $t_2(p)$ satisfont la contrainte c'.

Ensuite, nous déduisons de t_1 et t_2, un terme linéaire C'_g, intuitivement le préfixe commun de t_1 et t_2, tel qu'il existe un état $q_{\mathcal{B}}$ de $\mathcal{B}$, deux familles $(u_i)_{i\in[m]}$ et $(u'_i)_{i\in[m]}$ de termes de $\mathcal{L}(\mathcal{B}, q_{\mathcal{B}})$ tels que $C'_g[(u_i)]$ appartienne à $\mathcal{L}(\mathcal{A})$ mais pas $C'_g[(u'_i)]$.

Ce dernier point contredira la propriété (8). Nous en déduirons que notre hypothèse "$\mathcal{L}(\mathcal{A})$ est régulier" est fausse.

Puisque c et c' sont des contraintes pleines de EC'_n, nous obtenons en scindant tous les ensembles de c et c' en des singletons, la contrainte pleine c'' constituée uniquement de différences. Donc il existe une famille $(c_i)_{i\in[l]}$ (respectivement $(c'_j)_{j\in[l']}$) de contraintes pleines de EC'_n telle que $c_1 = c$, $c_l = c''$ (respectivement $c'_1 = c'$, $c'_{l'} = c''$) et pour tout i de $[l-1]$ (respectivement $[l'-1]$) c_i et c_{i+1} (respectivement c'_i et c'_{i+1}) ne différent que par la séparation d'un ensemble en deux. Puisque $\mathcal{A}$ est normalisé-complet, il existe pour toute contrainte c^t définie ci-dessus, une règle de membre gauche $f(q_1(x_1),\ldots,q_n(x_n))[c^t]$. Nous pouvons donc supposer sans perte de généralité que les contraintes c et c' ne différent que par la séparation d'un ensemble en deux, c'est-à-dire qu'il existe un entier m et des sous-ensembles $E_1,\ldots,E_m$, I et J de $[n]$ tels que

$$\begin{aligned} c &= (E_1,\ldots,E_m, I\cup J),\ card(c) = k+1;\\ c' &= (E_1,\ldots,E_m, I, J),\ card(c') = k+2. \end{aligned}$$

D'après les hypothèses faites sur $\mathcal{A}$ au début de cette section, nous avons $q \not\equiv_{\mathcal{A}} s$ puisque $q \neq s$. Donc il existe un terme contraint C sur $\Sigma\cup\mathcal{Q}$ tel que

$$\left(C[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_1^{@} \in \mathcal{Q}_f^{@}\right) \Leftrightarrow \left(C[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_1^{@} \notin \mathcal{Q}_f^{@}\right).$$

Nous supposons que $q_1^{@}$ appartienne à $\mathcal{Q}_f^{@}$ et donc que $s_1^{@}$ n'appartienne pas à $\mathcal{Q}_f^{@}$. D'après la proposition 219, il existe une instance d'états $\bar{C}$ de C sur $(\mathcal{L}(\mathcal{A},q'))_{q'\in\mathcal{Q}}$. Nous déduisons de la propriété (7) que $\bar{C}[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_1^{@}$ et $\bar{C}[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_1^{@}$.

Par les hypothèses faites sur c, il existe un état q_I de $\mathcal{Q}$ tel que $q_i = q_I$ pour tout entier i de $I\cup J$. Nous considérons le terme contraint $F_1 = f_c(s_1,\ldots,s_n)$ tel que pour tout entier k de $[n]$:

- Si k appartient à $I\cup J$, alors $s_k = x$;
- Sinon $s_k = q_k^{@}$.

D'après la proposition 219, il existe une instance d'états F'_1 de F_1 sur $(\mathcal{L}(\mathcal{A},q))_{q\in\mathcal{Q}}$. Donc $F'_1 = f_c(v_1,\ldots,v_n)$ avec pour tout entier k de $[n]$:

- Si k appartient à $I\cup J$, alors $v_k = x$;
- Sinon v_k est un terme contraint obtenu à partir d'un terme de $\mathcal{L}(\mathcal{A},q_k)$.

Puisque $F_1[q_I^{@}] \xrightarrow{*}_{\mathcal{A}} q^{@}$, nous avons $F'_1[q_I^{@}] \xrightarrow{*}_{\mathcal{A}} q^{@}$ d'après la propriété (7). Soit C_1 le terme contraint $\bar{C}[F'_1[q_I^{@}]]$. Alors $C_1 \xrightarrow{*}_{\mathcal{A}} \bar{C}[q^{@}] \xrightarrow{*}_{\mathcal{A}} q_1^{@}$.

Soit F'_2 le terme $f_{c'}(v_1,\ldots,v_n)$. Par les hypothèses faites sur c et c', nous montrons facilement que F'_2 est un terme contraint. De plus, $F'_2[q_I^{@}] \xrightarrow{*}_{\mathcal{A}} f_{c'}(q_1^{@},\ldots,q_n^{@}) \xrightarrow{*}_{\mathcal{A}} s^{@}$.

De manière similaire à ci-dessus, nous notons C_2 le terme contraint $\bar{C}[F'_2[q_I^{@}]]$. Notons que les termes de $T(\Sigma,\{x\})$ obtenues à partir de F'_1 et F'_2 (respectivement C_1 et C_2) en supprimant les contraintes des nœuds sont les mêmes. Finalement nous avons $C_2 \xrightarrow{*}_{\mathcal{A}} \bar{C}[s^{@}] \xrightarrow{*}_{\mathcal{A}} s_1^{@}$.

Notons que C_1 et C_2 sont des termes contraints sur $\Sigma\cup\{q_I\}$ sans occurrence de x.

Puisque nous avons supposé que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$ et que $\mathcal{L}(\mathcal{A},q_I)$ est infini par définition de $\mathcal{A}$, il existe un état $q_{\mathcal{B}}$ de $\mathcal{B}$ tel que le langage $T = \mathcal{L}(\mathcal{A},q_I)\cap\mathcal{L}(\mathcal{B},q_{\mathcal{B}})$ est infini.

Or C_1 est un terme contraint sur $\Sigma \cup \{q_I\}$ sans occurrence de x. Donc d'après la proposition 219, il existe une instance d'états C'_1 de C_1 sur T, c'est-à-dire qu'il existe une famille $(u_p)_{p \in \mathcal{SP}os(C_1)(q_I)}$ de termes de T telle que le terme C'_1 défini par :

- Pour toute position p de $\mathcal{P}os(C_1) \setminus \mathcal{SP}os(C_1)(q_I)$, $C'_1(p) = C_1(p)$ et,
- Pour toute position p de $\mathcal{SP}os(C_1)$, $C'_1|_p = lab_{u_p}$,

soit un terme contraint. Notons que C'_1 est sans occurrence de x. Puisque pour toute position p de $\mathcal{VP}os(C_1)(q_I)$, $lab_{u_p} \xrightarrow{*}_{\mathcal{A}} q_I$, nous avons $C'_1 \xrightarrow{*}_{\mathcal{A}} C_1 \xrightarrow{*}_{\mathcal{A}} q_1^{@}$. Donc le terme t_1 de $T(\Sigma)$ obtenu en supprimant les contraintes des nœuds de C'_1 appartient à $\mathcal{L}(\mathcal{A})$ puisque $t_1 \xrightarrow{*}_{\mathcal{A}} q_1(t_1)$.

De manière similaire, il existe une instance d'états C'_2 de C_2 sur T, c'est-à-dire qu'il existe une famille $(u_p)_{p \in \mathcal{SP}os(C_2)(q_I)}$ de termes de T telle que le terme C'_2 défini par :

- Pour toute position p de $\mathcal{P}os(C_2) \setminus \mathcal{SP}os(C_2)(q_I)$, $C'_2(p) = C_2(p)$ et,
- Pour toute position p de $\mathcal{SP}os(C_2)$, $C'_2|_p = lab_{u_p}$,

est un terme contraint et $C'_2 \xrightarrow{*}_{\mathcal{A}} s_1^{@}$. Donc le terme t_2 de $T(\Sigma)$ obtenu en supprimant les contraintes des nœuds de C'_2 n'appartient pas à $\mathcal{L}(\mathcal{A})$ puisque $t_2 \xrightarrow{*}_{\mathcal{A}} s_1(t_2)$. Notons que C'_2 est sans occurrence de x.

Soit C_g le terme de $T(\Sigma, \{x\})$ obtenu en supprimant les contraintes des nœuds de $\bar{C}[F'_1]$. Par construction de F'_2, C_g est également le terme obtenu en supprimant les contraintes des nœuds de $\bar{C}[F'_2]$.

Nous remplaçons toutes les occurrences de x dans C_g par de nouvelles variables distinctes. Il résulte de cette opération un contexte C'_g sur de nouvelles variables $(x_p)_{p \in \mathcal{VP}os(C_g)}$ défini par :

- Pour toute position p de $\mathcal{P}os(C_g) \setminus \mathcal{VP}os(C_g)$, $C'_g(p) = C_g(p)$ et,
- pour toute position p de $\mathcal{VP}os(C_g)$, $C'_g(p) = x_p$.

Nous pouvons montrer que $t_1 = C'_g[(u_p)]$ et $t_2 = C'_g[(u'_p)]$. De plus pour toute position p de $\mathcal{VP}os(C_g)$, nous avons $u_p \xrightarrow{*}_{\mathcal{B}} q_{\mathcal{B}}(u_p)$ et $u'_p \xrightarrow{*}_{\mathcal{B}} q_{\mathcal{B}}(u'_p)$. Donc la propriété (8) est contredite puisque t_1 appartient à $\mathcal{L}(\mathcal{A})$, t_2 n'appartient pas à $\mathcal{L}(\mathcal{A})$ et $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. Nous en déduisons que $\mathcal{L}(\mathcal{A})$ n'est pas régulier. □

Montrons maintenant que la condition du théorème 217 est suffisante.

Proposition 222. *Si pour toute paire de règles* $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ *et* $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c']} s(f(x_1, \ldots, x_n))$ *de* Δ, *nous avons* $q = s$. *Alors* $\mathcal{A}$ *est régulier et on peut construire un AAS le reconnaissant.*

Preuve : Supposons que pour toute paire de règles $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ et $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c']} s(f(x_1, \ldots, x_n))$ de Δ, nous avons $q = s$. Soit $\mathcal{B} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta')$ l'automate d'arbres standards dont l'ensemble de règles est défini par :

$$\forall f \in \Sigma_n, \forall (q_i)_{i \in [n]} \in \mathcal{Q},\ f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n)) \in \Delta$$

où q est déterminé par une règle $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{[c]} q(f(x_1, \ldots, x_n))$ de Δ (q existe car $\mathcal{A}$ est normalisé-complet et est unique d'après les hypothèses). Nous pouvons montrons facilement que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. Donc $\mathcal{L}(\mathcal{A})$ est régulier et est reconnu par l'*AAS* $\mathcal{B}$. □

Nous déduisons du théorème 217 que le problème de reconnaissabilité est décidable pour la classe *RATF* (théorème 194). De plus, d'après la proposition 222, lorsqu'un langage de la classe *RATF* est régulier, on peut construire un *AAS* le reconnaissant.

3.6 Applications

Dans cette section, nous considérons l'image par morphisme d'arbres puis par application d'un pas de réécriture d'un régulier.

Tout d'abord, soient Σ et Γ deux signatures, $\mathcal{L}$ un langage régulier de $T(\Sigma)$ et Φ un morphisme d'arbres de $T(\Sigma)$ dans $T(\Gamma)$. Nous nous intéressons à la question « $\Phi(\mathcal{L})$ régulier? ».

Dans le cas général, cette question est ouverte. Dans le cas particulier d'un morphisme linéaire, $\Phi(\mathcal{L})$ est régulier puisque la classe des réguliers est close par morphisme d'arbres linéaire (propriété 47). Nous nous intéressons maintenant au cas des morphismes semi-linéaires.

Proposition 223. *Si Φ est semi-linéaire, la question « $\Phi(\mathcal{L})$ régulier? » est décidable.*

Preuve : Supposons que Φ est semi-linéaire. Alors $\Phi(\mathcal{L})$ appartient à la classe *RATEF* puisque l'image d'un régulier par un morphisme semi-linéaire appartient à *RATEF* (propriété 137).

De plus le problème de reconnaissabilité étant décidable pour la classe *RATF* (théorème 194), il l'est aussi pour la classe *RATEF* puisque le pouvoir d'expression des *ATF* est supérieur à celui des *ATEF* (propriété 132). Donc la question « $\Phi(\mathcal{L})$ régulier? » est décidable.
□

Considérons maintenant un système de réécriture $\mathcal{R}$ sur Σ. Nous notons $\rightarrow$ la relation de réécriture définie par $\mathcal{R}$ et nous considérons l'ensemble

$$\mathcal{R}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists u \in \mathcal{L}, u \rightarrow t\}$$

correspondant à l'image de $\mathcal{L}$ par application d'un pas de $\mathcal{R}$. Nous nous intéressons à la question « $\mathcal{R}(\mathcal{L})$ régulier? ».

Pour un système quelconque, $\mathcal{R}(\mathcal{L})$ appartient à *RAR* donc la question « $\mathcal{R}(\mathcal{L})$ régulier? » est ouverte. Pour un système linéaire à droite, $\mathcal{R}(\mathcal{L})$ est régulier.

Dans le chapitre **Réécritures controlées** de la partie **Réécriture**, nous montrons que si $\mathcal{R}$ est semi-linéaire à droite, $\mathcal{R}(\mathcal{L})$ appartient à la classe *RATF* (preuve de la proposition 303). Donc pour un système semi-linéaire à droite, la question « $\mathcal{R}(\mathcal{L})$ régulier? » est décidable d'après la décidabilité du problème de reconnaissabilité pour la classe *RATF* (théorème 194).

3.7 Cas des *ATEC*

Théorème 224. *Le problème de reconnaissabilité est indécidable pour la classe RATEC.*

Pour montrer ce théorème, nous partons de la preuve d'indécidabilité du problème du vide pour les *ATEC* (chapitre 1). Nous considérons une machine de Turing $\mathcal{M}$ et $\mathcal{A}_{stop} =$

$\{\Gamma, \mathcal{Q}, \mathcal{Q}_f, \Delta\}$ l'*ATEC* reconnaissant l'ensemble des G-Arbres sur Γ codant un calcul réussi de la machine $\mathcal{M}$ débutant sur l'entrée vide (proposition 180).

Soient g et h des symboles de fonction respectivement unaire et binaire n'appartenant pas à Γ, q_t et q_f deux états n'appartenant pas à $\mathcal{Q}$ et $\mathcal{A} = \{\Gamma \cup \{g, h\}, \mathcal{Q} \cup \{q_t, q_f\}, \{q_f\}, \Delta'\}$ l'*ATEC* dont le système de réécriture Δ' contient Δ et les règles suivantes :

$$\begin{aligned} g(q(x)) &\rightarrow q_t(g(x)), \ \forall q \in \mathcal{Q}_f \\ g(q_t(x)) &\rightarrow q_t(g(x)) \\ h(q_t(g(x)), q_t(g(x))) &\rightarrow q_f(h(g(x), g(x))) \end{aligned}$$

Proposition 225. *La machine de Turing $\mathcal{M}$ s'arrête sur l'entrée vide si et seulement si $\mathcal{L}(\mathcal{A})$ n'est pas régulier.*

Preuve : Si la machine de Turing $\mathcal{M}$ s'arrête sur l'entrée vide, le langage $\mathcal{L}(\mathcal{A}, q_t)$ est infini. Nous pouvons alors montrer que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}, q_f)$ n'est pas régulier.

Par contre si $\mathcal{M}$ ne s'arrête pas sur l'entrée vide, $\mathcal{L}(\mathcal{A}, q_t)$ est vide. Alors $\mathcal{L}(\mathcal{A})$ est vide donc régulier. □

Puisque nous ne pouvons décider si la machine de Turing $\mathcal{M}$ s'arrête sur l'entrée vide, nous déduisons de la proposition précédente que le problème de reconnaissabilité est indécidable pour la classe *RATEC*.

Quatrième partie

Réécriture

Cette partie, composée de deux chapitres, est consacrée à l'étude de plusieurs domaines de la réécriture.

Dans le premier chapitre, nous étudions la théorie du premier ordre de la réécriture en un pas. Après avoir défini cette théorie, nous rappelons les liens existants entre celle-ci et l'unification contextuelle. Ensuite nous montrons en utilisant la méthode des Arbres-Grilles l'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas pour les systèmes de réécriture linéaires, minces, nœthériens et confluents. Puis nous montrons que la décidabilité du fragment existentiel de cette théorie munie de contraintes d'égalité et de contraintes d'appartenance à des langages reconnaissables par automates à tests entre frères, pour les systèmes de réécriture dont toutes les variables sont à profondeur un.

Dans le second chapitre, nous nous intéressons aux réécritures contrôlées. Dans un premier temps, nous définissons une méthode d'étude générale du langage $\mathcal{R}^*(\mathcal{L})$ où $\mathcal{R}$ est un système de réécriture et $\mathcal{L}$ un langage régulier. Ensuite, nous considérons les questions de décision « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » où $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des langages réguliers et $\mathcal{R}el(\mathcal{L}_1)$ est l'ensemble image des termes de $\mathcal{L}_1$ par une relation $\mathcal{R}el$. Ces questions sont décidables lorsque la relation est un morphisme d'arbres inverses ou un morphisme d'arbres linéaires. Nous étudions ces questions lorsque $\mathcal{R}el$ est l'application d'un pas de réécriture ou d'une réécriture parallèle, ou encore le calcul des formes normales ou des formes sententielles selon une passe IO, une passe OI, la stratégie de réécriture en une passe par la racine ou par les feuilles.

Chapitre 1

Théorie de la réécriture en un pas

Soient Σ une signature, $\mathcal{X}$ un ensemble de variables, $R_1, R_2, \ldots$ un ensemble dénombrable de systèmes de réécriture et un prédicat binaire $\rightarrow_{R_i}$ sur $T(\Sigma)$ pour tout entier i de $[n]$.

La théorie du premier ordre de la réécriture en un pas est l'ensemble des formules logiques du premier ordre construites sur la signature Σ, l'ensemble de variables $\mathcal{X}$ et l'ensemble des prédicats $\{\rightarrow_{R_i}\}$. La structure de cette théorie, notée $\mathcal{A}_{\Sigma,\cup_i R_i}$, est définie comme suit :

- Le support est l'ensemble des termes construits sur Σ c'est-à-dire $T(\Sigma)$,
- L'élément (respectivement la fonction) associé à toute constante e (respectivement tout symbole de fonction d'arité non nulle) est canonique, dans le sens où pour toute constante e de Σ (respectivement pour tout symbole de fonction f d'arité n non nulle), $e^{\mathcal{A}_{\Sigma,\cup_i R_i}} = e$ (respectivement $f^{\mathcal{A}_{\Sigma,\cup_i R_i}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ pour tous termes $t_1, \ldots, t_n$ de $T(\Sigma)$),
- Pour tout i, la relation $\rightarrow_{R_i}^{\mathcal{A}_{\Sigma,\cup_i R_i}}$ associée au symbole de prédicat $\rightarrow_{R_i}$ est la relation de réécriture associée à R_i sur $T(\Sigma)$.

Donc si s et t sont deux termes de $T(\Sigma)$, nous avons $\mathcal{A}_{\Sigma,\cup_i R_i} \models s \rightarrow_{R_i} t$ si et seulement si s se réécrit en t par l'application d'une règle du système R_i. Dans la suite, lorsqu'il n'y a qu'un seul système de réécriture R, nous omettons l'indice R dans les formules.

Le chapitre est organisé de la façon suivante. Tout d'abord, nous rappelons les différentes connexions existant entre la réécriture en un pas et l'unification contextuelle (section 1.1). Puis nous nous intéressons à la décidabilité de certaines classes de formules de la théorie du premier ordre de la réécriture en un pas (sections 1.2 et 1.3).

1.1 Réécriture en un pas et unification contextuelle

Nous définissons tout d'abord les notions d'unification sur les mots, d'unification contextuelle et d'unification linéaire du second ordre. Puis nous donnons quelques résultats concernant l'unification contextuelle. Finalement nous rappelons les différentes connexions existant entre la réécriture en un pas et l'unification contextuelle.

Soient une signature Σ, un ensemble dénombrable de variables du premier ordre $\mathcal{X}$ et un ensemble dénombrable de variables d'arité un dites variables de *contextes* ou encore du *second*

ordre. Nous désignons par C les variables de contexte. Un *terme du second ordre* est :

- Soit une variable du premier ordre x,
- Soit un terme $f(t_1, \ldots, t_n)$ où f est un symbole de Σ_n et $t_1, \ldots, t_n$ sont des termes du second ordre,
- Soit l'application d'une variable de contexte C sur un terme du second ordre t. Une telle application est notée $C(t)$.

En particulier, tout terme de $T(\Sigma, \mathcal{X})$ est un terme du second ordre. Les termes de $T(\Sigma, \mathcal{X})$ sont aussi nommés *termes du premier ordre.*

Une *contrainte de contexte* φ est une conjonction d'équations $t = t'$ entre termes du second ordre :

$$\varphi := t = t' \mid \varphi \wedge \varphi'.$$

Pour ce qui est de l'interprétation, les variables de contexte sont interprétées comme des fonctions de contexte, définies ci-dessous, et les termes du second ordre comme des termes. Une *fonction de contexte* γ est une fonction de $T(\Sigma, \mathcal{X})$ dans $T(\Sigma, \mathcal{X})$ décrite par une équation de la forme

$$\gamma(t) = s[x \leftarrow t] \qquad \text{pour tout arbre } t,$$

où s est un *contexte de trou* x, c'est-à-dire s est un contexte de $\mathcal{C}(\Sigma)$ possédant une seule occurrence de la variable x et aucune occurrence d'autre variable. Si γ a la représentation donnée ci-dessus, le *chemin d'exception* de γ est le chemin menant de la racine de s au trou x.

Une substitution β associe à chaque variable du premier ordre un arbre et à chaque variable de contexte une fonction de contexte. Alors l'interprétation $\beta(t)$ d'un terme du second ordre t par β est définie inductivement par :

$$\begin{aligned} \beta(f(t_1, \ldots, t_n)) &= f(\beta(t_1), \ldots, \beta(t_n)) \\ \beta(C(t)) &= \beta(C)(\beta(t)) \end{aligned}$$

Une substitution β satisfait une équation $t = t'$ si et seulement si $\beta(t) = \beta(t')$. Une *solution* d'une contrainte de contexte est une substitution qui satisfait toutes les équations de la contrainte. Une contrainte de contexte est dite *satisfiable* si elle admet une solution. Finalement l'*unification contextuelle* est le problème de satisfiabilité des contraintes de contexte.

Nous définissons maintenant les notions d'unification sur les mots et d'unification linéaire du second ordre.

Unification sur les mots. L'unification sur les mots est le problème de résolution d'équations de mots, une équation de mot étant une équation de la forme $u = v$ où u et v sont des mots construits sur un alphabet et un ensemble de variables d'arité zéro. Les variables sont interprétées par des mots sur l'alphabet. L'unification contextuelle est une généralisation de l'unification sur les mots puisque les lettres de l'alphabet peuvent être vus comme des symboles de fonction d'arité un et les variables comme des variables de contexte.

L'unification sur les mots a été découverte et étudiée par plusieurs communautés indépendantes de chercheurs (voir [7]). La notion d'unification sur les mots provient du domaine de la déduction automatique [71, 82] où elle est aussi appelée A-unification [6] avec un unique

symbole de fonction associatif. Elle a été présentée pour la première fois par A. Markov [63] et est appelée problème de Markov par les mathématiciens des pays de l'Est. Elle est appelée problème de Löb par les mathématiciens de l'Ouest, par exemple par A. Lentin et M.-P. Schützenberger [56, 55]. La décidabilité de l'unification sur les mots est restée pendant plus de vingt ans un problème ouvert. Mais en 1977, elle a été montrée décidable par G. Makanin [59]. Par la suite, plusieurs chercheurs ont cherché une meilleure description de l'algorithme de Makanin et à combler certains petits trous de sa preuve de correction [68, 69, 48, 79, 80]. La complexité de l'algorithme de Makanin est étudiée dans [52].

Unification linéaire du second ordre. Dans l'unification linéaire du second ordre, les variables de contexte peuvent être d'arité quelconque et sont interprétées par des contextes à plusieurs trous, c'est-à-dire par un terme linéaire dans lequel plusieurs variables apparaissent. L'unification contextuelle est donc un sous-problème de l'unification linéaire du second ordre.

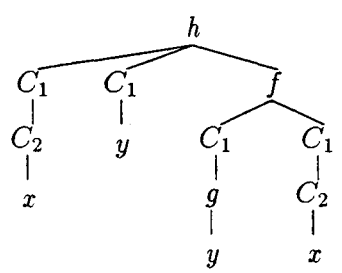
La décidabilité de l'unification linéaire du second ordre est un problème ouvert mais trois de ses fragments ont été montrés décidables [57]. Notons aussi que l'unification du second ordre (sans la restriction de linéarité) est indécidable [38].

Nous revenons maintenant à l'étude de l'unification contextuelle. La décidabilité de l'unification contextuelle est un problème ouvert mais comme pour l'unification linéaire du second ordre trois de ses fragments ont été montrés décidables [19, 77, 57].

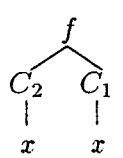
M. Schmidt-Schauss [77] définit les notions de préfixe du second ordre et de contrainte de contexte stratifiée. Le *préfixe du second ordre* d'une occurrence de variable (du premier ordre ou de contexte) dans un terme s est la suite de variables de contexte utilisées dans s sur le chemin menant de la racine de s à l'occurrence de la variable. Une contrainte de contexte φ est *stratifiée* si pour toute variable du premier ordre et de contexte, les préfixes du second ordre de toutes ses occurrences dans les termes apparaissant dans φ sont les mêmes (exemple 226). Schmidt-Schauss montre que l'unification contextuelle restreinte aux contraintes de contexte stratifiées est décidable.

Exemple 226. *Soient la signature* $\Sigma = \{h/3, f/2, g/1, a/0\}$, *deux variables du premier ordre* x *et* y, *et deux variables de contexte* C_1 *et* C_2. *Alors le terme* t *ci-dessous est stratifié, mais pas* s *puisque la variable* x *a pour préfixes du second ordre* C_1 *et* C_2.

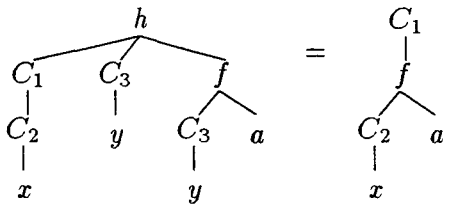
$t =$



$s =$

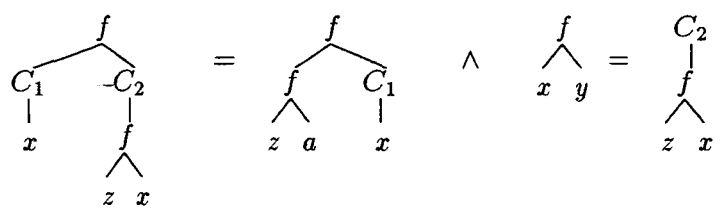


Et la contrainte de contexte suivante est stratifiée :

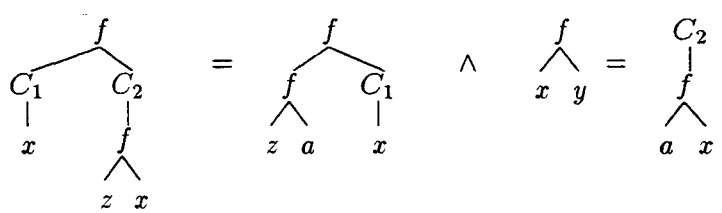


H. Comon [19] montre que le fragment existentiel de la théorie du premier ordre des contraintes de contexte uniformes est décidable, une contrainte de contexte φ étant *uniforme* si pour toute variable de contexte C, toutes les occurrences de C dans φ ont même sous-termes (exemple 227). J. Lévy [57] donne une preuve différente de ce résultat.

Exemple 227. *Soient la signature* $\Sigma = \{f/2, g/1, a/0\}$, *trois variables du premier ordre* x,y *et* z, *et deux variables de contexte* C_1 *et* C_2. *Alors la contrainte de contexte ci-dessous est uniforme.*



Par contre, la contrainte de contexte suivante n'est pas uniforme puisque toutes les occurrences de C_2 *n'ont pas même fils.*



Finalement l'unification contextuelle avec au plus deux occurrences de chaque variable (du premier ordre et de contexte) est décidable. Ce résultat a été montré pour la première fois par J. Lévy [57] dans le cadre de l'unification linéaire du second ordre. Une preuve adaptée à l'unification contextuelle est donnée par J. Niehren, M. Pinkal et P. Ruhrberg [45].

Nous rappelons maintenant les différentes connexions existant entre la réécriture en un pas et l'unification contextuelle.

J. Niehren, M. Pinkal et P. Ruhrberg [45] montrent que le fragment existentiel positif (c'est-à-dire sans négation $\neg$) de la théorie du premier ordre de la réécriture en un pas est

décidable. En fait si s et t sont deux termes et $r_e := l \to r$ une règle d'un système de réécriture, alors du point de vue de la satisfiabilité

$$s \xrightarrow{r_e} t \Leftrightarrow \exists C\ (s = C(l') \wedge t = C(r'))$$

où C est une variable de contexte, et l' et r' sont des termes obtenus à partir de l et r par un renommage de variables de sorte que l' et r' ne partagent aucune variable avec s et t. De plus la condition sur les variables fait que la contrainte de contexte ci-dessus est stratifiée. Donc toute conjonction de *contraintes de réécriture* ($s \xrightarrow{r_e} t$) est équivalente du point de vue de la satisfiabilité à une contrainte de contexte stratifiée existentiellement quantifiée. J. Niehren, M. Pinkal et P. Ruhrberg déduisent alors leur résultat de la décidabilité de l'unification contextuelle restreinte aux contraintes de contexte stratifiées (M. Schmidt-Schauss [77]).

J. Niehren, M. Pinkal et P. Ruhrberg déduisent également de la relation précédente et de l'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas (R. Trenein [90]) que le fragment $\exists^*\forall^*\exists^*$ de la théorie du premier ordre de l'unification contextuelle stratifiée est indécidable.

Il est facile de définir une légère généralisation des contraintes de réécriture et d'établir une relation étroite entre cette généralisation et l'unification contextuelle (J. Niehren et R. Treinen [67]). En fait, on considère des contraintes de réécriture indiquant quel est le terme du second ordre se trouvant au-dessus de la position où s'applique la réécriture. Formellement les contraintes de réécriture sont de la forme $x \xrightarrow{r_e} y$ en Δ où x, y sont des variables du premier ordre, $r_e := l \to r$ une règle de réécriture et Δ une *chaîne de variables de contexte* (c'est-à-dire soit ϵ, soit une variable de contexte, soit la concaténation de variables de contexte). L'interprétation de ϵ est la fonction identité sur l'ensemble des termes clos et la relation $s \xrightarrow{r_e} t$ en Δ est satisfiable s'il existe une substitution σ telle que $s = \sigma(\Delta(l))$ et $t = \sigma(\Delta(r))$.

On considère également un symbole de relation d'ordre $\leq$ sur les contextes.

J. Niehren et R. Treinen montrent premièrement qu'il existe des translations constructives préservant la satisfiabilité entre la classe des contraintes de contexte et la classe des contraintes de réécriture généralisées, une *contrainte de réécriture généralisée* étant une conjonction de contraintes $x \xrightarrow{r_e} y$ en Δ. Deuxièmement ils montrent qu'il existe également de telles translations entre la classe des contraintes de contexte stratifiées et la classe constituée des conjonctions de contraintes $x \xrightarrow{r_e} y$ en D, $C \leq D$ et $C \leq \epsilon$ où C et D sont des variables de contexte.

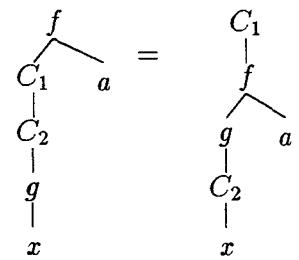
Tout d'abord, la translation d'une contrainte de réécriture généralisée en une contrainte de contexte se fait par la règle (1).

$$\frac{x \xrightarrow{l \to r} y \text{ en } \Delta}{x = \Delta(l') \wedge y = \Delta(r')} \quad (1)$$

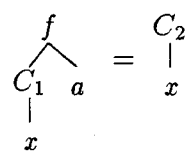
▶ où l' et r' sont obtenus à partir de l et r par un renommage de variables tel que les variables de l' et r' sont libres.

Examinons maintenant la transformation inverse. Nous illustrons cette transformation sur deux exemples, l'un concernant une contrainte de contexte stratifiée (exemple 228), l'autre une contrainte non stratifiée (exemple 229).

Exemple 228. *Soient la signature* $\Sigma = \{f/2, g/1, a/0\}$ *et la contrainte stratifiée :*



Exemple 229. *Soient la signature* $\Sigma = \{f/2, a/0\}$ *et la contrainte non stratifiée :*



Considérons un problème d'unification contextuelle $t_1 = t_2 \wedge \cdots \wedge t_{2n-1} = t_{2n}$. Si $x_1, \ldots, x_n$ sont des nouvelles variables du premier ordre, le problème se transforme en le problème $x_1 = t_1 \wedge x_1 = t_2 \wedge \cdots \wedge x_n = t_{2n-1} \wedge x_n = t_{2n}$.

On transforme tout d'abord ce dernier système en remplaçant toute équation $x = t$, avec x variable et t terme du second ordre, par une contrainte de réécriture en la première chaîne de variables de contexte rencontrée en parcourant t de sa racine vers ses feuilles. Cette transformation est obtenue par saturation du problème par les règles (2) et (3) (exemples 230 et 231).

$$\frac{x = \Delta(t(u_1, \ldots, u_n))}{\bigwedge_{i \in [n]} \left(z_i = \Delta(u_i) \wedge x \xrightarrow{t(y_1, \ldots, y_n) \to y_i} z_i \text{ en } \Delta \right)} \quad (2)$$

▶ pour un entier n strictement positif, une chaîne de contextes Δ, n termes $u_1, \ldots, u_n$ dont les symboles de tête sont des variables du premier ordre ou des variables de contexte, un terme du premier ordre $t(y_1, \ldots, y_n)$ et n variables du premier ordre libres $z_1, \ldots, z_n$.

$$\frac{x = \Delta t}{x \xrightarrow{t \to t} x \text{ en } \Delta} \quad (3)$$

▶ pour t clos.

On obtient une formule contenant uniquement des contraintes de réécriture et des équations de la forme $z = \Delta y$ avec Δ chaîne de variables de contexte.

Exemple 230. *Suite de l'exemple 228.*

La contrainte se transforme tout d'abord en le problème suivant :

$$x_1 = f(C_1(C_2(g(x))), a) \ \wedge \ x_2 = C_1(f(g(C_2(x)), a)).$$

Par une première application de la règle (2), nous obtenons le problème :

$$z_1 = C_1(C_2(g(x))) \ \wedge \ x_1 \xrightarrow{f(y_1,a)\to y_1} z_1 \ en \ \epsilon \ \wedge \ z_2 = C_1(C_2(x)) \ \wedge \ x_2 \xrightarrow{f(g(y_2),a)\to y_2} z_2 \ en \ C_1.$$

Par une nouvelle application de la règle (2) sur la contrainte $z_1 = C_1(C_2(g(x)))$, nous obtenons le problème :

$$z_3 = C_1(C_2(x)) \ \wedge \ z_1 \xrightarrow{g(y_3)\to y_3} z_3 \ en \ C_1C_2$$
$$\wedge \ x_1 \xrightarrow{f(y_1,a)\to y_1} z_1 \ en \ \epsilon \ \wedge \ z_2 = C_1(C_2(x)) \ \wedge \ x_2 \xrightarrow{f(g(y_2),a)\to y_2} z_2 \ en \ C_1.$$

Exemple 231. *Suite de l'exemple 229.*

La contrainte se transforme tout d'abord en le problème suivant :

$$x_1 = f(C_1(x), a) \ \wedge \ x_2 = C_2(x).$$

Par application de la règle (2), nous obtenons le problème :

$$z_1 = C_1(x) \ \wedge \ x_1 \xrightarrow{f(y_1,a)\to y_1} z_1 \ en \ \epsilon \ \wedge \ x_2 = C_2(x).$$

Ensuite, on supprime les équations de la forme $x = \Delta y$. Dans le cas où le problème de départ est stratifié alors la formule obtenue par saturation par les règles (2) et (3) est stratifiée puisque ces règles conservent la stratification. On sature alors le problème par la règle (4) puisque toutes les occurrences d'une variable y ont même préfixe du second ordre.

En fait, on exprime le fait que $z_1 = z_2 = \cdots = z_m$ et que Δ est un contexte de z_1 (exemple 232).

$$\frac{z_1 = \Delta y \wedge \cdots \wedge z_m = \Delta y}{\bigwedge_{i,j\in[m]} z_i \xrightarrow{v\to v} z_j \text{ en } \Delta} \quad (4)$$

▶ où v est une variable du premier ordre.

Exemple 232. *Suite de l'exemple 230.*

Par application de la règle (4), la formule $z_3 = C_1(C_2(x)) \wedge z_2 = C_1(C_2(x))$ *se transforme en :*

$$\bigwedge_{i,j\in\{2,3\}} z_i \xrightarrow{v\to v} z_j \text{ en } C_1C_2.$$

Dans le cas où le problème de départ n'est pas stratifié, on applique la règle (5). En fait, on exprime les différentes valeurs que peut prendre y. Ensuite les équations introduites par cette règle sont éliminées par la règle (3) (exemple 233).

$$\frac{z_1 = \Delta_1 y \wedge \cdots \wedge z_m = \Delta_m y}{\bigvee_{a \text{ constante}} \bigwedge_{i\in[m]} z_i = \Delta_i(E(a))} \quad (5)$$

▶ où E est une variable de contexte libre.

Exemple 233. *Suite de l'exemple 231.*

Par application de la règle (5), la formule $z_1 = C_1(x) \wedge x_2 = C_2(x)$ *se transforme en :*

$$z_1 = C_1(E(a)) \ \wedge \ x_2 = C_2(E(a))$$

puisque a *est la seule constante de la signature* Σ*. Ensuite par application de la règle (3), nous obtenons :*

$$z_1 \xrightarrow{a\to a} z_1 \text{ en } C_1E \ \wedge \ x_2 \xrightarrow{a\to a} x_2 \text{ en } C_2E.$$

Finalement la contrainte de contexte de départ est équivalente du point de vue de la satisfiabilité au système de contraintes de réécriture généralisées suivant :

$$x_1 \xrightarrow{f(y_1,a)\to y_1} z_1 \text{ en } \epsilon \ \wedge \ z_1 \xrightarrow{a\to a} z_1 \text{ en } C_1E \ \wedge \ x_2 \xrightarrow{a\to a} x_2 \text{ en } C_2E.$$

La dernière opération à réaliser dans le cas d'un problème stratifié est de transformer les relations de réécriture $x \xrightarrow{r_\epsilon} y$ en Δ où Δ n'est pas une variable de contexte. Pour cela, on remplace toute chaîne de variables de contexte par une nouvelle variable de contexte (règle (6)).

Finalement les définitions des nouvelles variables de contexte sont remplacées par des relations d'ordre sur les contraintes (règles (7) et (8)) (exemple 234).

$$\frac{x \xrightarrow{r_\epsilon} y \text{ en } \Delta}{x \xrightarrow{r_\epsilon} y \text{ en } C_\Delta \wedge C_\Delta = \Delta} \quad (6)$$

▶ où C_Δ est une variable de contexte libre.

$$\frac{C_{\Delta D} = \Delta D}{\Delta \leq C_{\Delta D}} \quad (7)$$

▶ où Δ est une chaîne de variables de contexte différente de ϵ.

$$\frac{C_\epsilon = \epsilon}{C_\epsilon \leq \epsilon} \quad (8)$$

Exemple 234. *Suite de l'exemple 232.*

En appliquant les règles (6), (7) et (8), nous obtenons que la contrainte de contexte stratifiée de départ est équivalente du point de vue de la satisfiabilité au système de contraintes de réécriture généralisées suivant :

$$x_1 \xrightarrow{f(y_1,a)\to y_1} z_1 \text{ en } C_\epsilon \wedge z_1 \xrightarrow{g(y_3)\to y_3} z_3 \text{ en } C_{C_1C_2} \wedge x_2 \xrightarrow{f(g(y_2),a)\to y_2} z_2 \text{ en } C_1$$
$$\wedge \left(\bigwedge_{i,j\in\{2,3\}} z_i \xrightarrow{v\to v} z_j \text{ en } C_{C_1C_2} \right) \wedge C_1 \leq C_{C_1C_2} \wedge C_\epsilon \leq \epsilon.$$

1.2 Indécidabilité

D'importantes propriétés décidables, telles que la confluence forte et la réductibilité inductive, s'expriment dans la théorie du premier ordre de la réécriture en un pas. Cette théorie semblait donc décidable mais cette conjecture s'avéra fausse lorsque R. Treinen [89] montra l'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas pour tout système de réécriture. Par la suite, plusieurs chercheurs s'intéressèrent à ce problème. R. Treinen [88] affina son résultat au fragment $\exists^*\forall^*$ pour les systèmes de réécriture linéaires par codage du problème de correspondance de Post, J. Marcinkowski [61] au fragment $\exists^*\forall^*$ pour les systèmes nœthériens clos à droite par codage des arbres Thue réguliers (outils définis dans [62] et [60]), et S. Vorobyov [94] au fragment $\exists^*\forall^*\exists^*$ pour les systèmes linéaires et nœthériens par codage du problème de correspondance de Post. Dans ce chapitre, nous affinons encore ce résultat.

Théorème 235. *Le fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas pour les systèmes de réécriture linéaires, minces et convergents est indécidable.*

Nous montrons ce résultat en utilisant la méthode des Arbres-Grilles que nous avons introduite dans la partie **Outils de décidabilité et d'indécidabilité.** Nous rappelons tout d'abord les différents éléments intervenant dans cette méthode.

Soit $\mathcal{M} = (Q, \{a, b\}, \{a, b, \square\}, \delta, q_s, \square, q_f)$, avec $Q = \{q_1, \ldots, q_k\}$, une instance de la classe restreinte des machines de Turing déterministes (définition 164). Nous définissons une signature Γ composée des symboles de fonction suivants :

$$\begin{aligned} f &: \textit{ternaire} \\ \bot_0 &: \textit{constante} \\ a_l, b_l, a_r, b_r, \square_r, q_1, \ldots, q_k &: \textit{constante} \end{aligned}$$

sur laquelle on définit un ensemble d'arbres particuliers, appelés G-Arbres (définition 175), représentant les grilles finies.

Définition 175 (G-Arbre). *Un terme clos t d'une signature contenant Γ est un* G-Arbre sur Γ si :

$$t \in T(\Gamma) \qquad \text{(PURETÉ)}$$

Pour tout sous-arbre $f(x, y, z)$ de t :

$$x = \bot_0 \text{ ou } tête(x) = f \qquad \text{(FILS-1)}$$
$$y = \bot_0 \text{ ou } tête(y) = f \qquad \text{(FILS-2)}$$
$$z \in \{a_l, b_l, a_r, b_r, \square_r, q_1, \ldots, q_k\} \qquad \text{(CONSTANTES)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), f(x_2, y_2, z_2), z_3)$ de t :

$$y_1 = x_2 \qquad \text{(EGALITÉ)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), y_2, z_2)$ de t :

$$tête(y_1) = f \qquad \text{(CORPS-1)}$$

Pour tout sous-arbre $f(x_1, f(f(x_2, y_2, z_2), y_3, z_3), z_1)$ de t :

$$tête(x_1) = f \qquad \text{(CORPS-2)}$$

Nous définissons ensuite un ensemble P'_T de contextes de $T(\Gamma, \mathcal{X})$ décelant les arbres qui ne codent pas un calcul de la machine $\mathcal{M}$. Finalement nous montrons que l'on peut caractériser l'ensemble des G-Arbres codant les calculs réussis de la machine $\mathcal{M}$ débutant sur l'entrée vide (proposition 177).

Proposition 177. *Un G-Arbre t sur Γ code un calcul réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide si et seulement si :*

- *t entoure l'arbre $f(x_1, f(x_2, f(x_3, \bot_0, \square_r), \square_r), q_s)$* (INIT1)
- *t n'entoure pas l'arbre $f(f(x_1, x_2, q_s), x_3, x_4)$* (INIT2)
- *t n'entoure pas l'arbre $f(x_1, f(x_2, x_3, q_s), x_4)$* (INIT3)
- *t n'entoure aucun des motifs de P'_T* (CALCUL)
- *t entoure l'arbre $f(\bot_0, x_1, q_f)$* (FINAL)

Ces rappels faits nous donnons l'idée de la preuve du théorème 235. Nous réduisons le problème de l'arrêt de la machine $\mathcal{M}$ en la satisfiabilité d'une formule close, nommée *stop*, dans une certaine structure $\mathcal{A}_{\Sigma_P, R_P}$ (définition 237) en utilisant la méthode des Arbres-Grilles :

$$stop := \exists x \left(grille(x) \wedge init(x) \wedge calcul(x) \wedge final(x) \right)$$

Les formules $grille(x)$, $init(x)$, $calcul(x)$ et $final(x)$ permettent de vérifier si un arbre satisfait les conditions de la proposition 177 :

- $grille(x)$ teste si un arbre est un G-Arbre sur Γ (définition 175);
- $init(x)$ teste si un arbre satisfait les conditions relatives à la configuration initiale (conditions INIT1, INIT2 et INIT3 de la proposition 177);
- $calcul(x)$ teste si un arbre n'entoure aucun des contextes de P'_T (condition CALCUL de la proposition 177);
- $final(x)$ teste si un arbre satisfait la condition sur l'état final (condition FINAL de la proposition 177).

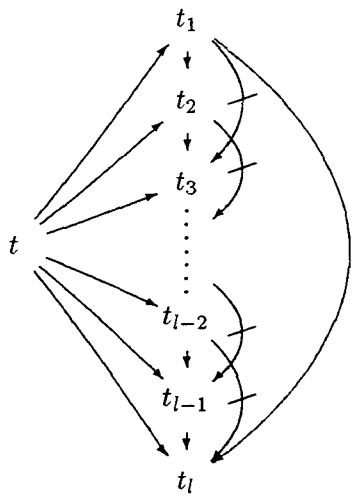
Ces formules dépendent d'une construction excluant et imposant la présence de motifs (contextes). Soit $\{\pi_1, \ldots, \pi_n\}$ un ensemble de motifs. La partie technique de la preuve réside en la définition d'une formule $match[\pi_i](x)$ (définition 247) qui teste si un arbre x entoure le motif π_i. Ce test s'effectue en utilisant une suite de réécritures appelée triangle.

Un *triangle* de longueur l est composé de $l+1$ arbres, $t, t_1, \ldots, t_l$, de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 1. Avec le système de réécriture que nous considérons, les négations $t_j \not\to t_{j+2}$, pour tout i de $[l-2]$, apparaissant dans la définition d'un triangle entrainent que la chaîne de réécriture de t_1 à t_l ne peut être réduite.

L'astuce utilisée dans la construction de la formule $match[\pi_i](x)$ est d'associer à un motif π_i un triangle de longueur $i+3$ tel que :

> *Si un triangle de longueur $i+3$ peut être construit à partir de x en utilisant le système de réécriture R_P alors x entoure π_i.*

FIG. 1 - Un triangle de longueur l.



La structure du triangle est également utilisée pour tester si un arbre est un G-Arbre sur Γ. En effet, la première condition qu'un arbre doit vérifier pour cela est d'appartenir à $T(\Gamma)$. Nous appelons *pureté* la propriété pour un arbre d'appartenir à $T(\Gamma)$ (définition 236) et nous utilisons un triangle de longueur 3 pour tester cette propriété (définition 245 et proposition 246).

Définition 236 (Pureté). *Soit t un arbre. Si t appartient à $T(\Gamma)$, il est dit* pur *sinon il est dit* impur.

L'idée de la preuve étant donnée, nous allons tout d'abord définir les signatures Σ, Σ_P et le système de réécriture R_P que nous allons utiliser pour construire la formule *stop*. Le système de réécriture R_P est défini par des ensembles de règles de réécriture numérotés par un entier. Pour simplifier les preuves, nous utilisons ces numéros pour désigner une règle de chacun de ces ensembles de règles. Par exemple, la notation "règle (6)" signifie "une règle de l'ensemble (6)".

Définition 237 (Signatures Σ, Σ_P, système de réécriture R_P). *La signature Σ est l'extension de la signature Γ par ajout d'un symbole unaire w et de deux symboles ternaires u et v. Soit $P = \{\pi_1, \ldots, \pi_n\}$ un ensemble de contextes (appelés motifs) de $T(\Sigma, \mathcal{X}) \setminus (\Gamma_0 \cup \mathcal{X})$. La signature Σ_P est définie comme suit :*

$$\begin{aligned}\Sigma_P \;:=\;& \Sigma \\ & \cup \{c_{\pi,o}/0 \mid \pi \in P, o \in \mathcal{IP}os(\pi)\} \\ & \cup \{e_{i,j}/0 \mid 1 \leq i \leq n, 1 \leq j \leq i+3\} \\ & \cup \{\tilde{f}/3, \tilde{u}/3, \tilde{v}/3, \tilde{w}/1\}\end{aligned}$$

Pour tout π de P et o de $\mathcal{P}os(\pi)$, soit $d_{\pi,o} := \pi|_o$ si $o \in \mathcal{LP}os(\pi)$, et $d_{\pi,o} := c_{\pi,o}$ si $o \in \mathcal{IP}os(\pi)$.

Le système de réécriture R_P est constitué des ensembles de règles suivantes :

- *Règles (1)-(3) pour vérifier qu'un arbre satisfait les égalités caractéristiques des G-Arbres (condition* EGALITÉ *de la définition 175) :*

$$\{f(x,y,z) \to u(x,y,z), f(x,y,z) \to r(x,y,z)\} \quad (1)$$
$$\{u(x,y,z) \to w(x)\} \quad (2)$$
$$\{r(x,y,z) \to w(y)\} \quad (3)$$

- *Règles (4)-(6) pour construire un triangle de longueur 3 afin de vérifier la (non) pureté d'un arbre :*

$$\{a_r \to b_r, b_r \to \Box_r, a_r \to \Box_r\} \quad (4)$$
$$\{\alpha \to a_r, \alpha \to b_r, \alpha \to \Box_r \mid \alpha \in \Sigma_P \backslash \Gamma, Arité(\alpha) = 0\} \quad (5)$$
$$\{\beta(x_1,\ldots,x_p) \to a_r, \beta(x_1,\ldots,x_p) \to b_r, \beta(x_1,\ldots,x_p) \to \Box_r \mid \beta \in \Sigma_P \backslash \Gamma, Arité(\beta) = p, p > 0\} \quad (6)$$

- *Règles (7)-(8) pour réduire un motif (figure 2(a)) en une constante spéciale :*

$$\{h(x_1,\ldots,x_p) \to \tilde{h}(x_1,\ldots,x_p) \mid h = f, r, u \text{ ou } w\} \quad (7)$$
$$\{\tilde{h}(d_{\pi,o1},\ldots,d_{\pi,op}) \to c_{\pi,o} \mid \pi \in P, o \in \mathcal{IP}os(\pi), tête(\pi|_o) = h, Arité(h) = p\} \quad (8)$$

- *Règles (9)-(11) pour construire un triangle (figure 2(b)) afin d'identifier un motif :*

$$\{c_{\pi_i,\epsilon} \to e_{i,j} \mid 1 \le i \le n, 1 \le j \le i+3\} \quad (9)$$
$$\{e_{i,j} \to e_{i,j+1} \mid 1 \le i \le n, 1 \le j \le i+2\} \quad (10)$$
$$\{e_{i,1} \to e_{i,i+3} \mid 1 \le i \le n\} \quad (11)$$

Avant d'aller plus loin, faisons les trois remarques suivantes.

Remarque 238. *Par définition de P, l'ensemble $P \cap (\Gamma_0 \cup \mathcal{X})$ est vide. Donc tous les motifs sont de hauteur au moins un.*

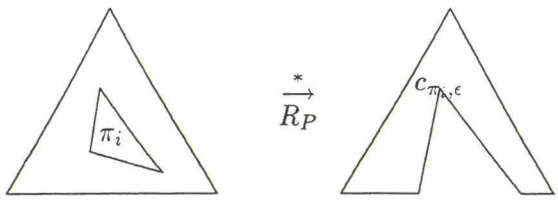
Remarque 239. *Le système de réécriture R_P est constitué uniquement de règles effaçantes (règles (2), (3), (6) et (8)) et de réétiquetages (règles (1), (4), (5), (7), (9), (10) et (11)). La figure 3 représente les réétiquetages du système.*

Les règles effaçantes du système de réécriture ne s'appliquent pas aux arbres purs, les seules règles du système s'appliquant aux arbres purs sont les règles (1), (4) et (7).

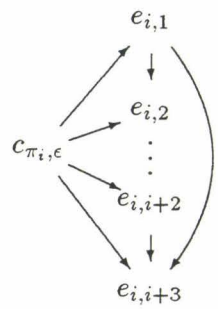
Nous montrons maintenant que le système R_P fait bien partie de la classe de systèmes citée dans le théorème 235.

Proposition 240. *Le systeme de réécriture R_P est linéaire, mince et convergent.*

FIG. **2** - Comment utiliser R_P pour vérifier si un arbre entoure un motif.

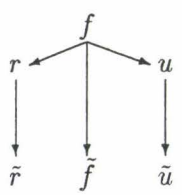


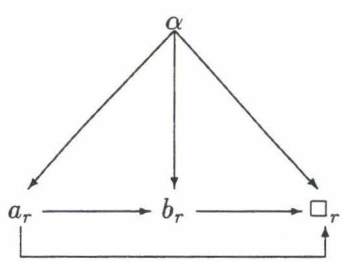
(a) Réduction d'un motif par les règles (7) et (8) de R_P. En partie gauche, l'arbre entoure π_i; en partie droite le motif est remplacé par la constante spéciale $c_{\pi_i,\epsilon}$.

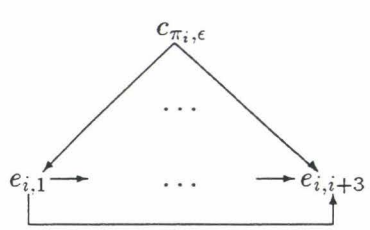


(b) Identification d'un motif π_i par utilisation d'un triangle de longueur $i+3$.

FIG. **3** - Schémas des réétiquetages.







Preuve : Le systeme de réécriture R_P est de façon évidente linéaire et mince. Pour la convergence de R_P, nous montrons qu'il est nœthérien puis confluent.

Nous prouvons que R_P est nœthérien par utilisation de l'ordre lexicographique de chemins (définition 18) sur $T(\Sigma_P)$ induit par l'ordre strict $\succ$ sur Σ_P suivant :

$$\begin{array}{rl} \forall \alpha \in \Sigma_P \backslash \Gamma, & \alpha \succ a_r \succ b_r \succ \Box_r \\ & f \succ u \succ r \succ w \\ \forall h \in \{f,u,r,w\}, \forall \pi \in P, \forall o \in \mathcal{IP}os(\pi), & h \succ \tilde{h} \succ c_{\pi,o} \\ \forall i \in [n], & c_{\pi_i,\epsilon} \succ e_{i,1} \succ e_{i,2} \succ \cdots \succ e_{i,i+3} \end{array}$$

D'après le théorème 19, le système R_P est nœthérien si $l \succ_{lpo} r$ pour toute règle $l \to r$ de R_P. Considérons la règle $f(x,y,z) \to u(x,y,z)$ (règle (1)). Par la condition (LPO1) de la définition des ordres lexicographiques de chemins (définition 18), nous avons $f(x,y,z) \succ_{lpo} x$, $f(x,y,z) \succ_{lpo} y$ et $f(x,y,z) \succ_{lpo} z$. Nous en déduisons que $f(x,y,z) \succ_{lpo} u(x,y,z)$ par la condition (LPO2b) de la définition (18) puisque $f \succ u$. Nous procédons de même pour les règles (2), (3) et (7).

Considérons maintenant la règle $a_r \to b_r$ (règle (4)). Par la condition (LPO2b) de la définition 18, nous avons $a_r \succ_{lpo} b_r$ puisque $a_r \succ b_r$ et b_r n'a pas de fils. Nous procédons de même pour les règles (5), (6), (8), (9), (10) et (11). Donc nous avons $l \succ_{lpo} r$ pour toute règle $l \to r$ de R_P. Nous en déduisons que R_P est nœthérien d'après le théorème 19.

Nous montrons maintenant que toute paire critique de R_P est joignable. Soient s et t deux termes de $T(\Sigma_P)$ tels que $\langle s,t \rangle$ soit une paire critique de R_P. Il existe :

- $l_1 \to r_1$ et $l_2 \to r_2$ deux règles de R_P telles que l_2 ne soit pas une variable et dont les variables ont été renommées de sorte que $\mathcal{V}ar(l_1) \cap \mathcal{V}ar(l_2) = \emptyset$,
- p une position de l_1 telle que $l_1|_p$ ne soit pas une variable et telle que $l_1|_p$ et l_2 s'unifient, et
- θ l'un des unificateurs les plus généraux de $l_1|_p$ et l_2,

tels que $s = r_1\theta$ et $t = (l_1\theta)[r_2\theta]_p$. D'après les règles de R_P, nous avons deux cas possibles pour p.

Premier cas : $p = \epsilon$. Alors $t = r_2\theta$. Nous en déduisons que les symboles de tête de s et t appartiennent à $\{a_r, b_r, \Box_r\} \cup (\Sigma_P \setminus \Gamma)$.

Second cas : p est un entier et $l_1|_p$ est une constante. Alors $l_1 \to r_1$ est la règle (8). Donc il existe un motif π de P et une position non feuille o de π tels que $s = c_{\pi,o}$. Et le symbole de tête de t appartient à $\{\tilde{f}, \tilde{u}, \tilde{v}, \tilde{w}\}$.

Donc dans les deux cas, $s \to \Box_r$ et $t \to \Box_r$ par les règles (4), (5) et (6). Donc toute paire critique de R_P est joignable et d'après le théorème 22, R_P est confluent. Finalement le système de réécriture R_P est linéaire, mince et convergent. □

Avant d'entrer dans la construction de la formule *stop*, nous énonçons quatre faits techniques (faits 241, 242, 243, et 244) du système de réécriture R_P que nous utilisons dans la suite de la démonstration du théorème 235.

Fait 241. *Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(a). Si $t \xrightarrow{r} t_2$ et r est un réétiquetage alors $p_1 = p_2 = p_3$.*

Preuve : Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(a) avec $t \to t_2$ par un réétiquetage r. Puisque r est un réétiquetage, nous avons $\mathcal{P}os(t) = \mathcal{P}os(t_2)$. Donc comme le système R_P est constitué uniquement de règles effaçantes et de réétiquetages, seuls des réétiquetages sont utilisés dans les réécritures de la figure 4(a) car $t \to t_1 \to t_2$.

Si $p_1 \neq p_3$ alors t et t_2 différent en deux positions distinctes donc on ne peut avoir $t \to t_2$ par un réétiquetage de R_P. De façon analogue $t_1 \not\to t_2$ (respectivement $t \not\to t_1$) si $p_1 \neq p_2$ (respectivement $p_2 \neq p_3$) car alors t_1 et t_2 (respectivement t et t_1) différent en deux positions distinctes. Nous en déduisons que $p_1 = p_2 = p_3$. □

Fait 242. *Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(a) tels qu'au moins une des réécritures est obtenue par application d'une règle effaçante. Alors $t \to t_2$ par une règle effaçante et $p_2 \trianglelefteq p_1$.*

Preuve : Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(a) tels qu'au moins une des réécritures est obtenue par application d'une règle effaçante. Nous considérons les cas $t \to t_1$ par une règle effaçante et $t_1 \to t_2$ par une règle effaçante, et montrons qu'alors $t \to t_2$ par une règle effaçante.

$t \to t_1$ **par une règle effaçante.** Alors $card(\mathcal{P}os(t_1)) < card(\mathcal{P}os(t))$. Or si $t \to t_2$ par un réétiquetage, on a $card(\mathcal{P}os(t_2)) = card(\mathcal{P}os(t))$, donc $t_1 \not\to t_2$ puisque le système de réécriture R_P est constitué uniquement de règles effaçantes et de réétiquetages (remarque 239). Nous en déduisons que $t \to t_2$ par une règle effaçante.

$t_1 \to t_2$ **par une règle effaçante.** Nous avons alors $card(\mathcal{P}os(t_2)) < card(\mathcal{P}os(t_1))$. De plus $card(\mathcal{P}os(t_1)) \leq card(\mathcal{P}os(t))$ puisque $t \to t_1$ et R_P est constitué uniquement de règles effaçantes et de réétiquetages. Donc $card(\mathcal{P}os(t_2)) < card(\mathcal{P}os(t))$. Nous en déduisons que $t \to t_2$ par une règle effaçante.

Finalement $t \to t_2$ par une règle effaçante. Montrons maintenant que $p_2 \trianglelefteq p_1$. Nous distinguons deux cas.

Premier cas : p_1 et p_2 sont parallèles. Alors t_1 et t_2 différent en deux positions incomparables. Donc pour que $t_1 \to t_2$, il faut que $p_3 \triangleleft p_1$ et $p_3 \triangleleft p_2$. Or

$$t_1 \underset{p_3}{\longrightarrow} t_2 \quad \Rightarrow \quad t_1(p_3) \neq t_2(p_3)$$
$$p_3 \triangleleft p_1 \text{ et } p_3 \triangleleft p_2 \quad \Rightarrow \quad t_1(p_3) = t_2(p_3)$$

Nous obtenons donc une contradiction.

Second cas : $p_1 \triangleleft p_2$. Pour toute position p parallèle à p_1 ou telle que $p \triangleleft p_1$, on a $t_1(p) = t_2(p)$. De plus $t_1(p_1) \neq t_2(p_1)$ puisque $t_2(p_1) = t(p_1)$ et $t \underset{p_1}{\longrightarrow} t_1$. Nous en déduisons que $p_3 = p_1$ d'après les règles du système R_P. Donc il existe deux règles $l_1 \to r_1$ et $l_2 \to r_2$ de R_P telles que :

$$t \xrightarrow{l_1 \to r_1} t_1 \xrightarrow{l_2 \to r_2} t_2$$

avec $tête(r_1) = tête(l_2)$ et $tête(r_2) = tête(l_1)$. Or il n'existe pas de telles règles dans R_P.

De par leur conclusion les deux cas ci-dessus sont impossibles. Nous avons donc $p_2 \trianglelefteq p_1$.

□

Fait 243. *Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(a). Alors si $t \to t_2$ par la règle (2) ou (3), la réécriture $t \to t_1$ se fait en une position effacée par la réécriture $t \to t_2$. Dons si $t \xrightarrow{(2)} t_2$ (respectivement $t \xrightarrow{(3)} t_2$), il existe α égal à 2 ou 3 (respectivement 1 ou 3), et une position p tels que $p_1 = p_2 \alpha p$.*

Preuve : Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(a). Supposons que $t \xrightarrow{(2)} t_2$. Il existe trois termes s_1, s_2, s_3 de $T(\Sigma_P)$ tels que $t|_{p_2} = u(s_1, s_2, s_3)$.

La règle (2) est effaçante donc $p_2 \trianglelefteq p_1$ d'après le fait 242. Supposons tout d'abord que $p_2 = p_1$. Alors $t(p_1) = u$. Donc si nous notons r_1 la règle telle que $t \xrightarrow[p_1]{r_1} t_1$, nous avons soit $r_1 = (2)$, soit $r_1 = (6)$, soit $r_1 = (7)$. Dans le premier cas, nous avons $t_1 = t_2$. Dans le second, $card(\mathcal{P}os(t_1)) < card(\mathcal{P}os(t_2))$. Et dans le dernier, $t_1|_{p_1} = \tilde{u}(s_1, s_2, s_3)$ et $t_2|_{p_1} = w(s_1)$ d'où $t_1|_{p_1} \not\to t_2|_{p_1}$. Donc dans tous les cas $t_1 \not\to t_2$. Nous en déduisons que $p_2 \triangleleft p_1$.

Donc il existe un entier α de [3] et une position p tels que $p_1 = p_2 \alpha p$. Supposons que $\alpha = 1$. Alors il existe un terme s_1' de $T(\Sigma_P)$ différent de s_1 tel que $t_1|_{p_2} = u(s_1', s_2, s_3)$. Nous en déduisons que $t_1 \not\to t_2$ puisque $t_2|_{p_2} = w(s_1)$. Finalement α est égal à 2 ou 3. Nous montrons de façon similaire que si $t \xrightarrow{(3)} t_2$, il existe α égal à 1 ou 3, et p position tels que $p_1 = p_2 \alpha p$. □

Fait 244. *Pour tous termes t, t_1 et t_2 de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(b), aucune règle effaçante n'est appliquée dans les différentes réécritures.*

Preuve : Soient t, t_1 et t_2 termes de $T(\Sigma_P)$ satisfaisant la propriété décrite en figure 4(b). Considérons que pour tout entier i de $[n]$, $t \xrightarrow[p_i]{r_i} t_i$.

Supposons que l'une des réécritures soit obtenue par application d'une règle effaçante. Alors nous déduisons des faits 241 et 242 que r_3 est une règle effaçante et que $p_3 \trianglelefteq p_2 \trianglelefteq p_1$. Donc r_3 est l'une des règles suivantes : (2), (3), (6) ou (8).

Nous montrons que $t_1 \to t_3$ en distinguant le cas où r_3 est la règle (2) et celui où r_3 est la règle (6). Le cas où r_3 est la règle (3) (respectivement (8)) est similaire au premier (respectivement second) cas.

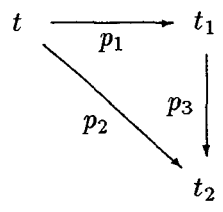
Cas où r_3 est la règle (2). D'après le fait 243, il existe α égal à 2 ou 3 et p position tels que $p_2 = p_3 \alpha p$. Donc puisque $p_2 \trianglelefteq p_1$, nous avons $t_1 \xrightarrow[p_3]{(2)} t_3$.

Cas où r_3 est la règle (6). Alors il existe un symbole β de $\Sigma_P \backslash \Gamma$, des termes $s_1, \ldots, s_{Arité(\beta)}$ de $T(\Sigma_P)$ et une constante c de $\{a_r, b_r, \square_r\}$ tels que $t|_{p_3} = \beta(s_1, \ldots, s_{Arité(\beta)})$ et $t_3|_{p_3} = c$.

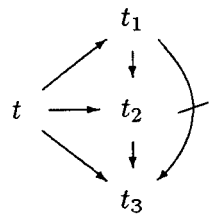
Si $p_3 \triangleleft p_1$, alors $t_1 \xrightarrow[p_3]{(6)} t_3$. Si $p_3 = p_1$, alors puisque $t_1 \not\to t_3$, $t_1(p_3)$ doit appartenir à Γ sinon $t_1 \xrightarrow[p_3]{} t_3$ par la règle (5) ou (6). Donc $t \xrightarrow[p_3]{(6)} t_1$. D'où $t_1(p_3)$ et $t_3(p_3)$ appartiennent à l'ensemble $\{a_r, b_r, \square_r\}$. Puisque $t_1 \to t_2 \to t_3$, nous en déduisons que $t_1(p_3) = a_r$, $t_2(p_3) = b_r$ et $t_3(p_3) = \square_r$ et donc $t_1 \xrightarrow[p_3]{(4)} t_3$.

Donc dans tous les cas, nous avons $t_1 \to t_3$ ce qui contredit l'hypothèse $t_1 \not\to t_3$ de la figure 4(b). Nous en déduisons qu'aucune règle effaçante n'est appliquée dans les différentes réécritures de la figure 4(b). □

FIG. 4 - Quatre propriétés techniques de R_P.



(a) Faits 241, 242 et 243

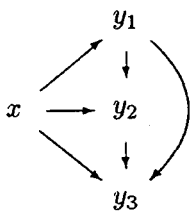


(b) Fait 244

Définissons maintenant à l'aide d'un triangle de longueur 3 la formule $impure(x)$ (définition 245) nous permettant de tester si un arbre satisfait la condition PURETÉ de la définition 175 des G-Arbres (proposition 246).

Définition 245 ($impure(x)$)**.**

$impure(x) := \exists y_1, y_2, y_3$

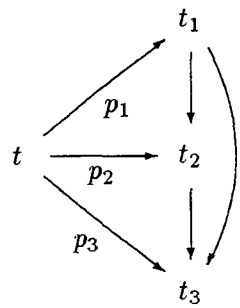


Proposition 246. *Pour tout terme t de $T(\Sigma_P)$: $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t)$ si et seulement si t est pur.*

Preuve : Soient $\varphi(x)$ la formule telle que $impure(x) = \exists y_1, y_2, y_3\, \varphi(x)$ et t un terme de $T(\Sigma_P)$.

Supposons dans un premier temps que t est impur. Il existe un contexte C de $\mathcal{C}(\Sigma_P)$, un symbole β de $\Sigma_p \setminus \Gamma$ d'arité p et p termes $t_1, \ldots, t_p$ de $T(\Sigma_P)$ tels que $t = C[\beta(t_1, \ldots, t_p)]$. Alors pour la valuation $\sigma := \{y_1 \leftarrow C[a_r], y_2 \leftarrow C[b_r], y_3 \leftarrow C[\Box_r]\}$, nous avons $\mathcal{A}_{\Sigma_P,R_P}, \sigma \models \varphi(t)$ par les règles 5 et 6. Donc $\mathcal{A}_{\Sigma_P,R_P} \models impure(t)$. Nous en déduisons que t est pur si $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t)$.

Supposons maintenant que t est pur et qu'il existe trois termes t_1, t_2, t_3 de $T(\Sigma_P)$ tels que $\mathcal{A}_{\Sigma_P,R_P}, \sigma \models \varphi(t)$ avec σ la valuation $\{y_1 \leftarrow t_1, y_2 \leftarrow t_2, y_3 \leftarrow t_3\}$, c'est-à-dire :



Puisque t est un terme pur, les règles effaçantes de R_P ne s'appliquent pas à t (remarque 239). Donc $t \to t_2$ par application d'un réétiquetage. Nous déduisons alors du fait 241 que $p_1 = p_2$. De façon analogue, nous avons $p_2 = p_3$ car $t \to t_3$ par application d'un réétiquetage. Donc $p_1 = p_2 = p_3$ et il existe un contexte C de $\mathcal{C}(\Gamma)$, un symbole α de Γ d'arité p, trois symboles β, γ, δ de Σ_P d'arité p, et p termes $s_1, \ldots, s_p$ de $T(\Sigma_P)$ tels que :

$$t = C[\alpha(s_1, \ldots, s_p)]$$
$$t_1 = C[\beta(s_1, \ldots, s_p)]$$
$$t_2 = C[\gamma(s_1, \ldots, s_p)]$$
$$t_3 = C[\delta(s_1, \ldots, s_p)]$$

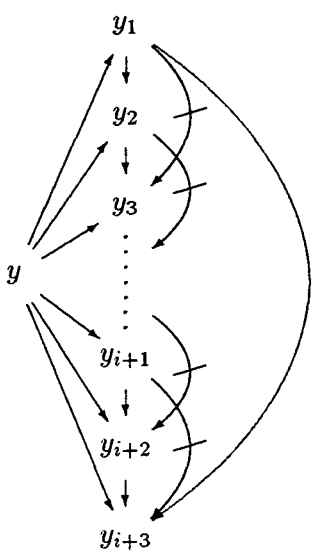
Comme seules les règles (1), (4) et (7) s'appliquent aux arbres purs (remarque 239), les symboles β, γ, δ appartiennent à l'ensemble $\{u, r, b_r, \Box_r, \tilde{f}, \tilde{u}, \tilde{r}, \tilde{w}\}$. Nous déduisons alors des règles de R_P que nous n'avons pas $t_1 \to t_2 \to t_3$ ce qui contredit les hypothèses de départ. Nous en déduisons que pour toute valuation σ, $\mathcal{A}_{\Sigma_P, R_P}, \sigma \not\models \varphi(t)$ d'où $\mathcal{A}_{\Sigma_P, R_P} \models \neg impure(t)$.

□

Nous définissons maintenant à l'aide d'un triangle de longueur $i{+}3$ la formule $match[\pi_i](x)$ (définition 247) nous permettant de tester si un arbre entoure un motif π_i de P (proposition 249).

Définition 247 ($match[\pi](x)$). *Pour tout entier i de $[n]$, soit k_i le cardinal de l'ensemble $\mathcal{IP}os(\pi_i)$ et la formule :*

$$match[\pi_i](x) := \exists y, y_1, \ldots, y_{i+3}\ x \to^{2k_i} y$$



La première partie de la formule $x \to^{2k_i} y$ réduit un motif π_i en sa constante associée $c_{\pi_i,\epsilon}$. La seconde forme un triangle utilisé afin d'identifier via sa longueur le motif dont une instance est sous-arbre de l'arbre de départ.

Remarque 248. *Nous déduisons de la remarque 238 que pour tout motif π_i de P, l'ensemble $\mathcal{IP}os(\pi_i)$ est non vide d'où k_i est non nul.*

Proposition 249. *Pour tout entier i de $[n]$ et tout terme t de $T(\Sigma)$:*

$$\mathcal{A}_{\Sigma_P,R_P} \models match[\pi_i](t) \text{ si et seulement si } t \text{ entoure } \pi_i.$$

Preuve : Soient i un entier de $[n]$, t un terme de $T(\Sigma)$ et $\varphi_i(x)$ la formule telle que $match[\pi_i](x) = \exists y, y_1, \ldots, y_{i+3}\ \varphi_i(x)$. Supposons que t entoure π_i et que π_i appartienne à $T(\Sigma, \mathcal{X}_m)$, m entier. Alors il existe un contexte C de $\mathcal{C}(\Sigma)$ et m termes $t_1, \ldots, t_m$ de $T(\Sigma)$ tels que $t = C[\pi_i[t_1, \ldots, t_m]]$.

Pour toute position o de $\mathcal{IP}os(\pi_i)$ telle que $tête(\pi_i|_o) = h$ et $Arité(h) = p$, nous avons $h(d_{\pi_i,o1}, \ldots, d_{\pi_i,op}) \xrightarrow{(7)} \tilde{h}(d_{\pi_i,o1}, \ldots, d_{\pi_i,op}) \xrightarrow{(8)} c_{\pi_i,o}$ car l'arité de h est différente de 0 puisque o est une position non feuille de π_i. Nous en déduisons que $t \to^{2k_i} C[c_{\pi_i,\epsilon}]$ par application de k_i fois la succession de réécritures $\xrightarrow{(7)}\xrightarrow{(8)}$. Donc si nous considérons la valuation $\sigma := \{y \leftarrow C[c_{\pi_i,\epsilon}], y_1 \leftarrow C[e_{i,1}], \ldots, y_{i+3} \leftarrow C[e_{i,i+3}]\}$, nous avons $\mathcal{A}_{\Sigma_P,R_P}, \sigma \models \varphi_i(t)$ d'où $\mathcal{A}_{\Sigma_P,R_P} \models match[\pi_i](t)$.

Supposons maintenant que $\mathcal{A}_{\Sigma_P,R_P} \models match[\pi_i](t)$. Il existe donc $i+4$ termes $t', t_1, \ldots, t_{i+3}$ de $T(\Sigma_P)$ tels que $\mathcal{A}_{\Sigma_P,R_P}, \sigma \models \varphi_i(t)$ avec σ la valuation $\{y \leftarrow t', y_1 \leftarrow t_1, \ldots, y_{i+3} \leftarrow t_{i+3}\}$. Donc $t', t_1, \ldots, t_{i+3}$ satisfont la propriété de la figure 5 et $t \to^{2k_i} t'$.

Tout d'abord montrons que $c_{\pi_i,\epsilon}$ est une feuille de t'. Pour tout entier j de $[i+3]$, supposons que $t' \xrightarrow[p_j]{} t_j$. D'après le fait 244, aucune règle effaçante n'est appliquée dans les réécritures de la figure 5 et d'après le fait 241, toutes les positions p_j sont égales à une position unique que nous notons p. Puisque seuls des réétiquetages sont appliqués dans la figure 5, les règles possibles sont les règles (1), (4), (5), (7), (9), (10) et (11). Soit α le symbole à la position p de t'. Remarquons que :

- Les règles (1) et (7) sont les seules règles applicables à une position non feuille. Nous déduisons du fait que les symboles de tête des parties droites de ces règles sont $u, r, \tilde{f}, \tilde{u}, \tilde{r}, \tilde{w}$ que nous ne pouvons construire un triangle en utilisant celles-ci. Donc p est une position feuille.

- Si α est une constante de Γ seule la règle (4) est applicable (remarque 239) et donc seulement des triangles de longueur 3 peuvent être construits (la longueur du triangle de la figure 5 est strictement supérieure à 3 puisque i est supérieur ou égal à un).

- S'il existe un entier k de $[i+3]$ tel que $t' \xrightarrow{(5)} t_k$, alors le triangle de la figure 5 peut se réduire par utilisation de la règle (4) ou (5).

  En effet supposons que la règle (5) soit appliquée. Nous notons k l'indice de l'arbre t_j obtenu par la première application de cette règle. Alors $t_k|_p$ est un symbole de $\{a_r, b_r, \Box_r\}$.

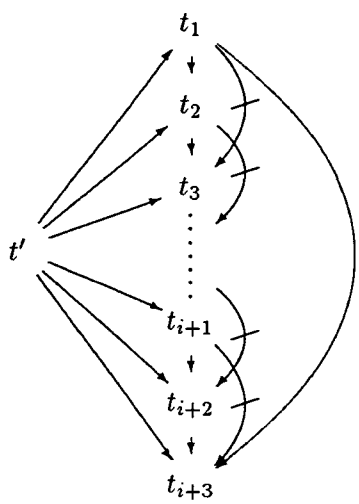
  - Si k est égal à un ou deux, alors $t' \xrightarrow{(5)} t_{k+1}$ et $t' \xrightarrow{(5)} t_{k+2}$ puisque $t_k \to t_{k+1} \to t_{k+2}$. Donc $t_k|_p = a_r$, $t_{k+1}|_p = b_r$ et $t_{k+2}|_p = \Box_r$. Finalement $t_k \xrightarrow[p]{(4)} t_{k+2}$.
  - Si k est strictement supérieur à deux, alors d'après les remarques précédentes, $t \to t_{k-2}$ par application de la règle (9), (10) ou (11). Donc il existe un entier l de $[n]$ et un entier m de $[l+3]$ tels que $t_{k-2}(p) = e_{l,m}$. Nous en déduisons que $t_{k-2} \xrightarrow[p]{(5)} t_k$.

- Puisque R_P termine (proposition 240), les arbres t_j sont deux à deux distincts. De plus, parmi les règles (9), (10), (11), seule la règle (9) permet d'obtenir deux arbres distincts à partir de t'.

Nous déduisons de ces remarques que pour tout entier j de $[i+3]$, $t' \xrightarrow{(9)} t_j$. Donc l'un des symboles de $\{c_{\pi_j,\epsilon} \mid j \in [n]\}$ est une feuille de t' et pour tout entier j de $[i+2]$, $t_j \xrightarrow[p]{(10)} t_{j+1}$ ou $t_j \xrightarrow[p]{(11)} t_{j+1}$. Nous en déduisons qu'il existe un entier k de $[n]$ et un entier l tels que pour tout entier j de $[i+3]$, $t_j(p) = e_{k,j+l}$. De plus comme $t_1 \xrightarrow[p]{} t_{i+3}$, nous avons $k = i$ et $l = 0$. Donc $c_{\pi_i,\epsilon}$ est une feuille de t'.

Nous devons maintenant montrer que puisque $t \rightarrow^{2k_i} t'$ et $c_{\pi_i,\epsilon}$ est feuille de t', t entoure π_i. Nous déduisons cela du fait que la seule possibilité pour que $c_{\pi_i,\epsilon}$ apparaisse dans t' à partir de t appartenant à $T(\Sigma)$ en $2k_i$ réécritures est d'utiliser k_i fois la succession de réécritures $\xrightarrow{(7)} \xrightarrow{(8)}$. Finalement si $\mathcal{A}_{\Sigma_P,R_P} \models \mathit{match}[\pi_i](t)$, nous obtenons que t entoure π_i. □

FIG. 5 -



Définissons maintenant la formule $\mathit{égal}(x)$ (définition 250) nous permettant de tester qu'un arbre satisfait la condition EGALITÉ de la définition 175 des G-Arbres (proposition 251).

Définition 250 ($\acute{e}gal(x)$).

$$\begin{aligned}\acute{e}gal(x) :=& \forall y, z, z', s, s', y' \\ & \Big(x \to^2 y \wedge match\big[f(r(x_1,x_2,x_3), u(x_4,x_5,x_6), x_7)\big](y) \\ & \wedge y \to z \wedge match\big[u(r(x_1,x_2,x_3), u(x_4,x_5,x_6), x_7)\big](z) \\ & \wedge z \to s \wedge match\big[u(w(x_1), u(x_2,x_3,x_4), x_5)\big](s) \\ & \wedge s \to y' \wedge match\big[w(w(x_1))\big](y') \\ & \wedge y \to z' \wedge match\big[r(r(x_1,x_2,x_3), u(x_4,x_5,x_6), x_7\big](z') \\ & \wedge z' \to s' \wedge match\big[r(r(x_1,x_2,x_3), w(x_4), x_5)\big](s')\Big) \;\Rightarrow\; s' \to y'\end{aligned}$$

Proposition 251. *Pour tout terme t de $T(\Sigma_P)$:*

$$\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t) \wedge \acute{e}gal(t)$$
si et seulement si
t est pur et satisfait la condition EGALITÉ *de la définition 175.*

Preuve : Soient t un terme de $T(\Sigma_P)$ et $\varphi(x)$ la formule telle que $\acute{e}gal(x) = \forall y, z, z', s, s', y'\varphi(x)$. Supposons que $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t) \wedge \acute{e}gal(t)$. Tout d'abord t est pur d'après le proposition 246 puisque $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t)$.

Montrons maintenant que t satisfait la condition EGALITÉ de la définition 175. Supposons qu'il existe sept termes $s_1, \ldots, s_7$ de $T(\Gamma)$ tels que $f(f(s_1,s_2,s_3), f(s_4,s_5,s_6), s_7)$ soit un sous-arbre de t. Il existe donc C contexte de $\mathcal{C}(\Gamma)$ tel que :

$$t = C\big[f(f(s_1,s_2,s_3), f(s_4,s_5,s_6), s_7)\big].$$

Nous en déduisons les propriétés suivantes :

$$\begin{aligned}
& t \xrightarrow{(1)} \xrightarrow{(1)} t_1 \;\wedge\; match\big[f(r(x_1,x_2,x_3), u(x_4,x_5,x_6), x_7)\big](t_1) \\
& \qquad\qquad \text{avec } t_1 = C\big[f(r(s_1,s_2,s_3), u(s_4,s_5,s_6), s_7)\big] \\
& t_1 \xrightarrow{(1)} t_2 \;\wedge\; match\big[u(r(x_1,x_2,x_3), u(x_4,x_5,x_6), x_7)\big](t_2) \\
& \qquad\qquad \text{avec } t_2 = C\big[u(r(s_1,s_2,s_3), u(s_4,s_5,s_6), s_7)\big] \\
& t_2 \xrightarrow{(3)} t_3 \;\wedge\; match\big[u(w(x_2), u(x_2,x_3,x_4), x_5)\big](t_3) \\
& \qquad\qquad \text{avec } t_3 = C\big[u(w(s_2), u(s_4,s_5,s_6), s_7)\big] \\
& t_3 \xrightarrow{(2)} t_4 \;\wedge\; match\big[w(w(x_2))\big](t_4) \\
& \qquad\qquad \text{avec } t_4 = C\big[w(w(s_2))\big] \\
& t_1 \xrightarrow{(1)} t_5 \;\wedge\; match\big[r(r(x_1,x_2,x_3), u(x_4,x_5,x_6), x_7)\big](t_5) \\
& \qquad\qquad \text{avec } t_5 = C\big[r(r(s_1,s_2,s_3), u(s_4,s_5,s_6), s_7)\big] \\
& t_5 \xrightarrow{(2)} t_6 \;\wedge\; match\big[r(r(x_1,x_2,x_3), w(x_4), x_5)\big](t_6) \\
& \qquad\qquad \text{avec } t_6 = C\big[r(r(s_1,s_2,s_3), w(s_4), s_7)\big]
\end{aligned}$$

Puisque $\mathcal{A}_{\Sigma_P,R_P} \models \acute{e}gal(t)$, nous avons $\mathcal{A}_{\Sigma_P,R_P}, \sigma \models \varphi(t)$ avec σ la valuation $\{y \leftarrow t_1, z \leftarrow t_2, s \leftarrow t_3, y' \leftarrow t_4, z' \leftarrow t_5, s' \leftarrow t_6\}$. Nous en déduisons que $t_6 \to t_4$. La règle (3) est la seule

applicable à cette réécriture, donc $t_4 = C[w(w(s_2))] = C[w(w(s_4))]$ d'où $s_2{=}s_4$. Finalement t satisfait la condition EGALITÉ de la définition 175. Donc si $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t) \wedge égal(t)$ alors t est pur et satisfait la condition EGALITÉ de la définition 175.

Supposons maintenant que t soit pur et satisfasse la condition EGALITÉ de la définition 175. Tout d'abord $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t)$ d'après la proposition 246.

Montrons que $\mathcal{A}_{\Sigma_P,R_P} \models égal(t)$. Supposons qu'il existe six termes $t_1, \ldots, t_6$ de $T(\Sigma_P)$ tels que :

$$\begin{array}{rlll} & t \to^2 t_1 & \wedge & match\big[f(r(x_1,x_2,x_3),u(x_4,x_5,x_6),x_7)\big](t_1) \\ \wedge & t_1 \to t_2 & \wedge & match\big[u(r(x_1,x_2,x_3),u(x_4,x_5,x_6),x_7)\big](t_2) \\ \wedge & t_2 \to t_3 & \wedge & match\big[u(w(x_2),u(x_2,x_3,x_4),x_5)\big](t_3) \\ \wedge & t_3 \to t_4 & \wedge & match\big[w(w(x_2))\big](t_4) \\ \wedge & t_1 \to t_5 & \wedge & match\big[r(r(x_1,x_2,x_3),u(x_4,x_5,x_6),x_7)\big](t_5) \\ \wedge & t_5 \to t_6 & \wedge & match\big[r(r(x_1,x_2,x_3),w(x_4),x_5)\big](t_6) \end{array}$$

Il existe donc un contexte C de $\mathcal{C}(\Sigma_P)$ et sept termes $s_1, \ldots, s_7$ de $T(\Sigma_P)$ tels que $t_1 = C[f(r(s_1,s_2,s_3),u(s_4,s_5,s_6),s_7)]$. Puisque t est pur, nous déduisons des règles du système R_P que $t = C[f(f(s_1,s_2,s_3),f(s_4,s_5,s_6),s_7)]$, C est un contexte de $\mathcal{C}(\Gamma)$ et $s_1, \ldots, s_7$ des termes purs. D'où $t_4 = C[w(w(s_2))]$ et $t_6 = C[r(r(s_1,s_2,s_3),w(s_4),s_7)]$.

De plus comme t satisfait la condition EGALITÉ de la définition 175, nous avons $s_2 = s_4$ et donc $t_6 \xrightarrow{(3)} t_4$. Finalement pour toute valuation σ de $\{y,z,z',s,s',y'\}$ dans $T(\Sigma_P)$, $\mathcal{A}_{\Sigma_P,R_P},\sigma \models \varphi(t)$ d'où $\mathcal{A}_{\Sigma_P,R_P} \models \neg impure(t) \wedge égal(t)$. □

Les formules $impure(x)$ et $égal(x)$ nous permettent donc de tester si un arbre satisfait les conditions PURETÉ et EGALITÉ de la définition 175 des G-Arbres (propositions 246 et 251). Nous en déduisons la formule $grille(x)$ (définition 252) qui teste si un arbre est un G-Arbre sur Γ (proposition 255).

Définition 252 ($grille(x)$).

$grille(x) :=$

$$\neg impure(x) \qquad \text{(R-PURE)}$$

$$\wedge \bigwedge_{h\in\Sigma_P\setminus\{\perp_0,f\}} \Big(\neg match\big[f(h(x_3,\ldots,x_{2+Arité(h)}),x_1,x_2)\big](x) \wedge \neg match\big[f(x_1,h(x_3,\ldots,x_{2+Arité(h)})),x_2)\big](x)\Big) \qquad \text{(FILS1\&2)}$$

$$\wedge \bigwedge_{h\in\{f,\perp_0\}\cup(\Sigma_P\setminus\Gamma)} \neg match\big[f(x_1,x_2,h(x_3,\ldots,x_{2+Arité(h)}))\big](x) \qquad \text{(R-FEUILLE)}$$

$$\wedge\ égal(x) \qquad \text{(R-EGAL)}$$

$$\wedge \bigwedge_{h\in\Sigma_P\setminus\{f\}} \neg match\big[f(f(x_1,h(x_5,\ldots,x_{4+Arité(h)}),x_2),x_3,x_4)\big](x) \qquad \text{(R-CORPS1)}$$

$$\wedge \bigwedge_{h\in\Sigma_P\setminus\{f\}} \neg match\big[f(h(x_7,\ldots,x_{6+Arité(h)}),f(f(x_1,x_2,x_3),x_4,x_5),x_6)\big](x) \qquad \text{(R-CORPS2)}$$

Remarque 253. *Dans les formules, lorsque h est une constante, $h(x_i, \ldots, x_{i-1+Arité(h)}) = h$.*

Remarque 254. *La formule $grille(x)$ appartient au fragment $\forall^*$ car les occurrences des formules grille et match sont négatives.*

Proposition 255. *Soient la signature $\Sigma_{P'} \supseteq \Sigma$ et le système de réécriture $R_{P'}$ construits par application de la définition 237, où P' est l'ensemble des motifs mentionnés dans les formules match de la définition 252. Alors un arbre t de $T(\Sigma_{P'})$ est un G-Arbre sur Γ si et seulement si $\mathcal{A}_{\Sigma_{P'},R_{P'}} \models grille(t)$.*

Remarque 256. *L'ensemble de motifs P' satisfait la restriction imposée par la définition 237 puisque $P' \cap (\Gamma_0 \cup \mathcal{X}) = \emptyset$. Donc $\Sigma_{P'}$ et $R_{P'}$ sont bien définis.*

Preuve : Soit t un arbre de $T(\Sigma_{P'})$. Dans le cas où $\mathcal{A}_{\Sigma_{P'},R_{P'}} \models \neg impure(t)$, nous avons t est pur d'après la proposition 246. Donc t est un terme de $T(\Sigma)$ ce qui nous permet d'appliquer le proposition 249. Nous avons donc les propriétés suivantes :

$$\mathcal{A}_{\Sigma_{P'},R_{P'}} \models \neg impure(t) \wedge \bigwedge_{h \in \Sigma_P \setminus \{\perp_0, f\}} \Big(\neg match\big[f(h(x_3, \ldots, x_{2+Arité(h)}), x_1, x_2)\big](t) \wedge \neg match\big[f(x_1, h(x_3, \ldots, x_{2+Arité(h)})), x_2)\big](t) \Big)$$

si et seulement si

t est pur et satisfait les conditions FILS-1 et FILS-2 de la définition 175.

$$\mathcal{A}_{\Sigma_{P'},R_{P'}} \models \neg impure(t) \wedge \textstyle\bigwedge_{h \in \{f, \perp_0\} \cup (\Sigma_P \setminus \Gamma)} \neg match\big[f(x_1, x_2, h(x_3, \ldots, x_{2+Arité(h)}))\big](t)$$

si et seulement si

t est pur et satisfait la condition CONSTANTES de la définition 175.

$$\mathcal{A}_{\Sigma_{P'},R_{P'}} \models \neg impure(t) \wedge \bigwedge_{h \in \Sigma_P \setminus \{f\}} \neg match\big[f(f(x_1, h(x_5, \ldots, x_{4+Arité(h)}), x_2), x_3, x_4)\big](t)$$

si et seulement si

t est pur et satisfait la condition CORPS-1 de la définition 175.

$$\mathcal{A}_{\Sigma_{P'},R_{P'}} \models \neg impure(t) \wedge \bigwedge_{h \in \Sigma_P \setminus \{f\}} \neg match\big[f(h(x_7, \ldots, x_{6+Arité(h)}), f(f(x_1, x_2, x_3), x_4, x_5), x_6)\big](t)$$

si et seulement si

t est pur et satisfait la condition CORPS-2 de la définition 175.

En tenant compte de ces propriétés et de la proposition 251, nous obtenons que t est un G-Arbre sur Γ si et seulement si $\mathcal{A}_{\Sigma_{P'},R_{P'}} \models grille(t)$. □

Maintenant que nous savons reconnaître à partir d'une formule les G-Arbres, nous allons utiliser la proposition 177 pour reconnaître les G-Arbres codant les calculs réussis de la machine $\mathcal{M}$ débutant sur l'entrée vide. Nous définissons pour cela les formules $\mathit{init}(x)$, $\mathit{calcul}(x)$ et $\mathit{final}(x)$ qui nous permettent de vérifier si un arbre satisfait les conditions énoncées par la proposition 177. Ensuite nous définissons la formule *stop* qui code le problème de l'arrêt des machines de Turing en une formule du fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas. Finalement nous en déduisons la preuve du théorème 235.

Définition 257 (*stop*).

$$\begin{aligned}
\mathit{init}(x) &:= \mathit{match}\big[f(x_1, f(x_2, f(x_3, \bot_0, \Box_r), \Box_r), q_s)\big](x) \\
&\quad \wedge \neg \mathit{match}\big[f(f(x_1, x_2, q_s), x_3, x_4)\big](x) \\
&\quad \wedge \neg \mathit{match}\big[f(x_1, f(x_2, x_3, q_s), x_4)\big](x) \\
\mathit{calcul}(x) &:= \bigwedge_{p \in P'_T} \neg \mathit{match}\big[p\big](x) \\
\mathit{final}(x) &:= \mathit{match}\big[f(\bot_0, x_1, q_f)\big](x) \\
\mathit{stop} &:= \exists x \, \big(\mathit{grille}(x) \wedge \mathit{init}(x) \wedge \mathit{calcul}(x) \wedge \mathit{final}(x)\big)
\end{aligned}$$

Preuve : théorème 235. Soit P l'ensemble de tous les motifs utilisés dans les constructions précédentes. P satisfait la restriction imposée par la définition 237 puisque $P \cap (\Gamma_0 \cup \mathcal{X}) = \emptyset$ donc nous pouvons construire $\Sigma_P \supseteq \Sigma$ et R_P par application de la définition 237. Soit t un terme de $T(\Sigma_P)$. Nous déduisons de la proposition 249, la propriété suivante :

$$\mathcal{A}_{\Sigma_P, R_P} \models \neg \mathit{impure}(t) \wedge \mathit{init}(t) \wedge \mathit{calcul}(t) \wedge \mathit{final}(t)$$
si et seulement si
t est pur et satisfait les conditions de la proposition 177.

En tenant compte de cette propriété et des propositions 255 et 177, nous obtenons :

$$\mathcal{A}_{\Sigma_P, R_P} \models \mathit{grille}(t) \wedge \mathit{init}(t) \wedge \mathit{calcul}(t) \wedge \mathit{final}(t)$$
si et seulement si
t est G-Arbre sur Γ codant un calcul réussi de la machine $\mathcal{M}$ débutant sur l'entrée vide.

Donc la machine de Turing s'arrête sur l'entrée vide si et seulement si $\mathcal{A}_{\Sigma_P, R_P} \models \mathit{stop}$. Or le système de réécriture R_P est linéaire, mince et convergent (proposition 240) et la formule *stop* a une forme prénexe appartenant au fragment $\exists^*\forall^*$ donc c'est une formule de ce fragment. Donc nous déduisons de l'indécidabilité du problème de l'arrêt des machines de Turing sur l'entrée vide (théorème 166) que le fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas pour les systèmes de réécriture linéaires, minces et convergents est indécidable. □

1.3 Décidabilité

Dans la section précédente, nous avons montré l'indécidabilité d'un fragment très restreint de la théorie du premier ordre de la réécriture en un pas pour des systèmes de réécriture également très restreints (fragment $\exists^*\forall^*$, et systèmes linéaires, minces et convergents).

Nous nous intéressons maintenant à la décidabilité du fragment existentiel de cette théorie. Ce problème est ouvert mais Niehren *et al.* [66] ont montré que son fragment existentiel positif est décidable. En fait, toute formule de ce fragment est équivalente en terme de solutions à un problème d'unification du second ordre stratifiée, les problèmes de ce type étant montrés décidables par M. Schmidt-Schauss [77]. Dans [46], F. Jacquemard transforme toute formule du fragment existentiel positif de la théorie de la réécriture en un pas en une formule décidable de la théorie faible monadique à deux successeurs, mais ce résultat n'est valable que pour les systèmes de réécriture tels que pour toute règle $l \to r$, toutes les occurrences d'une même variable dans $l \to r$ sont à la même profondeur.

Dans ce chapitre, nous montrons la décidabilité du fragment existentiel de la théorie du premier ordre de la réécriture en un pas pour les systèmes de réécriture ultra-minces (toutes les occurrences de variables apparaissant dans les membres gauche et droit des règles sont à la profondeur un). Plus exactement notre résultat concerne la théorie (définition 258) dont les formules sont construites sur les formules atomiques suivantes : $x \to_{\mathcal{R}} y$, $x = y$ et $x \in \mathcal{L}$ où $\mathcal{L}$ est un langage reconnaissable par automates à tests entre frères (*RATF*).

Définition 258. *Soient Σ une signature, $\mathcal{X}$ un ensemble de variables, $R_1, \ldots, R_u$ des systèmes de réécriture et $L_1, \ldots, L_q$ des langages de la classe RATF. Soit l'ensemble P constitué des prédicats suivants :*

- $\to_{R_i}$ *pour tout entier i de $[u]$: prédicat binaire sur $T(\Sigma)$.*
- $\in L_i$ (contrainte d'appartenance) *pour tout entier i de $[q]$: prédicat unaire sur $T(\Sigma)$.*
- $=$ (contrainte d'égalité) *: prédicat binaire sur $T(\Sigma)$.*

La théorie du premier ordre de la réécriture en un pas étendue est l'ensemble des formules logiques du premier ordre construites sur Σ, $\mathcal{X}$, P et les formules atomiques $x \to_{\mathcal{R}_i} y$ pour tout i de $[u]$, $x \in L_i$ pour tout i de $[n]$ et $x = y$ avec x et y variables. La structure de cette théorie, notée $\mathcal{T}$, est définie comme suit :

- *Le support est l'ensemble des termes construits sur Σ c'est-à-dire $T(\Sigma)$,*
- *L'élément (respectivement la fonction) associé à toute constante e (respectivement tout symbole de fonction d'arité non nulle) est canonique, dans le sens où pour toute constante e de Σ (respectivement pour tout symbole de fonction f d'arité non nulle), $e^{\mathcal{T}} = e$ (respectivement $f^{\mathcal{T}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ pour tous termes $t_1, \ldots, t_n$ de $T(\Sigma)$).*
- *Pour tout i de $[u]$, la relation $\to^{\mathcal{T}}$ associée au symbole de prédicat $\to_{R_i}$ est la relation de réécriture associée à R_i sur $T(\Sigma)$.*
- *Pour tout i de $[q]$, la relation $(\in L_i)^{\mathcal{T}}$ associée au symbole de prédicat $\in L_i$ est la relation d'appartenance ensembliste au langage L_i.*

- *La relation* $=^{\mathcal{T}}$ *associée au symbole de prédicat* $=$ *est l'égalité sur* $T(\Sigma)$.

Donc si s et t sont deux termes de $T(\Sigma)$, $\mathcal{T} \models (t \in L_i)$, pour i élément de $[q]$, si et seulement si t appartient au langage L_i, et $\mathcal{T} \models (s = t)$ si et seulement si s et t sont le même terme.

Remarque 259. *Les systèmes de réécriture* $R_1, \ldots, R_u$ *et les langages* $L_1, \ldots, L_q$ *sont des paramètres de la théorie définie ci-dessus mais nous ne les mentionnons pas afin de simplifier les notations.*

Nous allons montrer dans la suite de ce chapitre le théorème suivant :

Théorème 260. *Le fragment existentiel de la théorie du premier ordre de la réécriture en un pas étendue pour les systèmes de réécriture ultra-minces est décidable.*

Nous donnons dans ce chapitre une procédure de décision basée sur des règles d'inférence pour résoudre les formules existentielles construites sur les formules atomiques $x \to_{\mathcal{R}} y$, $x = y$ et $x \in \mathcal{L}$, $\mathcal{L}$ appartenant à la classe $RATF$.

Par souci de clarté, nous considérons des formules utilisant les prédicats $\xrightarrow{r}$ où r est une règle de réécriture, plutôt que les prédicats $\to_{\mathcal{R}_i}$. La première présentation est plus commode pour l'exposition de notre algorithme et est équivalente à la seconde puisque $\mathcal{T} \models (x \to_{\mathcal{R}_i} y)$ si et seulement si $\mathcal{T} \models (\bigvee_{r \in \mathcal{R}_i} x \xrightarrow{r} y)$.

De plus, si les formules sont écrites en forme normale disjonctive, $\varphi = \bigvee_{i \in I} \varphi_i$, alors φ est satisfiable si et seulement si au moins l'une des formules φ_i est satisfiable. Dans la suite de ce chapitre, nous pouvons donc considérer uniquement le cas des formules sans disjonction. Enfin, nous désignons par $\mathcal{R}$ le système de réécriture $R_1 \cup \cdots \cup R_u$.

Les formules que nous souhaitons obtenir par notre algorithme sont des formules sans contrainte de réécriture et sans contrainte de non réécriture. Donc le but de celui-ci est d'éliminer celles-ci. Pour une formule avec n variables, l'algorithme essaye de trouver n termes satisfaisant la formule de la façon suivante. A chaque pas, il devine dans un premier temps un symbole de tête pour chaque variable x, c'est-à-dire qu'une substitution de la forme $x \leftarrow f(x_1, \ldots, x_n)$ est appliquée. Ainsi les nouvelles variables $x_1, \ldots, x_n$ sont ajoutées au problème. Dans un second temps, l'algorithme essaye de satisfaire chaque relation de réécriture $f(x_1, \ldots, x_n) \xrightarrow{r} g(y_1, \ldots, y_m)$ soit à la racine de $f(x_1, \ldots, x_n)$, soit en-dessous. Chaque cas nécessite d'ajouter aux problèmes des égalités, ou des différences, ou des relations de réécriture entre les nouvelles variables. Dans le cas d'une relation de non réécriture $f(x_1, \ldots, x_n) \not\xrightarrow{r} g(y_1, \ldots, y_m)$, un traitement analogue est effectué: l'algorithme essaye de contredire la règle r au lieu de la satisfaire.

Considérons l'exemple suivant :

$$P_0 := \exists x, y, \quad x \xrightarrow{r} y \wedge y \not\xrightarrow{r} x \wedge x \neq y$$

avec r la règle $f(\alpha, \beta) \to f(a, \alpha)$ où α, β sont des variables, a une constante et f un symbole d'arité 2. Tout d'abord, des symboles de têtes sont devinés pour x et y. Par exemple, $x = f(x_1, x_2)$ et $y = f(y_1, y_2)$. Nous obtenons la formule suivante :

$$P_1 := \exists x_1, x_2, y_1, y_2, \quad f(x_1, x_2) \xrightarrow{r} f(y_1, y_2) \wedge f(y_1, y_2) \not\xrightarrow{r} f(x_1, x_2) \wedge f(x_1, x_2) \neq f(y_1, y_2).$$

La contrainte de réécriture $f(x_1, x_2) \xrightarrow{r} f(y_1, y_2)$ de P_1 est :

- Soit appliquée à la racine, et alors la contrainte de réécriture est effaçée et la formule $x_1 = y_2 \wedge y_1 = a$ est ajoutée,

- Soit propagée sur l'une des branches de la racine, par exemple la contrainte est transformée en $x_1 \xrightarrow{r} y_1 \wedge x_2 = y_2$.

Concernant la partie négative de P_1, nous devons vérifier que la contrainte de non réécriture ne s'applique pas à la racine, c'est-à-dire que $x_1 \neq a$ ou $y_1 \neq x_2$, et de plus qu'elle ne s'applique pas sous celle-ci. Puisque la contrainte $x \neq y$ apparaît dans P_0, nous pouvons distinguer trois cas et ajouter de façon non déterministe l'une des trois formules :

$$y_1 \overset{r}{\not\to} x_1 \wedge y_1 \neq x_1 \wedge y_2 = x_2 \qquad y_2 \overset{r}{\not\to} x_2 \wedge y_2 \neq x_2 \wedge y_1 = x_1 \qquad y_1 \neq x_1 \wedge y_2 \neq x_2$$

Après cela, nous obtenons une nouvelle formule et le nombre de contraintes de réécriture et le nombre de contraintes de non réécriture n'ont pas augmenté. Pour s'assurer de la terminaison, nous prouvons que la partie de la formule non concernée par les réécritures peut être résolue indépendamment. Donc le nombre de parties réécriture différentes pour un nombre de variables donné étant borné, la terminaison est assurée.

Au contraire si au lieu de $x \neq y$, la contrainte $x = y$ apparaît dans P_0, nous obtenons la contrainte $f(x_1, x_2) \overset{r}{\not\to} f(x_1, x_2)$, et devons donc dire que $x_1 \overset{r}{\not\to} x_1$ et $x_2 \overset{r}{\not\to} x_2$. Donc le nombre de contraintes de non réécriture augmente. Mais une propriété importante des systèmes de réécriture ultra-minces est que l'ensemble des solutions des contraintes de la forme $x \overset{r}{\not\to} x$ est reconnaissable par automates à tests entre frères. Les contraintes $x \overset{r}{\not\to} x$ sont donc transformées en contraintes d'appartenance à des langages de *RATF*. Les contraintes d'appartenance évitent donc d'augmenter le nombre de réécritures. Elles augmentent également le pouvoir d'expression des formules considérées et simplifient les preuves et constructions (elles permettent par exemple de prendre en compte des termes clos).

L'expressivité du fragment considéré est relativement faible par rapport aux propriétés classiques utiles en théorie de la réécriture. Par exemple, les formules peuvent exprimer la confluence seulement pour un nombre borné de pas (à cause de la restriction "en un pas") et pour les systèmes de réécriture ultra-minces. Néanmoins, les résultats d'indécidabilité vus dans la section précédente et les connexions avec l'unification contextuelle montrent que la résolution des formules de réécriture en un pas est difficile et intéressante en elle-même.

Nos méthodes pourraient être adaptées aux systèmes de réécriture quelconques mais la terminaison de l'algorithme serait beaucoup plus difficile à obtenir. Il semble que cette difficulté apparaît dès que des règles effaçantes sont autorisées.

Remarque 261. *Toutes les variables apparaissant dans les formules que nous considérons sont quantifiées existentiellement. Donc afin d'alléger les notations, nous omettons les quantificateurs dans la suite de ce chapitre.*

De plus, nous ne faisons pas la distinction entre une formule P et la formule Q obtenue en supprimant les quantifications de P. Donc par abus de notation nous disons que P est satisfiable si et seulement s'il existe une valuation σ de $Var(P)$ dans $T(\Sigma)$ telle que $\mathcal{T}, \sigma \models P$.

Ce chapitre est organisé de la façon suivante. Nous montrons tout d'abord que l'on peut se débarrasser des expressions de la forme $x \overset{r}{\not\to} x$ en utilisant des automates à tests entre frères (section 1.3.1). Ensuite, nous présentons l'algorithme de décision (section 1.3.2), le système de règles se trouvant en section 1.3.5. Dans la section 1.3.3, nous donnons la preuve du théorème 260. Finalement, nous discutons des extensions possibles de la méthode que nous utilisons (section 1.3.4).

1.3.1 Relations de non réécriture et *ATF*

Notre procédure de décision doit traiter des expressions de la forme $x \not\xrightarrow{\mathcal{T}} x$. Nous allons voir qu'en utilisant des automates à tests entre frères, on peut se débarrasser de ces contraintes. Nous allons en fait remplacer les contraintes de réécriture $x \not\xrightarrow{\mathcal{T}} x$ par des contraintes d'appartenance à des langages de *RATF*. Ces contraintes d'appartenance seront également utilisées pour le traitement des parties closes des règles de réécriture.

Cette transformation des contraintes de réécriture $x \not\xrightarrow{\mathcal{T}} x$ en des contraintes d'appartenance conduit à la définition de langages de *RATF* ne se trouvant pas dans notre problème de départ.

Notations

Les règles de réécriture peuvent contenir des termes clos (par exemple, $g(a)$ dans la règle $f(g(a), x) \to f(x, x)$). Nous considérons une famille $L_{q+1}, \ldots, L_{q'}$ de singletons telle que tout singleton contient un terme clos apparaissant dans une règle de $\mathcal{R}$ et tout terme clos appartient à un unique singleton. Puisque les langages $L_{q+1}, \ldots, L_{q'}$ sont finis, ils sont réguliers et donc appartiennent à la classe *RATF* (propriétés 130 et 132).

Nous considérons aussi les langages L_r définis par $\{t \in T(\Sigma) \mid t \not\xrightarrow{\mathcal{T}} t\}$ et nous allons montrer que ces langages appartiennent à *RATF*. Les langages L_r sont indicés de la façon suivante $L_{q'+1}, \ldots, L_n$. Donc tout langage de *RATF* que nous considérons appartient à $\{L_1, \ldots, L_n\}$.

Connexions

Proposition 262. *Pour toute règle r de $\mathcal{R}$, le langage $\{t \in T(\Sigma) \mid t \not\xrightarrow{\mathcal{T}} t\}$ appartient à la classe RATF.*

Preuve : Soit une règle $r := l_r \to r_r$ de $\mathcal{R}$. Un terme t de $T(\Sigma)$ vérifie $t \xrightarrow{r} t$ si et seulement s'il existe une position p de t telle que $t|_p$ soit une instance close de l_r et r_r. Donc si nous notons σ l'un des unificateurs les plus généraux de l_r et r_r, l'ensemble $\{t \in T(\Sigma) \mid t \xrightarrow{r} t\}$ est l'ensemble des termes entourant $l_r\sigma$. Nous pouvons montrer que $l_r\sigma$ est semi-linéaire puisque le système $\mathcal{R}$ est ultra-mince. Donc puisque l'ensemble des termes entourant un terme semi-linéaire est reconnaissable par *ATF* (propriété 122) et que la classe *RATF* est close par intersection (propriété 114), le langage $\{t \in T(\Sigma) \mid t \xrightarrow{r} t\}$ est dans *RATF*.

Nous déduisons alors de la clôture par complément de la classe *RATF* (propriété 114) que $\{t \in T(\Sigma) \mid t \not\xrightarrow{\mathcal{T}} t\}$ appartient à la classe *RATF* . □

Automate produit

La proposition précédente nous permet de transformer les contraintes de réécriture de la forme $x \not\xrightarrow{\mathcal{T}} x$ en des contraintes d'appartenance de la forme $x \in L_r$. De façon naturelle, les conjonctions de non réécriture de la forme $x \not\xrightarrow{\mathcal{T}} x \wedge x \not\xrightarrow{\mathcal{T}'} x$ sont transformées en contraintes d'appartenance de la forme $x \in L_r \cap L_{r'}$. Or il est plus pratique de pouvoir considérer un seul automate pour manipuler toutes les conjonctions possibles. Ceci est réalisé par une construction classique d'*automate produit*. Nous donnons ci-dessous la construction d'un tel automate et ses propriétés.

Soient $\mathcal{A}_1, \ldots, \mathcal{A}_n$ des automates à tests entre frères normalisés-complets (propriété 113) reconnaissant respectivement les langages $L_1, \ldots, L_n$. Chaque automate $\mathcal{A}_i$ est déterminé par

un quadruplet $(\Sigma, \mathcal{S}_i, \mathcal{S}_{f_i}, \Delta_i)$ et nous supposons que les ensembles d'états sont deux à deux disjoints. L'*automate produit* $\mathcal{A}$ est défini par $(\Sigma, \mathcal{S}, \mathcal{S}_f, \Delta)$ où $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ et Δ est l'ensemble des règles de la forme :

$$f(s^1, \ldots, s^m) \xrightarrow{[c]} s$$

où c est une contrainte pleine, chaque s^j un n-uplet $(s_1^j, \ldots, s_n^j)$ de $\mathcal{S}$, s un n-uplet $(s_1, \ldots, s_n)$ de $\mathcal{S}$ et pour tout i de $[m]$, la règle $f(s_i^1, \ldots, s_i^m) \xrightarrow{[c]} s_i$ appartient à Δ_i.

Nous pouvons montrer que l'automate produit $\mathcal{A}$ est un automate à tests entre frères normalisé-complet puisque chaque automate $\mathcal{A}_i$ l'est. Donc il existe pour tout terme t de $T(\Sigma)$ un état s tel que t appartienne à $\mathcal{L}(\mathcal{A}, s)$. Nous pouvons également montrer que pour tout terme t de $T(\Sigma)$ et n-uplet $(s_1, \ldots, s_n) = s$ de $\mathcal{S}$, $t \to_{\mathcal{A}}^* s(t)$ si et seulement si pour tout i de $[n]$, $t \to_{\mathcal{A}_i}^* s_i(t)$.

L'ensemble des états finaux $\mathcal{S}_f$ est inutile dans notre étude et nous nous intéressons uniquement aux ensembles de termes $\mathcal{L}(\mathcal{A}, s)$ pour tout état s de $\mathcal{S}$ et $\mathcal{L}(\mathcal{A}, \mathcal{S}) = \{t \in \mathcal{L}(\mathcal{A}, s) \mid s \in S\}$. En effet, si s_i est un état de l'automate $\mathcal{A}_i$ et t un terme tel que $t \to_{\mathcal{A}_i}^* s_i(t)$, alors il existe $s = (s_1, \ldots, s_n)$ tel que $t \to_{\mathcal{A}}^* s(t)$. De plus pour tout terme t tel que $t \to_{\mathcal{A}}^* s(t)$, nous avons $t \to_{\mathcal{A}_i}^* s_i(t)$. Donc si s_i est un état final de l'automate $\mathcal{A}_i$, $\mathcal{L}(\mathcal{A}, s) \subseteq L_i$. Et si s_i n'est pas un état final de l'automate $\mathcal{A}_i$, alors tout terme t tel que $t \to_{\mathcal{A}}^* s(t)$ n'appartient pas à L_i puisque $t \to_{\mathcal{A}_i}^* s_i(t)$ et $\mathcal{A}_i$ est déterministe. Donc $\mathcal{L}(\mathcal{A}, s) \cap L_i = \emptyset$.

1.3.2 Algorithme de décision

Dans cette section, nous décrivons et commentons notre algorithme de décision. Dans un premier temps, nous construisons l'automate produit $\mathcal{A}$ à partir du système $\mathcal{R}$ et des langages $L_1, \ldots, L_n$.

Exemple 263. *Soient la signature* $\Sigma = \{f/2, a/0\}$, *le système de réécriture* $\mathcal{R}$ *constitué de la seule règle* $r := f(\alpha, \beta) \to f(a, \alpha)$ *avec* α *et* β *variables et les langages* L_1 *et* L_2 *définis par :*

- $L_1 = \{t \in T(\Sigma) \mid t \not\xrightarrow{\mathcal{I}} t\} = \{a\}$, *et*
- L_2 *l'ensemble des termes binaires équilibrés sur* Σ *(exemple 131).*

Alors l'automate produit $\mathcal{A}$ *peut être défini par* $\mathcal{A} = (\Sigma, \{s_a, s_f, s_p\}, \{s_a, s_f, s_p\}, \Delta)$ *avec* Δ *constitué des règles suivantes :*

$$\begin{array}{rcll} a & \to & s_a(a) & f(s_f(x_1), s_f(x_2)) \xrightarrow{[1=2]} s_f(f(x_1, x_2)) \\ f(s_a(x_1), s_a(x_2)) & \to & s_f(f(x_1, x_2)) & f(s_f(x_1), s_f(x_2)) \xrightarrow{[1\neq 2]} s_p(f(x_1, x_2)) \\ f(s(x_1), s'(x_2)) & \to & s_p(f(x_1, x_2)) & \text{pour tout } (s, s') \notin \{(s_a, s_a), (s_f, s_f)\} \end{array}$$

Nous avons alors $\mathcal{L}(\mathcal{A}, s_a) = L_1$ *et* $\mathcal{L}(\mathcal{A}, s_a) \cup \mathcal{L}(\mathcal{A}, s_f) = L_2$.

Les formules que nous considérons à partir de maintenant sont des conjonctions de contraintes $x \xrightarrow{r} x$, $x \xrightarrow{r} y$, $x \not\xrightarrow{r} y$ (x et y variables distinctes), $x \in \mathcal{L}(\mathcal{A}, s)$ et $x \neq y$. Nous appelons *problèmes* de telles formules. L'un des paramètres de notre algorithme est un type particulier de problèmes, appelés problèmes d'entrée. Un *problème d'entrée* P' est un problème tel que :

- Pour toute variable x de P', il existe un unique état s de $\mathcal{A}$ tel que la contrainte $x \in \mathcal{L}(\mathcal{A}, s)$ apparaît dans P', et
- Pour tout couple de variables distinctes x et y de P', la contrainte $x \neq y$ apparaît dans P'.

Pour obtenir un problème d'entrée P' à partir d'une formule P du fragment existentiel de la théorie du premier ordre de la réécriture en un pas étendue, nous utilisons les transformations non déterministes suivantes :

1. Pour toute contrainte $x \not\xrightarrow{r} x$ de P, nous choisissons s tel que $\mathcal{L}(\mathcal{A}, s) \subseteq L_r$ et nous remplaçons la contrainte $x \not\xrightarrow{r} x$ par la contrainte d'appartenance $x \in \mathcal{L}(\mathcal{A}, s)$;
2. Pour tout couple de variables distinctes x, y tel que $x \neq y$ n'est pas dans P, soit nous ajoutons $x \neq y$, soit nous substituons x par y;
3. Pour toute contrainte d'appartenance $x \notin L_i$, nous choisissons un état s tel que $\mathcal{L}(\mathcal{A}, s) \cap L_i = \emptyset$ et nous remplaçons la contrainte $x \notin L_i$ par la contrainte d'appartenance $x \in \mathcal{L}(\mathcal{A}, s)$;
4. Pour toute contrainte d'appartenance $x \in L_i$, nous choisissons un état s tel que $\mathcal{L}(\mathcal{A}, s) \subseteq L_i$ et nous remplaçons la contrainte $x \in L_i$ par la contrainte d'appartenance $x \in \mathcal{L}(\mathcal{A}, s)$;
5. Pour toute variable x de P n'apparaissant pas dans une contrainte d'appartenance, nous choisissons un état s et ajoutons la contrainte d'appartenance $x \in \mathcal{L}(\mathcal{A}, s)$.

Puisque l'ensemble des états est fini, il est évident qu'il existe un nombre fini de choix pour chacune de ces transformations. De plus P est satisfiable si et seulement s'il existe un problème P' obtenu par les transformations précédentes satisfiable.

Quand L_r est non vide, il existe des états $s_{r_1}, \ldots, s_{r_k}$ de l'automate produit $\mathcal{A}$ tels que $\bigcup_{i \in [k]} \mathcal{L}(\mathcal{A}, s_{r_k}) = L_r$. Dans la première transformation, nous sélectionnons l'un d'eux. Si L_r est vide, il n'existe pas de tel état et le problème est insatisfiable. Dans la seconde transformation, nous devinons pour tout couple de variables si elles sont égales ou non. Les trois transformations suivantes sont similaires à la première.

Exemple 264. *(suite de l'exemple 263).*

Soit le problème P suivant :

$$x \xrightarrow{r} y \wedge y \not\xrightarrow{r} z \wedge x' \in L_2.$$

Alors un problème d'entrée possible obtenu à partir de P en posant $y = z$ est :

$$P_0 := x \xrightarrow{r} y \wedge y \in \mathcal{L}(\mathcal{A}, s_a) \wedge x \neq y \wedge x' \in \mathcal{L}(\mathcal{A}, s_f) \wedge x' \neq x \wedge x' \neq y.$$

L'algorithme de décision est détaillé en page 180. Nous en donnons une présentation récursive et non déterministe. L'algorithme prend en entrée deux paramètres : (i) Un problème P satisfaisant les propriétés décrites ci-dessus; (ii) Une liste de problèmes initialement vide. Il transforme de façon non déterministe les problèmes suivant un ensemble de règles d'inférence. Les transformations modifient essentiellement la partie réécriture des problèmes, c'est-à-dire les formules atomiques dont les variables apparaissent dans une contrainte de réécriture et non réécriture. Quand aucun échec est détecté, la terminaison est basée soit sur une absence de partie réécriture, soit sur une relation $\preceq$ sur les problèmes évitant les calculs infinis.

Avant d'expliquer notre algorithme, nous considérons les notations suivantes utilisées dans celui-ci.

Définition 265. *Soit P un problème.*

- *La* partie réécriture de P *est* $P^R = P^{R,R} \wedge P^{R,\in} \wedge P^{R,\neq}$ *avec :*
  - $P^{R,R}$ *la conjonction des formules atomiques* $x \xrightarrow{r} y$, $x \not\xrightarrow{r} y$ *et* $x \xrightarrow{r} x$ *apparaissant dans* P,
  - $P^{R,\in}$ *la conjonction des formules atomiques* $x \in \mathcal{L}(\mathcal{A}, s)$ *apparaissant dans* P *avec* x *variable de* $P^{R,R}$, *et*
  - $P^{R,\neq}$ *la conjonction des formules atomiques* $x \neq y$ *apparaissant dans* P *avec* x *et* y *variables de* $P^{R,R}$.
- *La* partie appartenance de P, *notée* $P^{\in}$, *est la conjonction des formules atomiques* $x \in \mathcal{L}(\mathcal{A}, s)$ *apparaissant dans* P *avec* x *variable n'appartenant pas à* $P^{R,R}$.
- *La* partie différence de P, *notée* $P^{\neq}$, *est la conjonction des formules atomiques* $x \neq y$ *apparaissant dans* P *avec au moins une des variables* x, y *n'appartenant pas à* $P^{R,R}$.

Dans l'algorithme, un échec peut subvenir quand il y a trop de différences par rapport aux contraintes d'appartenance.

Définition 266. *Soit P un problème existentiel. Alors*

$$D_P(s) = card(\{x \in Var(P) \mid \textit{la contrainte } x \in \mathcal{L}(\mathcal{A}, s) \textit{ est dans } P\}).$$

Intuitivement, $D_P(s)$ est le nombre minimal de termes clos différents qui doivent appartenir à $\mathcal{L}(\mathcal{A}, s)$ si nous voulons satisfaire la contrainte $x \in \mathcal{L}(\mathcal{A}, s)$.

Exemple 267. *(suite de l'exemple 264).*
Pour le problème d'entrée P_0, nous avons :

$$\begin{array}{lcl} P_0^R & := & x \xrightarrow{r} y \wedge y \in \mathcal{L}(\mathcal{A}, s_a) \wedge x \neq y \\ P_0^{\in} & := & x' \in \mathcal{L}(\mathcal{A}, s_f) \\ P_0^{\neq} & := & x' \neq x \wedge x' \neq y \end{array} \qquad \begin{array}{lcl} D_{P_0}(s_a) & = & 1 \\ D_{P_0}(s_f) & = & 1 \end{array}$$

Enfin la notation $Q \preceq P$ (définition 268) exprime qu'un problème Q est moins contraint qu'un problème P. Donc nous pouvons dire que grossièrement les transformations qui peuvent être faite sur P peuvent l'être sur Q.

Tout d'abord nous notons $Q \subseteq Q'$ pour deux problèmes Q et Q' s'il existe P' tel que Q' soit syntaxiquement égal à $Q \wedge P'$.

Définition 268. *Soient P et Q deux problèmes. Nous notons $Q \preceq P$ si et seulement s'il existe une application injective ρ de $Var(Q^R)$ dans $Var(P^R)$ telle que :*

- $\rho(Q^{R,R}) = P^{R,R}$ *et*
- $\rho(Q^{R,\in}) \subseteq P^{R,\in}$.

Nous disons que $Q \prec P$ si en plus $\rho(Q^{R,\in}) \neq P^{R,\in}$.

Dans la suite, nous appelons l'application ρ un *renommage des variables de réécriture.*

Les règles d'inférence ($\mathcal{S}_{equal}$, $\mathcal{S}_{diff}$, $\mathcal{S}_{gen}$, $\mathcal{S}_{apply}$, $\mathcal{S}_{member}$) sont données pour des systèmes de réécriture ultra-minces (voir page 187). Elles peuvent facilement être adaptées à n'importe quel système, mais la terminaison est uniquement montrée pour des systèmes ultra-minces.

Afin d'expliquer notre algorithme, nous distinguons les tests d'arrêt et le reste de la procédure appelé partie inférence.

Partie inférence

Nous nous concentrons sur les lignes 12 à 23 pour étudier le cœur de l'algorithme 1. Dans les lignes 12-13, un symbole de tête est deviné pour chaque variable de P^R, et de nouvelles variables sont introduites. Pour toute paire x, y de ces nouvelles variables, nous devinons si $x = y$ ou $x \neq y$ (lignes 14 à 18). Puis nous procédons de sorte que les prédicats $\xrightarrow{r}$, $\not\xrightarrow{r}$, $\in$ et $\neq$ n'agissent que sur les nouvelles variables :

- Considérant les formules atomiques $f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m)$, les différences doivent être effacées (dans le cas où $f \neq g$) ou propagées aux nouvelles variables $x_1, \ldots, x_n$ et $y_1, \ldots, y_m$. Ceci est réalisé ligne 19 avec le système $\mathcal{S}_{diff}$.

- Ligne 20, un traitement similaire est fait pour les contraintes d'appartenance. Il consiste tout simplement à suivre les règles de l'automate produit $\mathcal{A}$ (système $\mathcal{S}_{member}$).

- Ligne 21, les relations de réécriture et non réécriture sont propagées ou supprimées (systèmes $\mathcal{S}_{gen}$ et $\mathcal{S}_{apply}$). La notation $\xrightarrow{r}_\bullet$ est utilisée pour désigner une réécriture à la racine. Par exemple, nous notons $s \xrightarrow{r}_\bullet t$ si s se réécrit en t par application de la règle r à la racine de s et t.

Cette étape de l'algorithme introduit, bien sûr, de nouvelles égalités et différences entre variables puisque les règles de l'automate peuvent être non linéaires. Donc une simplification du problème (ligne 22) doit être faite. Puisqu'un choix a été fait ligne 15 concernant les égalités et différences entre nouvelles variables, la simplification consiste simplement en deux règles, un échec pour les différences et une substitution pour les égalités (système $\mathcal{S}_{equal}$).

Algorithme 1 : DecQuasiMince

1: DecQuasiMince(P, PP) : Booléen
   {P un problème, PP une liste de problèmes}
2: **Si** P contient $\bot$ **Alors**
3: Retourner FAUX
4: **Sinon Si** il existe un état $s \in \mathcal{S}$ tel que $D_P(s) > card(\mathcal{L}(\mathcal{A}, s))$ **Alors**
5: Retourner FAUX
6: **Sinon Si** P^R est vide **Alors**
7: Retourner VRAI
8: **Sinon Si** $Q \preceq P$ avec $Q \in PP$ **Alors**
9: Retourner FAUX
10: **Sinon**
11: Mettre P sur PP
   {Deviner un symbole de tête pour chaque variable}
12: Soient $f_1, \ldots, f_n$ symboles de Σ
13: Soit $P^{(0)}$ la formule $P^R\{x_1 \leftarrow f_1(x_1^1, \ldots, x_{p_1}^1), \ldots, x_n \leftarrow f_n(x_1^n, \ldots, x_{p_n}^n)\}$
   avec $\{x_1, \ldots, x_n\} = \mathcal{V}ar(P^R)$ et $x_1^1, \ldots, x_{p_1}^1, \ldots, x_1^n, \ldots, x_{p_n}^n$ nouvelles variables
   {Deviner les égalités et différences entre nouvelles variables}
14: Deviner une partition des nouvelles variables $\mathcal{V}ar(P^{(0)}) = \biguplus_i \pi_i$
15: Soit $P^{(1)}$ le problème obtenu en
16: 1. Prenant une variable x_{π_i} dans chaque classe π_i
17: 2. Substituant pour tout i, chaque x de π_i par x_{π_i}
18: 3. Ajoutant pour tout i, j tel que $i \neq j$, $x_{\pi_i} \neq x_{\pi_j}$
   {Propager ou effacer les différences}
19: Soit $P^{(2)}$ la clôture de $P^{(1)}$ par $\mathcal{S}_{diff}$
   {Propager ou effacer les contraintes d'appartenance}
20: Soit $P^{(3)}$ la clôture de $P^{(2)}$ par $\mathcal{S}_{member}$
   {Propager ou effacer les réécritures}
21: Soit $P^{(4)}$ la clôture de $P^{(3)}$ par $\mathcal{S}_{gen}$, $\mathcal{S}_{apply}$
   {Simplifier le problème par les égalités et différences}
22: Soit P' la clôture de $P^{(4)}$ par $\mathcal{S}_{equal}$
   {Appel récursif avec le nouveau problème}
23: Retourner DecQuasiMince(P', PP)
24: **Fin Si**

Tests d'arrêt

La terminaison est contrôlée aux lignes 2 à 9. Nous en donnons une présentation informelle suivant quatre cas.

1. Echec (ligne 2) : Le problème contient le symbole $\bot$. Ce symbole est déduit de contradictions simples, comme par exemple $x \neq x$ avec x variable (règle (16)). Les échecs peuvent aussi apparaître lorsqu'une relation de non réécriture ne peut être satisfaite (règle (11)).

   En fait, un échec est obtenu lorsque les choix qui ont été faits ne mènent pas à une solution.

2. Trop de différences (ligne 4) : La quantité $D_P(s)$ désigne le nombre de variables distinctes dont l'interprétation doit appartenir à $\mathcal{L}(\mathcal{A}, s)$. Par exemple, considérons le problème $x \neq y \wedge x \in L_r \wedge y \in L_r$ où la règle r est $a \to a$ et la signature Σ ne contient que deux constantes a et b. Le langage L_r est le singleton $\{b\}$ et le problème impose qu'il existe deux termes dans L_r.

3. Solution (ligne 6): Il n'y a plus de réécriture ni de non réécriture dans le problème. Rappelons que P^R contient les contraintes de réécriture et de non réécriture de P. Donc le problème obtenu est un problème simple d'unification et de disunification, mais avec des contraintes d'appartenance. Le test de la ligne 4 assure que la conjonction des contraintes d'appartenance et de différence est satisfiable.

4. Boucle (ligne 8): Le problème courant a moins de solutions ou est équivalent en terme de solutions à un problème calculé précédemment. Ceci est le test principal nécessaire à l'obtention de la terminaison. Deux problèmes P et Q sont équivalents en terme de solutions s'ils contiennent les mêmes contraintes de (non) réécriture et les mêmes contraintes d'appartenance à un renommage de variables près.

   Le test $Q \preceq P$ assure que P est plus (ou aussi) contraint qu'un autre problème Q rencontré précédemment.

1.3.3 Preuve de la décidabilité

Propriétés

Soit P une formule du fragment existentiel de la théorie du premier ordre de la réécriture en un pas étendue et P_0 l'un des problèmes d'entrée obtenus à partir de P par les transformations vues au début de la section 1.3.2. La fonction DecQuasiMince est appelée avec les paramètres P_0, $\emptyset$. Un calcul de DecQuasiMince conduit à la construction d'une suite de formules PP mémorisée dans le second paramètre. Nous désignons par $P_0, \ldots, P_k$ les éléments de PP après k passages dans DecQuasiMince. Puisque DecQuasiMince est non déterministe, nous considérons l'arbre $\mathbf{T}_{dec}$ de tous les calculs possibles étiqueté par des problèmes : le symbole de tête est le problème initial P_0 et une branche dont les étiquettes sont $P_0, \ldots, P_k$ représente un calcul de $(P_0, \emptyset)$ à (P_k, PP_k) où PP_k est la suite de formules $P_0, \ldots, P_k$.

Nous caractérisons tout d'abord les propriétés des formules préservées par l'algorithme.

Proposition 269. *Chaque formule P_i de T_{dec} contient le symbole $\bot$ ou est un problème d'entrée.*

Preuve : Soit P_i une formule de $\mathbf{T}_{dec}$. D'après DecQuasiMince, P_i est une conjonction de formules atomiques de la forme $\bot$, $x \xrightarrow{r} y$, $x \not\xrightarrow{r} y$, $x \xrightarrow{r} x$, $x \in \mathcal{L}(\mathcal{A}, s)$, $x \neq y$ où x, y sont des variables distinctes de $\mathcal{X}$, r une règle de $\mathcal{R}$ et s un état de $\mathcal{S}$ (lignes 13 à 22). Nous en déduisons que soit P_i contient le symbole $\bot$, soit P_i est un problème. Remarquons que si P_i contient le symbole $\bot$, il constitue une feuille de $\mathbf{T}_{dec}$.

Considérons que P_i ne contienne pas le symbole $\bot$. Montrons que pour toutes variables distinctes x et y de P_i, la contrainte $x \neq y$ apparaît dans P_i. Les variables apparaissant dans P_i sont uniquement des variables fraîchement introduites puisque DecQuasiMince propage ou élimine les contraintes d'appartenance, de différences et de réécriture afin que les prédicats $\xrightarrow{r}$, $\not\xrightarrow{r}$, $\in$ et $\neq$ agissent uniquement sur les nouvelles variables. De plus, nous devinons des différences et égalités pour tout couple de ces variables (lignes 14 à 18) et toutes les égalités entre variables sont éliminées par substitution (application du système $\mathcal{S}_{equal}$ ligne 22). Donc si x et y sont deux variables distinctes de P_i, $x \neq y$ est dans P_i.

Finalement le problème d'entrée initial P_0 satisfait la propriété : pour toute variable x de P_0, il existe un unique état s de $\mathcal{A}$ tel que la contrainte $x \in \mathcal{L}(\mathcal{A}, s)$ apparaisse dans P_0. Cette propriété est préservée par l'algorithme puisque les contraintes d'appartenance sont propagées ou effacées (application du système $\mathcal{S}_{member}$ ligne 20). Finalement P_i est un problème d'entrée.

□

Donc puisque toute formule P_i de $\mathbf{T}_{dec}$ est un problème, nous avons d'après la définition 265, $P_i = P_i^R \wedge P_i^{\in} \wedge P_i^{\neq}$.

Terminaison

Pour montrer la terminaison de notre algorithme, nous montrons le lemme suivant.

Lemme 270. *Soit $P_1, P_2, \ldots$ une suite infinie de problèmes tels que $P_1^{R,R} = P_2^{R,R} = \ldots$ à des renommages de variables de réécriture près. Alors il existe deux entiers i et j avec $i < j$ tels que $P_i \preceq P_j$.*

Preuve : Soit $P_1, P_2, \ldots$ une suite infinie de problèmes tels que $P_1^{R,R} = P_2^{R,R} = \ldots$ à des renommages de variables de réécriture près.

Soit k le nombre de variables distinctes apparaissant dans les $P_i^{R,R}$ pour tout entier i. Pour tout entier i, $P_i^{R,\in}$ contient au plus k variables distinctes puisque $\mathcal{V}ar(P_i^{R,\in}) \subseteq \mathcal{V}ar(P_i^{R,R})$. Donc le nombre d'états de $\mathcal{A}$ étant fini et le nombre de variables distinctes des parties $P_i^{R,\in}$ étant borné par k, le nombre de parties $P_i^{R,\in}$ distinctes est borné. Donc il existe deux entiers i et j avec $i < j$ tels que $P_i^{R,\in} \subseteq P_j^{R,\in}$ à un renommage de variables de réécriture près.

Or il existe un renommage des variables de réécriture ρ de $\mathcal{V}ar(P_i^R)$ dans $\mathcal{V}ar(P_j^R)$ tel que $\rho(P_i^{R,R}) = P_j^{R,R}$. Donc puisque $\mathcal{V}ar(P_i^{R,\in}) \subseteq \mathcal{V}ar(P_i^{R,R})$, $\mathcal{V}ar(P_j^{R,\in}) \subseteq \mathcal{V}ar(P_j^{R,R})$ et ρ est injective, nous avons $\rho(P_i^{R,\in}) \subseteq P_j^{R,\in}$. Finalement $P_i \preceq P_j$. □

Proposition 271. *DecQuasiMince termine.*

Preuve : La terminaison est une conséquence de la finitude de $\mathbf{T}_{dec}$.

Chaque nœud de $\mathbf{T}_{dec}$ à un nombre fini de fils. Soit un nœud étiqueté par un problème P, le nombre de ses fils est le nombre de choix différents qui peuvent être faits dans DecQuasiMince quand nous effectuons les opérations suivantes : Deviner un symbole de tête pour chaque

variable de P^R (ligne 12); Deviner les différences et égalités entre nouvelles variables (ligne 15); Propager ou effacer les différences entre variables non nouvelles (lignes 14 à 18); Propager ou effacer les contraintes d'appartenance (ligne 20); Propager ou effacer les réécritures (ligne 21). Puisque l'ensemble $\mathcal{V}ar(P^R)$, le système de réécriture et le nombre d'états de l'automate produit sont finis, chaque nœud a un nombre fini de fils.

Chaque branche est finie. Tout d'abord, nous montrons que, pour chaque branche $P_0, P_1, \ldots$ le nombre de contraintes de réécriture et le nombre de contraintes de non réécriture n'augmente pas.

Avec les règles des systèmes $\mathcal{S}_{diff}$, $\mathcal{S}_{member}$, $\mathcal{S}_{equal}$, la partie P^R du problème considérée n'est pas modifiée. Les règles de $\mathcal{S}_{gen}$ génèrent des réécritures et non réécritures à la racine (respectivement $\rightarrow_\bullet$ et $\not\rightarrow_\bullet$), et n'augmentent pas le nombre de contraintes où les symboles $\rightarrow$ et $\not\rightarrow$ interviennent. Le système $\mathcal{S}_{apply}$ efface toutes les contraintes où les symboles $\rightarrow_\bullet$ et $\not\rightarrow_\bullet$ interviennent. Donc pour tout entier i non nul, le nombre de contraintes de réécriture (respectivement de non réécriture) de P_i et inférieur ou égal au nombre de contraintes de réécriture (respectivement de non réécriture) de P_{i-1}.

Nous en déduisons que si nous supposons qu'il existe une branche infinie dans $\mathbf{T}_{dec}$, il existe sur cette branche un nombre infini de problèmes $P_1, P_2, \ldots$ tels que $P_1^{R,R} = P_2^{R,R} = \ldots$ à des renommages de variables près. Donc d'après le lemme 270, il existe deux entiers i et j avec $i < j$ tels que $P_i \preceq P_j$. D'après l'algorithme, DecQuasiMince retourne FAUX dans ce cas (ligne 8). Donc $\mathbf{T}_{dec}$ ne possède pas de branche infinie.

Finalement $\mathbf{T}_{dec}$ est fini d'où DecQuasiMince termine. □

Correction

Nous montrons que notre algorithme est correct (proposition 275). Tout d'abord, nous pouvons montrer la correction des règles d'inférence (lemme 272), c'est-à-dire que si la formule du dessous d'une règle est satisfiable dans la structure $\mathcal{T}$, alors la formule de dessus de cette règle est satisfiable dans $\mathcal{T}$.

Notons que puisque le symbole $\bot$ est engendré lorsque les choix qui ont été faits ne mènent pas à une solution, nous avons $\mathcal{T} \not\models P$ pour tout problème P contenant le symbole $\bot$.

Lemme 272. *Toutes les règles des systèmes $\mathcal{S}_{gen}$, $\mathcal{S}_{apply}$, $\mathcal{S}_{diff}$, $\mathcal{S}_{member}$, $\mathcal{S}_{equal}$ sont correctes.*

Le lemme suivant montre qu'un problème sans partie réécriture est satisfiable et que nous pouvons nous passer des contraintes qui ne sont pas dans la partie réécriture.

Lemme 273. *Pour tout problème P_i de $\mathbf{T}_{dec}$, $P_i^{\in} \wedge P_i^{\neq}$ est satisfiable. De plus, si P_i^R est satisfiable, P_i est satisfiable.*

Preuve : Soit P_i un problème de $\mathbf{T}_{dec}$. Pour tout état s de $\mathcal{A}$, il existe un terme clos t tel que t appartienne à $\mathcal{L}(\mathcal{A}, s)$ puisque $\mathcal{A}$ est normalisé-complet. De plus pour toute variable x de P_i, il existe au plus une contrainte de la forme $x \in \mathcal{L}(\mathcal{A}, s)$ apparaissant dans P_i (règle 20). Nous en déduisons que $P_i^{\in}$ est satisfiable. Or le test de la ligne 4 est évalué à FAUX, c'est-à-dire que les contraintes d'appartenance sont compatibles avec les différences du problème. Donc la conjonction $P_i^{\in} \wedge P_i^{\neq}$ est satisfiable.

Supposons maintenant que P_i^R est satisfiable. Il existe donc une valuation σ de $\mathcal{V}ar(P_i^R)$ dans $T(\Sigma)$ telle que $\mathcal{T}, \sigma \models P_i^R$. Si le problème P_i est insatisfiable, cela ne peut provenir que des contraintes d'appartenance puisque P_i^R et $P_i^{\in} \wedge P_i^{\neq}$ sont satisfiables. Or d'après le test

de la ligne 4, les contraintes d'appartenance de P_i sont compatibles avec les différences du problème. Donc il existe une extension σ' de σ aux variables de P_i telle que $\mathcal{T}, \sigma' \models P_i$. □

Lemme 274. *Pour tout problème P_{i+1} de $\mathbf{T}_{dec}$, si P_{i+1} est satisfiable, P_i^R est satisfiable.*

Preuve : Soit P_{i+1} un problème satisfiable de $\mathbf{T}_{dec}$. Nous pouvons construire une valuation σ de $\mathcal{V}ar(P_i^R)$ dans $T(\Sigma)$ à partir des choix faits aux lignes 12 à 14 dans l'appel DecQuasiMince (P_i, PP_i). Puisque les règles sont correctes (lemme 272), nous avons $\mathcal{T}, \sigma \models P_i^R$. D'où P_i^R est satisfiable. □

Proposition 275. *S'il existe un calcul de DecQuasiMince à partir du problème P_0 qui retourne* VRAI, *alors le problème P_0 est satisfiable.*

Preuve : S'il existe un calcul de DecQuasiMince à partir du problème P_0 qui retourne VRAI, alors il existe une branche $P_0, P_1, \ldots, P_k$ de $\mathbf{T}_{dec}$ telle que P_k^R est vide. Donc $P_k = P_k^{\in} \wedge P_k^{\neq}$. Puisque $P_k^R = \emptyset$, P_k est satisfiable d'après le lemme 273. Nous pouvons alors montrer que pour i de $[k]$, P_i est satisfiable par applications successives des lemmes 274 et 273. Finalement P_0 est satisfiable. □

Complétude

Nous montrons maintenant que notre algorithme est complet (proposition 279). Soit $\mathbf{T}'_{dec}$ l'arbre des calculs de DecQuasiMince obtenu sans faire le test de la ligne 8. Les lemmes 276 et 277 justifient que lorsque nous stoppons l'exploration d'une branche en utilisant le test de la ligne 8, la complétude de l'algorithme est préservée.

Lemme 276. *Soient P_1 et Q_1 tels que Q_1 soit un ancêtre de P_1 dans $\mathbf{T}'_{dec}$ (Q_1 est à une position inférieure à celle de P_1), et $Q_1 \preceq P_1$. Alors s'il existe une branche de P_1 à P_x telle que P_x ne contient pas $\bot$, il existe une branche de même longueur de Q_1 à Q_x telle que $Q_x \preceq P_x$ et Q_x ne contient pas $\bot$.*

Preuve : La preuve est faite par induction sur la longueur de la branche de P_1 à P_x. Notons $P_1, P_2, \ldots, P_x$ les problèmes sur la branche de P_1 à P_x. Nous montrons qu'il existe une branche $Q_1 Q_2 \ldots Q_x$ de $\mathbf{T}'_{dec}$ telle que pour tout entier i de $[x]$, Q_i ne contient pas $\bot$, $Q_i \preceq P_i$ et $D_{Q_i}(s) \leq card(\mathcal{L}(\mathcal{A}, s))$.

Base. Pour $x = 1$, nous avons par hypothèse $Q_1 \preceq P_1$. De plus puisque Q_1 est un ancêtre de P_1 dans $\mathbf{T}'_{dec}$, les tests des lignes 2 et 4 sont évalués à FAUX pour Q_1. Donc Q_1 ne contient pas $\bot$ et $D_{Q_1}(s) \leq card(\mathcal{L}(\mathcal{A}, s))$ pour tout état s.

Induction. Soit i un entier de $[x-1]$. Supposons que la propriété soit vraie pour tout entier $j \leq i$. Donc il existe une branche $Q_1 Q_2 \ldots Q_i$ de $\mathbf{T}'_{dec}$ telle que pour tout entier j de $[i]$, Q_j ne contient pas $\bot$, $Q_j \preceq P_j$ et $D_{Q_j}(s) \leq card(\mathcal{L}(\mathcal{A}, s))$ pour tout état s.

Il y a deux choses à montrer : (i) Les tests des lignes 2, 4 et 6 sont évalués à FAUX pour Q_i ; (ii) P_i, Q_i et P_{i+1} étant donnés, nous pouvons construire Q_{i+1} ne contenant pas $\bot$ tel que $Q_i \preceq P_i$ et $D_{Q_i}(s) \leq card(\mathcal{L}(\mathcal{A}, s))$ pour tout état s.

(i) Par hypothèse d'induction, Q_i ne contient pas $\bot$ et $D_{Q_i}(s) \leq card(\mathcal{L}(\mathcal{A}, s))$ pour tout état s donc les tests des lignes 2 et 4 sont évalués à FAUX pour Q_i.

De plus $Q_i \preceq P_i$ donc il existe un renommage de variables ρ de $Var(Q_i^R)$ dans $Var(P_i^R)$ tel que $\rho(Q_i^{R,R}) = P_i^{R,R}$ et $\rho(Q_i^{R,\in}) \subseteq P_i^{R,\in}$. Or P_i est un ancêtre de P_{i+1} donc le test ligne 6 est évalué à FAUX pour P_i. Donc $P_i^{R,R} \neq \emptyset$, d'où $Q_i^{R,R} \neq \emptyset$. Le test ligne 6 est donc évalué à FAUX pour Q_i.

(ii) Les choix faits aux lignes 12, 15, 20, 21 et 22 pour P_i^R peuvent être mimés par Q_i^R à travers ρ. On peut montrer que le problème Q_{i+1} ainsi obtenu satisfait : $\perp$ n'apparaît pas dans Q_{i+1}, $Q_{i+1} \preceq P_{i+1}$ et $D_{Q_{i+1}}(s) \leq card(\mathcal{L}(\mathcal{A}, s))$ pour tout état s.

Finalement $Q_1 Q_2 \ldots Q_{i+1}$ est de façon évidente une branche de $\mathbf{T}'_{dec}$ ce qui termine l'induction.

Nous appliquons maintenant ce résultat à $x - 1$. Donc il existe une branche de Q_1 à Q_x de même longueur que la branche de P_1 à P_x telle que $Q_x \preceq P_x$ et Q_x ne contient pas $\perp$. □

Lemme 277. *S'il existe une branche de $\mathbf{T}'_{dec}$ terminée par P tel que P^R soit vide, alors il existe une branche de $\mathbf{T}_{dec}$ terminée par Q tel que Q^R soit vide.*

Preuve : Supposons qu'il existe une branche de $\mathbf{T}'_{dec}$ terminée par P tel que P^R soit vide. Il existe donc une branche $P_0 P_1 \ldots P_k$ de $\mathbf{T}'_{dec}$ telle que $P_k = P$. S'il n'existe pas d'entiers i et j avec $i < j$ tels que $P_i \preceq P_j$ alors la branche est dans $\mathbf{T}_{dec}$. Sinon, des applications successives du lemme 276 permettent de construire une nouvelle branche $Q_0 Q_1 \ldots Q_l$ de $\mathbf{T}'_{dec}$ telle que $P_0 = Q_0$, $Q_l \preceq P_k$ et il n'existe pas d'entiers i et j avec $i < j$ tels que $Q_i \preceq Q_j$. Cette branche est alors dans $\mathbf{T}_{dec}$. De plus P_k^R est vide, donc Q_l^R est vide puisque $Q_l \preceq P_k$. □

Soient P un problème satisfiable et σ une valuation de $Var(P)$ dans $T(\Sigma)$ telle que $\mathcal{T}, \sigma \models P$. Alors σ est appelée solution de hauteur h de P si h est la hauteur maximum des termes de $\sigma(Var(P^R))$.

Nous montrons maintenant qu'à partir de tout nœud de $\mathbf{T}'_{dec}$ étiqueté par un problème ayant une solution de hauteur h non nulle, il existe un calcul de l'algorithme privé du test de la ligne 8 calculant un problème satisfiable.

Lemme 278. *Pour tout nœud η étiqueté par P dans $\mathbf{T}'_{dec}$, si P a une solution de hauteur h non nulle, alors il existe un fils de η dans $\mathbf{T}'_{dec}$ étiqueté par un problème P' ayant une solution de hauteur inférieure ou égale à h-1.*

Preuve : Soit un nœud η étiqueté par P dans $\mathbf{T}'_{dec}$ tel que P possède une solution de hauteur h non nulle. Considérons une solution σ de hauteur h du problème P.

Puisque P est satisfiable, les tests des lignes 2 à 6 sont évalués à FAUX pour P. Donc DecQuasiMince peut deviner les substitutions correspondant à la solution σ aux lignes 12 à 19. Puisque ces substitutions s'appliquent sur P^R, nous obtenons un nouveau problème $P^{(2)}$ dont la solution est de hauteur inférieure ou égale à h-1. Dans la suite, nous examinons notre système d'inférence et montrons qu'il est complet. Concentrons-nous d'abord sur les contraintes de réécriture, c'est-à-dire examinons les systèmes $\mathcal{S}_{gen}$ et $\mathcal{S}_{apply}$.

Supposons que $P^{(2)}$ contienne la contrainte $f(x_1, \ldots, x_n) \xrightarrow{r} g(y_1, \ldots, y_m)$. Cette contrainte est satisfiable puisque $P^{(2)}$ est satisfiable. Il existe deux cas : soit (i) la réécriture s'applique à la racine des deux termes $f(x_1, \ldots, x_n)$ et $g(y_1, \ldots, y_m)$, soit (ii) elle s'applique sous la racine. Dans le premier cas, la règle (1) s'applique ainsi que la règle (7). Dans le second cas, nous pouvons dire que tous les sous-termes de $f(x_1, \ldots, x_n)$ et $g(y_1, \ldots, y_m)$ sont deux à deux

égaux sauf pour un couple pour lequel la relation de réécriture est satisfaite, ceci correspond à la règle (2).

Supposons maintenant que $P^{(2)}$ contienne la contrainte $f(x_1, \ldots, x_n) \not\xrightarrow{\mathcal{I}} g(y_1, \ldots, y_m)$. Alors la contrainte est satisfaite à la racine. Si $f = g$, nous devons également dire qu'elle est aussi satisfaite pour toute position sous la racine. Par contre, si $f \neq g$, la satisfaction de la contrainte à la racine suffit (règle (3)).

A la racine. Une règle de réécriture ne peut s'appliquer à la racine, soit si la règle n'est pas de la bonne forme (règle (9)), soit si les contraintes de non linéarité de la règle ne sont pas satisfaites (règle (10)), soit si l'un des sous-termes de $f(x_1, \ldots, x_n)$ ou $g(y_1, \ldots, y_m)$ n'est pas le bon terme clos de la règle (règle (12)).

Sous la racine. Nous avons $f = g$. Afin d'éviter l'explosion du nombre de non réécriture, nous utilisons une astuce consistant à considérer les trois cas suivants :

1. Soit la contrainte est satisfaite à la racine parce qu'il existe deux positions pour lesquelles les sous-termes de $f(x_1, \ldots, x_n)$ et $f(y_1, \ldots, y_n)$ sont différents (règle (5));
2. Tous les sous-termes de $f(x_1, \ldots, x_n)$ et $f(y_1, \ldots, y_n)$ sont égaux deux à deux sauf en une position où la contrainte de non réécriture est propagée (règle (4));
3. Quand tous les sous-termes sont égaux, nous avons $x_1 = y_1, x_2 = y_2, \ldots, x_n = y_n$. Nous devons donc dire que $x_i \not\xrightarrow{\mathcal{I}} x_i$ pour tout i. Nous transformons ces contraintes en contraintes d'appartenance de la forme $x_i \in \mathcal{L}(\mathcal{A}, s_i)$ où s_i est un état tel que $\mathcal{L}(\mathcal{A}, s_i) \subseteq L_r$ (règle (6)).

La propagation des contraintes d'appartenance est également évidente puisqu'elles sont satisfiables et donc aucun échec ne peut apparaître (seule la règle (18) est appliquée). Pour la même raison, nous avons aussi $D_{P'}(s) > card(\mathcal{L}(\mathcal{A}, s))$ pour tout état s où P' est le problème construit ligne 22. Nous en déduisons que P' possède une solution de hauteur inférieure ou égale à h-1. Puisque P' est satisfiable, les tests des lignes 2 et 4 sont évalués à `FAUX` pour P'. De plus h-1 étant non nul, le test de la ligne 6 est évalué à `FAUX` pour P'. Donc P' est l'étiquette d'un fils de η dans $\mathbf{T}'_{dec}$. □

Proposition 279. *Si le problème P_0 est satisfiable, il existe un calcul de DecQuasiMince à partir de P_0 retournant* `VRAI`.

Preuve : Supposons que P^0 soit satisfiable. Nous montrons tout d'abord par induction sur la hauteur de la solution de P_0 qu'il existe une branche de $\mathbf{T}'_{dec}$ terminée pas P tel que P^R soit vide. Nous supposons que P_0^R est non vide car ce cas est trivial.

Base. La hauteur de la solution de P_0 est zéro, donc toutes les variables de P_0^R sont interprétées par des constantes. Examinons le comportement de DecQuasiMince. Puisque P_0 est satisfiable, les tests des lignes 2 à 6 sont évalués à `FAUX`. Donc on peut deviner les substitutions correspondant à la solution de P_0 aux lignes 12 à 19. En conséquence, toutes les relations de réécriture peuvent être effacées ligne 21, puisque toutes s'appliquent. Les relations de non réécriture ont aussi une solution de hauteur zéro, donc elles peuvent aussi être satisfaites et effacées par les systèmes $\mathcal{S}_{gen}$ et $\mathcal{S}_{apply}$. Pour les

contraintes d'appartenance, puisque les substitutions correspondant à la solution ont été choisies, aucun $\bot$ est calculé et la conjonction des relations d'appartenance et des différences est satisfiable. Donc nous obtenons ligne 22 un problème P' satisfiable dont la partie réécriture est vide.

Induction. Ce cas est résolu par applications répétées du lemme 278.

Par le lemme 277, il existe une branche de $\mathbf{T}_{dec}$ terminée par Q tel que Q^R est vide. Donc il existe un calcul de DecQuasiMince à partir de P_0 retournant VRAI. □

Finalement nous déduisons des propositions 275 et 279, la proposition 280 qui termine la preuve du théorème 258.

Proposition 280. *DecQuasiMince est correct et complet: le problème P_0 est satisfiable si et seulement s'il existe un calcul de DecQuasiMince à partir de P_0 retournant* VRAI.

1.3.4 Extensions et remarques

Complexité Il est facile de montrer qu'avec l'ordre $\preceq$ que nous considérons, chaque branche de $\mathbf{T}_{dec}$ a une borne exponentielle et donc que notre algorithme est NEXPTIME.

Extensions Les restrictions sur les systèmes de réécriture impliquent que les contraintes d'unification de la forme $x = t$ où t est un terme de $T(\Sigma, \mathcal{X})$ apparaissent au cours des calculs. Cette propriété montre que la satisfiabilité dépend uniquement des contraintes de réécriture et d'appartenance. Quand nous diminuons les restrictions des systèmes, cette propriété est perdue, et donc nous ne pouvons prouver la terminaison de notre algorithme. Notons que dans ce cas, nous pouvons concevoir un semi-algorithme en modifiant légèrement le système d'inférence.

Par exemple, considérons le problème $P := \{x = h(x_1, x_2)\ ;\ x \xrightarrow{r} x_2\ ;\ x_1 \xrightarrow{r'} y\}$ avec $r := f(\alpha_1, \alpha_2) \to \alpha_2$ et $r' := f(\alpha_1, \alpha_2) \to g(\alpha_1, \alpha_2)$. A cause des égalités introduites entre termes de différentes profondeurs, nous pouvons générer une suite de problèmes dont la taille augmente selon notre ordre. Nous n'avons trouvé ni de critère, ni de stratégie pour limiter de telles expansions tout en préservant à la fois la complétude et la terminaison.

Il semble que la difficulté survienne dès qu'une règle effaçante est autorisée, même dans le cas de contraintes positives. Pour une meilleure compréhension du problème, une première approche serait d'étudier le cas où toutes les variables apparaissent à la même profondeur dans les règles de réécriture.

Connexions avec l'unification contextuelle Nous avons vu dans la section 1.1 que l'unification contextuelle et la réécriture sont fortement liées. Cette connexion suggère l'étude des techniques appliquées à l'unification contextuelle dans [77, 78] afin de les reformuler dans notre cadre.

1.3.5 Système de règles

Une réécriture peut s'appliquer à la racine...

$$\frac{f(x_1,\ldots,x_n) \xrightarrow{r} g(y_1,\ldots,y_m) \wedge P}{f(x_1,\ldots,x_n) \xrightarrow{r}_{\bullet} g(y_1,\ldots,y_m) \wedge P} \quad (1)$$

...ou sous la racine (propagation). Nous choisissons un sous-terme où la réécriture s'applique et nous disons que les autres sous-termes sont égaux.

$$\frac{f(x_1,\ldots,x_n) \xrightarrow{r} f(y_1,\ldots,y_n) \wedge P}{x_i \xrightarrow{r} y_i \bigwedge_{j\neq i} x_j = y_j \wedge P} \quad (2)$$

Maintenant pour les relations de non réécriture.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r} g(y_1,\ldots,y_m) \wedge P}{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} g(y_1,\ldots,y_m) \wedge P} \quad (3)$$

▶ Où $f \neq g$.

Il y a trois cas pour les relations de non réécriture appliquées sous la racine. Dans tous les cas, la relation de non réécriture doit être satisfaite à la racine. Premier cas : une seule position où les sous-termes sont différents.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r} f(y_1,\ldots,y_n) \wedge P}{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} f(y_1,\ldots,y_n) \wedge x_i \not\xrightarrow{r} y_i \wedge x_i \neq y_i \bigwedge_{j\neq i} x_j = y_j \wedge P} \quad (4)$$

Second cas : élimination. Il existe deux positions où les sous-termes sont différents.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r} f(y_1,\ldots,y_n) \wedge P}{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} f(y_1,\ldots,y_n) \wedge x_l \neq y_l \wedge x_m \neq y_m \wedge P} \quad (5)$$

▶ Où $l \neq m$.

Troisième cas : appartenance. Quand tous les sous-termes à la même position sont égaux, la relation de non réécriture doit être propagée à chaque position fils de la racine. Mais nous utilisons des contraintes d'appartenance pour dire qu'un terme ne peut se réécrire en lui-même.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r} f(y_1,\ldots,y_n) \wedge P}{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} f(y_1,\ldots,y_n) \bigwedge_i (x_i = y_i \wedge x_i \in \mathcal{L}(\mathcal{A}, s_i)) \wedge P} \quad (6)$$

▶ Où $s_1,\ldots,s_n$ sont des états tels que $f(s_1,\ldots s_n) \to s \in \Delta$ avec $\mathcal{L}(\mathcal{A}, s) \subseteq L_r$.

Système de règles 1: Le système $\mathcal{S}_{gen}$ exprime qu'une relation de (non) réécriture doit être satisfaite soit à la racine, soit sous la racine. Ce système est correct pour tout système de réécriture.

Application d'une relation de réécriture.

$$\frac{f(x_1^1,\ldots,x_n^1) \xrightarrow{r}_{\bullet} g(x_1^2,\ldots,x_m^2) \wedge P}{\bigwedge_{(j,k,l,p)\in I} (x_l^j = x_p^k) \wedge \bigwedge_{(j,k)\in J} (x_l^j \in \mathcal{L}(\mathcal{A}, s_{\alpha_l^j})) \wedge P} \quad (7)$$

▶ Où r est de la forme $f(\alpha_1^1,\ldots,\alpha_n^1) \to g(\alpha_1^2,\ldots,\alpha_m^2)$, $I = \{(j,k,l,p) \mid \alpha_l^j = \alpha_m^k, \{j,k\} \subseteq \{1,2\}, (j,l) \neq (k,p)\}$, $J = \{(j,l) \mid \alpha_l^j \in T(\Sigma)\}$ et $s_{\alpha_l^j}$ est tel que $\mathcal{L}(\mathcal{A}, s_{\alpha_l^j}) = \{\alpha_l^j\}$.

Une relation de réécriture échoue.

$$\frac{f(x_1^1,\ldots,x_n^1) \xrightarrow{r}_{\bullet} g(x_1^2,\ldots,x_m^2) \wedge P}{\bot} \quad (8)$$

▶ Où r est de la forme $f'(\alpha_1^1,\ldots,\alpha_n^1) \to g'(\alpha_1^2,\ldots,\alpha_m^2)$, et $f' \neq f$ ou $g' \neq g$.

Pour les relations de non réécriture. Premier cas : la règle n'a pas la bonne structure.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} g(y_1,\ldots,y_m) \wedge P}{P} \quad (9)$$

▶ Où $r \equiv \alpha \to \beta$ et $tête(\alpha) \neq f$ ou $tête(\beta) \neq g$.

Second cas : une égalité n'est pas satisfaite. Une contrainte d'égalité concerne des sous-termes en membre gauche, ou en membre droit, ou dans les deux.

$$\frac{f(x_1^1,\ldots,x_n^1) \not\xrightarrow{r}_{\bullet} g(x_1^2,\ldots,x_m^2) \wedge P}{x_l^j \neq x_p^k \wedge P} \quad (10)$$

▶ Où r est de la forme $f(\alpha_1^1,\ldots,\alpha_n^1) \to g(\alpha_1^2,\ldots,\alpha_m^2)$ avec $\alpha_l^j = \alpha_p^k$, $\{j,k\} \subseteq \{1,2\}$ et $(j,l) \neq (k,p)$.

Troisième cas : la règle a la bonne structure et il n'y a pas de contrainte d'égalité dans celle-ci.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} g(y_1,\ldots,y_m) \wedge P}{\bot} \quad (11)$$

▶ Où $r \equiv \alpha \to \beta$, $Var(\alpha) \cap Var(\beta) = \emptyset$, $tête(\alpha) = f$ et $tête(\beta) = g$.

Quatrième cas : les termes clos de la règle assurent la relation de non réécriture.

$$\frac{f(x_1,\ldots,x_n) \not\xrightarrow{r}_{\bullet} g(y_1,\ldots,y_m) \wedge P}{x_o^i \in \mathcal{L}(\mathcal{A}, s) \wedge P} \quad (12)$$

▶ Où r est de la forme $f(\alpha_1^1,\ldots,\alpha_n^1) \to g(\alpha_1^2,\ldots,\alpha_m^2)$, s est tel que $\mathcal{L}(\mathcal{A}, s) \cap \{\alpha_o^i\} = \emptyset$ et $(i,o) \in J = \{(i',o') \mid \alpha_{o'}^{i'} \in T(\Sigma)\}$.

Système de règles 2: Le système $\mathcal{S}_{apply}$ est constitué des règles appliquant les (non) réécritures. Il est correct uniquement pour les systèmes de réécriture ultra-minces.

Effacer les différences déjà résolues par le choix du symbole de tête.

$$\frac{f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m) \wedge P}{P} \quad (13)$$

▶ Où $f \neq g$.

Propager les différences. Nous pouvons omettre les différences entre variables "anciennes".

$$\frac{f(x_1, \ldots, x_n) \neq f(y_1, \ldots, y_n) \wedge P}{x_i \neq y_i \wedge P} \quad (14)$$

▶ Où f est d'arité strictement plus grande que zéro.

Différences non satisfaites.

$$\frac{a \neq a \wedge P}{\bot} \quad (15)$$

▶ Où a est une constante.

Système de règles 3: Système $\mathcal{S}_{diff}$. Propagation ou effacement des différences.

Echec.

$$\frac{x \neq x \wedge P}{\bot} \quad (16)$$

Substitution.

$$\frac{x = y \wedge P}{P[x/y]} \quad (17)$$

Système de règles 4: Système $\mathcal{S}_{equal}$. Traitement des égalités et différences.

Propager une contrainte d'appartenance.

$$\frac{f(x_1,\ldots,x_n) \in \mathcal{L}(\mathcal{A},s) \wedge P}{\bigwedge_i x_i \in \mathcal{L}(\mathcal{A},s_i) \bigwedge_{k \approx_{cl}} x_k = x_l \bigwedge_{k \not\approx_{cl}} x_k \neq x_l \wedge P} \quad (18)$$

▶ Où $f(s_1,\ldots s_n) \xrightarrow{[c]} s \in \Delta$.

Premier échec: pas de règle dans l'automate.

$$\frac{f(x_1,\ldots,x_n) \in \mathcal{L}(\mathcal{A},s) \wedge P}{\bot} \quad (19)$$

▶ S'il n'existe pas de règle $f(s_1,\ldots s_n) \xrightarrow{[c]} s \in \Delta$.

Deuxième échec: l'automate est déterministe.

$$\frac{f(x_1,\ldots,x_n) \in \mathcal{L}(\mathcal{A},s) \wedge f(x_1,\ldots,x_n) \in \mathcal{L}(\mathcal{A},s') \wedge P}{\bot} \quad (20)$$

▶ Si $s \neq s'$.

Système de règles 5: Système $\mathcal{S}_{member}$. Propagation ou effacement des contraintes d'appartenance.

Chapitre 2

Réécritures contrôlées

Dans la littérature [76, 24, 21], plusieurs études du langage $\mathcal{R}^*(\mathcal{L})$ où $\mathcal{R}$ est un système de réécriture et $\mathcal{L}$ un langage régulier ont été faites. Ce langage, image de $\mathcal{L}$ par application d'un nombre quelconque de pas de réécriture, n'est pas en général régulier même si $\mathcal{R}(\mathcal{L})$, l'image de $\mathcal{L}$ par application d'un pas de réécriture, l'est. Les constructions utilisées pour l'étude de ce langage nous semblaient similaires. Nous avons donc cherché une méthode d'étude générique. Pour un terme donné, nous marquons l'ensemble des positions où des réécritures s'appliquent et nous en déduisons sous certaines conditions la construction d'un automate d'arbres standard reconnaissant $\mathcal{R}^*(\mathcal{L})$.

Notre approche nous permet également de définir une méthode générale d'étude des questions de décision « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » où $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des langages réguliers et $\mathcal{R}el(\mathcal{L}_1)$ est l'ensemble image des termes de $\mathcal{L}_1$ par une relation $\mathcal{R}el$. Ces questions sont décidables lorsque la relation est un morphisme d'arbres inverse ou un morphisme d'arbres linéaire car alors $\mathcal{R}el(\mathcal{L}_1)$ est un langage régulier. Mais en général, le langage $\mathcal{R}el(\mathcal{L}_1)$ n'est pas forcément régulier, nous cherchons donc à transformer les questions précédentes en des questions équivalentes faisant intervenir des langages réguliers. Lorsque $\mathcal{R}el$ dépend d'un système de réécriture, nous marquons l'ensemble des positions où des réécritures s'appliquent et nous vérifions que la dérivation définie ainsi consiste en l'application de la relation $\mathcal{R}el$ à l'aide d'un langage dit de contrôle.

Nous étudions ces questions pour différentes relation $\mathcal{R}el$ dépendantes d'un système de réécriture. Nous considérons tout d'abord les cas où $\mathcal{R}el$ est l'application d'un pas de réécriture et l'application d'une réécriture parallèle, cette dernière opération consistant à appliquer en un pas de dérivation des réécritures s'appliquant sur des sous-termes à des positions incomparables. Ensuite nous considérons le cas des réécritures en une passe.

Pour qu'un terme t se réécrive en un terme u en une passe, il faut que toutes les occurrences des membres gauches de règles utilisées dans la dérivation de t en u apparaissent dans t sans chevauchement. De plus, à cause des non linéarités, il est indispensable de préciser si les sous-termes sont réécrits avant ou après avoir testé les égalités imposées par les règles. Il existe deux stratégies usuelles pour résoudre ce problème [28, 29, 30]. La première, appelée OI (Outermost–Innermost), consiste à réécrire le terme de la racine vers les feuilles et la seconde, appelée IO (Innermost–Outermost), consiste à réécrire le terme des feuilles vers la racine. Z. Fülöp *et al.* dans [33] introduisent deux nouvelles stratégies: la réécriture en une passe par la racine et la réécriture en une passe par les feuilles. Ces stratégies sont respectivement des cas particuliers de réécriture en une passe OI et en une passe IO dans lesquelles, une

réécriture concerne toujours les positions immédiatement adjacentes aux parties du terme réécrit précédemment. Z. Fülöp *et al.* dans [33] montrent que la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable pour les systèmes linéaires à gauche lorsque $\mathcal{R}el$ est l'une des relations suivantes : calcul des formes normales ou des formes sententielles selon la stratégie de réécriture en une passe par la racine ou par les feuilles.

Ce chapitre est organisé comme suit. Nous commençons par définir les différentes réécritures en un pas et en une passe citées ci-dessus (section 2.1). Puis, nous étudions le langage $\mathcal{R}^*(\mathcal{L})$ où $\mathcal{R}$ est un système de réécriture et $\mathcal{L}$ un langage régulier (section 2.2). Ensuite, nous définissons une méthode d'étude générale des questions de décision « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » où $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des langages réguliers et $\mathcal{R}el(\mathcal{L}_1)$ est l'ensemble image des termes de $\mathcal{L}_1$ par une relation $\mathcal{R}el$ (section 2.3). Finalement, nous étudions ces deux problèmes suivant différentes relations $\mathcal{R}el$: application d'un pas de réécriture ou d'une réécriture parallèle, ou encore calcul des formes normales ou des formes sententielles selon une passe IO, une passe OI, la stratégie de réécriture en une passe par la racine ou par les feuilles (section 2.4). Nous obtenons de bons résultats de décidabilité lorsque le langage de contrôle est régulier.

Nous considérons pour le reste de ce chapitre une signature Σ, un ensemble de variables $\mathcal{X}$, un système de réécriture $\mathcal{R}$ composé de n règles notées $\textbf{règle}_i \equiv l_i \to r_i$ et deux langages $\mathcal{L}_1$ et $\mathcal{L}_2$ réguliers.

2.1 Définitions

2.1.1 Réécriture en un pas

La *réécriture en un pas* consiste à appliquer une seule règle de réécriture. Donc si notons $\to$ la relation de réécriture définie par $\mathcal{R}$, l'ensemble image d'un langage $\mathcal{L}$ par application d'un pas de réécriture est :

$$\mathcal{R}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists u \in \mathcal{L},\ u \to t\}.$$

Exemple 281. *Soient la signature* $\Sigma = \{f/2, g/1, a/0, b/0\}$ *et le système* $\mathcal{R}$ *sur* Σ *constitué des règles suivantes :*

$$f(x,x) \to g(x) \qquad b \to a$$

Alors pour $\mathcal{L} = \{f(b,b)\}$, *nous avons* $\mathcal{R}(\mathcal{L}) = \{g(b),\ f(a,b),\ f(b,a)\}$.

2.1.2 Réécriture parallèle

La *réécriture parallèle* consiste à appliquer en une seule fois un ensemble de règles sur des positions parallèles.

Soient u et t deux termes de $T(\Sigma)$. Si nous notons $\Vdash$ la relation de réécriture parallèle, $t \Vdash u$ si et seulement s'il existe un entier k, un contexte C de $\mathcal{C}^{\Sigma}(k)$, une famille $(i_j)_{j\in[k]}$ de

$[n]$ (ensemble des indices des règles de réécriture) et une famille $(\sigma_{i_j})_{j\in[k]}$ de substitutions sur $T(\Sigma,\mathcal{X})$ tels que :

$$t = C[l_{i_1}\sigma_{j_1},\ldots,l_{i_k}\sigma_{j_k}] \quad \text{et} \quad u = C[r_{i_1}\sigma_{j_1},\ldots,r_{i_k}\sigma_{j_k}].$$

L'ensemble image d'un langage $\mathcal{L}$ par application d'une réécriture parallèle est :

$$\mathcal{R}_{||}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists u \in \mathcal{L},\ u \Vdash t\}.$$

Exemple 282. *Soient la même signature Σ et le même système $\mathcal{R}$ sur Σ que dans l'exemple précédent. Alors pour $\mathcal{L} = \{f(g(b),g(b))\}$, nous avons*

$$\mathcal{R}_{||}(\mathcal{L}) = \{f(g(b),g(b)),\ f(g(b),g(a)),\ f(g(a),g(b)),\ f(g(a),g(a)),\ g(g(b))\}.$$

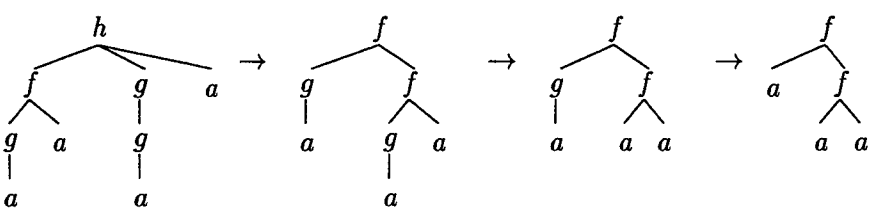
2.1.3 Réécriture en une passe

Dans le cas où le système $\mathcal{R}$ est linéaire, un terme t se réécrit en un terme u en une passe, si toutes les occurrences des règles utilisées dans la dérivation de t en u peuvent être appliquées en même temps (c'est-à-dire toutes les occurrences des membres gauches de règles apparaissent dans t sans chevauchement).

Exemple 283. *Soient la signature $\Sigma = \{h/3, f/2, g/1, a/0\}$ et le système $\mathcal{R}$ constitué des règles suivantes :*

$$h(x,g(y),z) \to f(y,x) \qquad g(x) \to a$$

Alors la chaîne de dérivation ci-dessous peut être réalisée en une passe.



Dans le cas général $\mathcal{R}$ n'est pas linéaire, nous devons alors préciser si les sous-termes sont réécrits avant ou après avoir testé les égalités imposées par les règles. Il existe deux stratégies usuelles pour résoudre ce problème [28, 29, 30] :

Passe OI. Cette stratégie consiste à réécrire le terme en une passe en allant de la racine (*Outermost*) vers les feuilles (*Innermost*).

Passe IO. Cette stratégie consiste à réécrire le terme en une passe en allant des feuilles vers la racine.

Si un terme t se réécrit en un terme u en une passe OI (respectivement IO), on note $t \to^{oi} u$ (respectivement $t \to^{io} u$).

Exemple 284. *Soient la signature* $\Sigma = \{f/2, g/2, a/0, b/0\}$ *et le système* $\mathcal{R}$ *constitué des règles suivantes :*

$$f(x,x) \to g(x,x) \qquad a \to b$$

Nous avons les chaînes de dérivation suivantes :

Nous avons donc $f(a,a) \to^{oi} g(a,b)$ *mais* $f(a,a) \not\to^{io} g(a,b)$, *et* $f(a,b) \to^{io} g(b,b)$ *mais* $f(a,b) \not\to^{oi} g(b,b)$.

Dans les deux sections suivantes, nous présentons deux nouvelles réécritures en une passe introduites par Z. Fülöp *et al.* dans [33] : la réécriture en une passe par la racine (*one-pass root-started rewriting*) et la réécriture en une passe par les feuilles (*one-pass leaf-started rewriting*). La première (respectivement seconde) est une réécriture en une passe OI (respectivement OI) dans laquelle une réécriture concerne toujours les positions immédiatement adjacentes aux parties du terme réécrites précédemment.

2.1.4 Réécriture en une passe par la racine

Dans la réécriture d'un terme t de $T(\Sigma)$ en une passe par la racine, la première partie de t réécrite contient sa racine. La réécriture continue ensuite en direction des feuilles de sorte que chaque pas de réécriture s'applique à la racine d'un sous-terme non encore réécrit dont la position est immédiatement adjacente à une partie du terme déjà réécrite.

Pour la définition formelle, un système de réécriture $\mathcal{R}_{\#}$, dans lequel un symbole spécial force ce mode de réécriture, est associé à $\mathcal{R}$.

Définition 285 (Z. Fülöp *et al.*, **[33]).** *Soit* $\#$ *un nouveau symbole de fonction unaire. Le système de réécriture en une passe par la racine associé à* $\mathcal{R}$ *est le système* $\mathcal{R}_{\#}$ *dont les règles sont obtenues en ajoutant un* $\#$ *à la racine du membre gauche et devant les variables du membre droit de toute règle de* $\mathcal{R}$. *Donc pour toute règle* $l(x_1, \ldots, x_p) \to r(x_1, \ldots, x_p)$ *de* $\mathcal{R}$, *la règle suivante appartient à* $\mathcal{R}_{\#}$:

$$\#(l(x_1, \ldots, x_p)) \to r(\#(x_1), \ldots, \#(x_p)).$$

Exemple 286. *Soient la signature* $\Sigma = \{f/2, g/1, c/0\}$ *et le système* $\mathcal{R}$ *constitué des règles suivantes :*

$$f(g(x_1), x_2) \to f(x_1, g(x_2)) \qquad g(x_1) \to g(c)$$

Alors le système $\mathcal{R}_{\#}$ *associé à* $\mathcal{R}$ *est constitué des règles :*

$$\#(f(g(x_1), x_2)) \to f(\#(x_1), g(\#(x_2))) \qquad \#(g(x_1)) \to g(c)$$

De façon évidente, le système $\mathcal{R}_{\#}$ est nœthérien. Afin de retrouver la chaîne de dérivation de $\mathcal{R}$ en une passe par la racine à partir des chaînes de dérivation de $\mathcal{R}_{\#}$, on introduit le morphisme d'arbre Ψ de $T(\Sigma \cup \{\#\})$ dans $T(\Sigma)$ effaçant uniquement les occurrences du symbole $\#$. Si

$$\#(t) \to_{\mathcal{R}_{\#}} t_1 \to_{\mathcal{R}_{\#}} t_2 \to_{\mathcal{R}_{\#}} \dots \to_{\mathcal{R}_{\#}} t_k$$

est une chaîne de dérivation de $\mathcal{R}_{\#}$ débutant sur un terme t de $T(\Sigma)$, alors

$$t \to_{\mathcal{R}} \Psi(t_1) \to_{\mathcal{R}} \Psi(t_2) \to_{\mathcal{R}} \dots \to_{\mathcal{R}} \Psi(t_k)$$

est une chaîne de dérivation de $\mathcal{R}$ en une passe par la racine. Si t_k est irréductible par $\mathcal{R}_{\#}$, $\Psi(t_k)$ est appelé une forme normale de t pour $\mathcal{R}$ selon la réécriture en une passe par la racine.

Par la suite, nous notons $\to^{rs}$ la relation de réécriture en une passe par la racine. Par exemple, on a $t \to^{rs} \Psi(t_i)$ pour tout entier i de $[k]$.

Exemple 287. *Nous considérons le système $\mathcal{R}$ de l'exemple 286. Alors*

$$f(g(g(c)), c) \to^{rs} f(g(c), g(c))$$

puisque

$$\#(f(g(g(c)), c)) \to_{\mathcal{R}_{\#}} f(\#(g(c)), g(\#(c))) \to_{\mathcal{R}_{\#}} f(g(c), g(\#(c))).$$

2.1.5 Réécriture en une passe par les feuilles

Dans la réécriture d'un terme t de $T(\Sigma)$ en une passe par les feuilles, la dérivation commence par ses feuilles pour remonter jusqu'à sa racine de sorte que toutes les variables d'un membre gauche de règle appliquée dans un pas de dérivation soient substituées par des parties du terme déjà réécrites.

Cette fois-ci, le système de réécriture $\mathcal{R}^{\#}$ utilisé pour définir ce mode de réécriture est construit en deux étapes.

Définition 288. *Soit $\mathcal{R}_e$ une extension du système $\mathcal{R}$ constitué des règles suivantes:*

$$l[y_1, \dots, y_p] \to r[y_1, \dots, y_p]$$

pour

- *toute règle $l \to r$ de $\mathcal{R}$ avec l et r termes de $T(\Sigma, \mathcal{X}_p)$,*
- *tout p-uplet $y_1, \dots, y_p$ de termes de $\mathcal{X} \cup \Sigma_0$ tel que les variables apparaissant dans l'ensemble $\{y_1, \dots, y_p\}$ soient deux à deux distinctes.*

Soient $\Sigma' = \{f' \mid f \in \Sigma\}$ une copie disjointe de Σ telle que pour tout symbole f de Σ, f et f' ont même arité, et $\#$ est un nouveau symbole de fonction unaire. Le système de réécriture en une passe par les feuilles associé à $\mathcal{R}$ est le système $\mathcal{R}^{\#}$ dont les règles sont obtenues en ajoutant un $\#$ devant les variables du membre gauche et à la racine du membre droit de toute règle de $\mathcal{R}_e$. Donc pour toute règle $l(x_1, \ldots, x_p) \to r(x_1, \ldots, x_p)$ de $\mathcal{R}_e$:

$$l(\#(x_1), \ldots, \#(x_p)) \to \#(r'(x_1, \ldots, x_p)) \in \mathcal{R}^{\#}$$

avec r' obtenu à partir de r en remplaçant tout symbole f de Σ par son symbole correspondant de Σ'.

Exemple 289. *Soient la signature $\Sigma = \{f/2, g/1, c/0\}$ et le système $\mathcal{R}$ constitué des règles suivantes :*

$$f(g(x_1), x_2) \to f(x_1, c) \qquad g(c) \to c$$

Alors le système $\mathcal{R}_e$ est constitué des règles :

$$\begin{array}{rcl@{\qquad}rcl} f(g(x_1), x_2) & \to & f(x_1, c) & f(g(c), x_2) & \to & f(c, c) \\ f(g(x_1), c) & \to & f(x_1, c) & f(g(c), c) & \to & f(c, c) \\ g(c) & \to & c & & & \end{array}$$

Soit $\Sigma' = \{f'/2, g'/1, c'/0\}$. Alors le système $\mathcal{R}^{\#}$ associé à $\mathcal{R}$ est constitué des règles :

$$\begin{array}{rcl@{\qquad}rcl} f(g(\#(x_1)), \#(x_2)) & \to & \#(f'(x_1, c')) & f(g(c), \#(x_2)) & \to & \#(f'(c', c')) \\ f(g(\#(x_1)), c) & \to & \#(f'(x_1, c')) & f(g(c), c) & \to & \#(f'(c', c')) \\ g(c) & \to & \#(c') & & & \end{array}$$

De façon évidente, le système $\mathcal{R}^{\#}$ est nœthérien et ne peut faire qu'une passe sur un terme donné puisque les membres gauches et les membres droits de ses règles n'ont en commun que le symbole $\#$. Afin de retrouver la chaîne de dérivation de $\mathcal{R}$ en une passe par les feuilles à partir des chaînes de dérivation de $\mathcal{R}^{\#}$, on introduit le morphisme d'arbre Ψ' de $T(\Sigma \cup \Sigma' \cup \{\#\})$ dans $T(\Sigma)$ effaçant les occurrences du symbole $\#$ et les primes des symboles f' de Σ'. Si

$$t \to_{\mathcal{R}^{\#}} t_1 \to_{\mathcal{R}_{\#}} t_2 \to_{\mathcal{R}^{\#}} \ldots \to_{\mathcal{R}^{\#}} t_k$$

est une chaîne de dérivation de $\mathcal{R}^{\#}$ débutant sur un terme t de $T(\Sigma)$, alors

$$t \to_{\mathcal{R}} \Psi'(t_1) \to_{\mathcal{R}} \Psi'(t_2) \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} \Psi'(t_k)$$

est une chaîne de dérivation de $\mathcal{R}$ en une passe par les feuilles. Si t_k est irréductible par $\mathcal{R}^{\#}$, $\Psi'(t_k)$ est appelé forme normale de t pour $\mathcal{R}$ selon la réécriture en une passe par les feuilles.

Par la suite, nous notons $\to^{ls}$ la relation de réécriture en un passe par les feuilles. Par exemple, on a $t \to^{ls} \Psi'(t_i)$ pour tout entier i de $[k]$.

Exemple 290. *Nous considérons le système $\mathcal{R}$ de l'exemple 289. Alors*

$$f(g(c), g(c)) \to^{ls} f(c, c)$$

puisque

$$f(g(c), g(c)) \to_{\mathcal{R}^\#} f(g(c), \#(c')) \to_{\mathcal{R}^\#} \#(f'(c', c')).$$

Le système $\mathcal{R}_e$ peut sembler redondant mais sans les nouvelles règles qu'il contient, certaines réécritures en une passe par les feuilles seraient oubliées. En particulier, si aucune règle de $\mathcal{R}$ ne contient de termes clos, aucune réécriture en une passe par les feuilles pourrait débuter puisque toutes les branches des membres gauches des règles de $\mathcal{R}^\#$ contiendraient le symbole $\#$.

2.1.6 Notations

Pour un langage $\mathcal{L}$ sur Σ, nous notons :

- $\mathsf{FS}_{\mathsf{oi}}(\mathcal{L})$ l'ensemble des formes sententielles de $\mathcal{L}$ pour $\mathcal{R}$ selon la réécriture en une passe OI :

$$\mathsf{FS}_{\mathsf{oi}}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists s \in \mathcal{L}, s \to^{oi} t\}.$$

- $\mathsf{FS}_{\mathsf{io}}(\mathcal{L})$ l'ensemble des formes sententielles de $\mathcal{L}$ pour $\mathcal{R}$ selon la réécriture en une passe IO :

$$\mathsf{FS}_{\mathsf{io}}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists s \in \mathcal{L}, s \to^{io} t\}.$$

- $\mathsf{FS}_{\downarrow}(\mathcal{L})$ l'ensemble des formes sententielles de $\mathcal{L}$ pour $\mathcal{R}$ selon la réécriture en une passe par la racine :

$$\mathsf{FS}_{\downarrow}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists s \in \mathcal{L}, s \to^{rs} t\}.$$

- $\mathsf{FN}_{\downarrow}(\mathcal{L})$ l'ensemble des formes normales de $\mathcal{L}$ pour $\mathcal{R}$ selon la réécriture en une passe par la racine.

- $\mathsf{FS}_{\uparrow}(\mathcal{L})$ l'ensemble des formes sententielles de $\mathcal{L}$ pour $\mathcal{R}$ selon la réécriture en une passe par les feuilles :

$$\mathsf{FS}_{\uparrow}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists s \in \mathcal{L}, s \to^{ls} t\}.$$

- $\mathsf{FN}_{\uparrow}(\mathcal{L})$ l'ensemble des formes normales de $\mathcal{L}$ pour $\mathcal{R}$ selon la réécriture en une passe par les feuilles.

Notons que pour tout terme t de $T(\Sigma)$, les ensembles précédents pour $\mathcal{L} = \{t\}$ sont finis donc réguliers. Par contre pour un langage $\mathcal{L}$ quelconque de $T(\Sigma)$, ces ensembles ne sont pas nécessairement réguliers même si $\mathcal{L}$ l'est.

2.2 Réécritures multiples

Nous nous intéressons dans cette section à l'étude du langage $\mathcal{R}^*(\mathcal{L})$ où $\mathcal{R}$ est un système de réécriture et $\mathcal{L}$ un langage régulier :

$$\mathcal{R}^*(\mathcal{L}) = \{t \mid \exists s \in \mathcal{L}, s \xrightarrow{*}_{\mathcal{R}} t\}.$$

Ce langage n'est pas en général régulier même si le langage $\mathcal{R}(\mathcal{L})$ l'est. Pourtant certains chercheurs ont montré que celui-ci est régulier pour certaines classes de systèmes de réécriture :

- M. Dauchet et S. Tison [24] pour tout système $\mathcal{R}$ clos, c'est-à-dire dont les membres gauches et droits de règles sont clos;
- K. Salomaa [76] pour les systèmes monadiques linéaires à droite, une règle $l \to r$ étant monadique si l est de hauteur supérieure ou égale à un, et r de hauteur au plus un;
- J.-L. Coquidé *et al.* [21] pour les systèmes semi-monadiques linéaires, une règle $l \to r$ étant semi-monadique si l est de hauteur supérieure ou égale à un, et r est soit une variable, soit un terme dont les variables sont à la profondeur un.

F. Jacquemard [47, 46] quant à lui montre que l'ensemble des termes clos ayant une forme sententielle en forme normale pour un système $\mathcal{R}$ croissant est régulier, une règle $l \to r$ étant croissante si elle est linéaire et si toutes les variables communes à l et r apparaissent dans l à profondeur zéro ou un. Les méthodes utilisées dans les preuves de ces résultats sont similaires puisqu'elles consistent à construire de proche en proche un automate reconnaissant le langage étudié à partir d'un automate reconnaissant $\mathcal{L}$.

Nous avons donc cherché à définir une méthode générale d'étude du langage $\mathcal{R}^*(\mathcal{L})$. L'idée principale de notre étude consiste à marquer les positions où une réécriture s'applique. Pour cela, nous associons à toute règle **règle**$_i$ du système $\mathcal{R}$ un symbole $\bullet_i$ n'appartenant pas à Σ dont l'arité est égale au nombre de variables distinctes de **règle**$_i$. Soit Λ l'ensemble de ces nouveaux symboles et la signature $\Gamma = \Sigma \cup \Lambda$. Soient Ψ_l et Ψ_r les morphismes d'arbres de $T(\Gamma)$ dans $T(\Sigma)$ déterminés par les applications ψ_l et ψ_r de Γ dans $T(\Sigma, \mathcal{X})$ définies par :

- Pour tout symbole f d'arité p de Σ, $\psi_r(f) = \psi_l(f) = f(x_1, \ldots, x_p)$, et
- Pour tout i de $[n]$, $\psi_l(\bullet_i) = l_i$ et $\psi_r(\bullet_i) = r_i$.

En fait, pour un terme t donné, on peut voir $\Psi_l^{-1}(t)$ comme l'ensemble des termes de $T(\Gamma)$ obtenus en remplaçant les occurrences de membres gauches de règles dans t par des symboles $\bullet_i$ de Λ. Nous montrons tout d'abord la proposition suivante :

Proposition 291. *Pour tout langage $\mathcal{L}'$ sur Σ,*

$$\mathcal{L}' \cup \mathcal{R}(\mathcal{L}') \subseteq \Psi_r(\Psi_l^{-1}(\mathcal{L}')) \subseteq \mathcal{R}^*(\mathcal{L}').$$

Preuve : Tout d'abord, nous avons de façon évidente $\mathcal{L}' \cup \mathcal{R}(\mathcal{L}') \subseteq \Psi_r(\Psi_l^{-1}(\mathcal{L}'))$. Nous montrons maintenant la propriété suivante :

$$\forall t \in T(\Sigma), \forall u \in \Psi_l^{-1}(t), \quad \Psi_r(u) \in \mathcal{R}^*(\{t\}) \tag{1}$$

par induction sur le nombre de marques de u (symboles de Λ). Soient t un terme de $T(\Sigma)$ et u un terme de $\Psi_l^{-1}(t)$.

Base. Le terme u ne contient aucune marque. Alors $\Psi_l(u) = \Psi_r(u) = u$. Or $\Psi_l(u) = t$ puisque u est un terme de $\Psi_l^{-1}(t)$. Donc $t = u$. Nous en déduisons que u appartient à $\mathcal{R}^*(\{t\})$.

Induction. Soit k un entier. Supposons que la propriété (1) soit vraie pour tout terme t' de $T(\Sigma)$ et tout terme u' de $\Psi_l^{-1}(t')$ possédant au plus k marques.

Supposons que u contienne $k+1$ marques. Nous distinguons deux cas suivant que le symbole de tête de u soit une marque ou non.

tête(u) **est un symbole de** Λ. Il existe un entier i de $[n]$ et p termes $u_1, \ldots, u_p$ de $T(\Gamma)$, avec p le nombre de variables distinctes de règle$_i$, tels que $u = \bullet_i(u_1, \ldots, u_p)$. Soit i un entier de $[p]$. Le terme u_i appartient à $\Psi_l^{-1}(\Psi_l(u_i))$ et possèdent au plus k marques. Donc par hypothèse d'induction, le terme $\Psi_r(u_i)$ appartient à $\mathcal{R}^*(\Psi_l(u_i))$. D'où

$$l_i[\Psi_l(u_1), \ldots, \Psi_l(u_p)] \xrightarrow{*}_{\mathcal{R}} r_i[\Psi_r(u_1), \ldots, \Psi_r(u_p)].$$

Or $\Psi_l(u) = t = l_i[\Psi_l(u_1), \ldots, \Psi_l(u_p)]$ et $\Psi_r(u) = r_i[\Psi_r(u_1), \ldots, \Psi_r(u_p)]$. Donc le terme $\Psi_r(u)$ appartient à $\mathcal{R}^*(\{t\})$.

tête(u) **est un symbole de** Σ. Alors il existe un entier p, un contexte C de $\mathcal{C}^p(\Sigma)$ et p termes $u_1, \ldots, u_p$ de $T(\Gamma)$ dont les symboles de tête sont une marque tels que $u = C[u_1, \ldots, u_p]$.
Soit i un entier de $[p]$. Le terme u_i appartient à $\Psi_l^{-1}(\Psi_l(u_i))$, contient au plus $k+1$ marques et possède pour symbole de tête une marque, donc par hypothèse d'induction ou par le point précédent, le terme $\Psi_r(u_i)$ appartient à $\mathcal{R}^*(\Psi_l(u_i))$. Donc

$$C[\Psi_l(u_1), \ldots, \Psi_l(u_p)] \xrightarrow{*}_{\mathcal{R}} C[\Psi_r(u_1), \ldots, \Psi_r(u_p)].$$

De plus, puisque le contexte C ne contient aucune marque, nous avons $\Psi_l(u) = t = C[\Psi_l(u_1), \ldots, \Psi_l(u_p)]$ et $\Psi_r(u) = C[\Psi_r(u_1), \ldots, \Psi_r(u_p)]$. Nous en déduisons que le terme $\Psi_r(u)$ appartient à $\mathcal{R}^*(\{t\})$.

Donc la propriété (1) est vraie et nous en déduisons que $\Psi_r(\Psi_l^{-1}(\mathcal{L}')) \subseteq \mathcal{R}^*(\mathcal{L}')$. □

Nous considérons maintenant la suite de langages $(\mathcal{L}_k)_{k\in\mathbb{N}}$ définie par $\mathcal{L}_0 = \mathcal{L}$ et pour tout entier k non nul, $\mathcal{L}_k = \Psi_r(\Psi_l^{-1}(\mathcal{L}_{k-1}))$. Nous déduisons de la proposition 291 que pour tout entier k,

$$\mathcal{L}_k \cup \mathcal{R}(\mathcal{L}_k) \subseteq \mathcal{L}_{k+1} \subseteq \mathcal{R}^*(\mathcal{L}_k).$$

Donc pour tout entier k, $\mathcal{L}_k \subseteq \mathcal{L}_{k+1} \subseteq \mathcal{R}^*(\mathcal{L}_k)$ d'où $\mathcal{L}_k \subseteq \mathcal{L}_{k+1} \subseteq \mathcal{R}^*(\mathcal{L}_0)$. Donc la suite $(\mathcal{L}_k)_{k\in\mathbb{N}}$ est croissante et bornée par $\mathcal{R}^*(\mathcal{L}_0)$ c'est-à-dire :

$$\mathcal{L}_0 \subseteq \mathcal{L}_1 \cdots \subseteq \mathcal{L}_k \subseteq \cdots \subseteq \mathcal{R}^*(\mathcal{L}_0).$$

Nous considérons maintenant le cas où Ψ_l est linéaire et Ψ_r est tel que pour tout symbole de fonction f de Γ :

- Soit $\psi_r(f)$ est une constante ou une variable,

- Soit $\psi_r(f)$ est un terme linéaire de hauteur un ne possédant pas de sous-terme clos, c'est-à-dire qu'il existe un symbole g de Σ d'arité n non nulle et $x_1, \ldots, x_n$ variables distinctes tels que $\psi_r(f) = g(x_1, \ldots, x_n)$.

Nous montrons qu'alors la suite $(\mathcal{L}_k)_{k\in\mathbb{N}}$ est stationnaire à partir d'un certain indice. Puis, nous en déduisons que $\mathcal{R}^*(\mathcal{L})$ est régulier. Nous montrons tout d'abord la proposition suivante.

Proposition 292. *Soit $\mathcal{L}'$ un langage régulier et $\mathcal{A}$ un automate d'arbres standard reconnaissant $\mathcal{L}'$. Alors il existe un AAS $\mathcal{D}$ dont l'ensemble d'états est inclus dans celui de $\mathcal{A}$ reconnaissant $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$.*

Remarque 293. *Les constructions que nous utilisons dans la preuve de la proposition 292 s'inspire des preuves de clôture par morphismes inverses et morphismes linéaires de la classe des réguliers (propriété 47) que l'on trouve dans les ouvrages traitant des automates d'arbres standards (par exemple [20]).*

Dans la preuve classique de clôture par morphismes inverses, l'automate de départ doit être déterministe. Mais cette restriction n'est indispensable que dans le cas où le morphisme n'est pas linéaire.

Remarque 294. *Afin de simplifier les constructions, les états des automates que nous considérons dans la preuve suivante sont des constantes, c'est-à-dire que les règles de transition sont de la forme $f(q_1, \ldots, q_n) \to q$ où f est un symbole d'arité n et $q_1, \ldots, q_n, q$ des états.*

Preuve : Soient $\mathcal{L}'$ un langage régulier et $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta_\mathcal{A})$ un *AAS* reconnaissant $\mathcal{L}'$. Tout d'abord $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$ est régulier par clôture par morphismes inverses et morphismes linéaires de la classe des réguliers (propriété 47).

Par le théorème 42, il existe un automate réduit $\mathcal{B} = (\Gamma, \mathcal{Q}', \mathcal{Q}'_f, \Delta_\mathcal{B})$ équivalent à $\mathcal{A}$. Remarquons que $\mathcal{Q}'$ est inclus dans $\mathcal{Q}$. Nous considérons l'automate $\mathcal{C} = (\Sigma, \mathcal{Q}', \mathcal{Q}'_f, \Delta_\mathcal{C})$ dont les règles sont définies par :

$$\forall f \in \Gamma_n, \forall q_1, \ldots, q_n, q \in \mathcal{Q},\ (\Psi_l(f)[q_1, \ldots, q_n] \xrightarrow{*}_\mathcal{B} q) \Rightarrow (\Psi_r(f)(q_1, \ldots, q_n) \to q \in \Delta_\mathcal{C}).$$

Tout d'abord, les règles de $\Delta_\mathcal{C}$ sont bien définies puisque pour tout symbole de Γ, $\Psi_r(f)$ est soit une constante, soit une variable, soit un terme de hauteur un ne possédant aucun sous-terme clos. Remarquons que $\mathcal{C}$ peut contenir des ϵ-transitions (cas où $\Psi_r(f)$ est une variable).

Nous allons montrer que $\mathcal{C}$ reconnaît $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$. Nous montrons tout d'abord la propriété suivante :

$$\forall u \in T(\Gamma),\ (\Psi_l(u) \xrightarrow{*}_\mathcal{B} q) \Rightarrow (\Psi_r(u) \xrightarrow{*}_\mathcal{C} q) \tag{2}$$

par induction sur la hauteur du terme u. Soit u un terme de $T(\Gamma)$.

Base. Le terme u est de hauteur zéro donc il existe une constante a de Γ telle que $t = a$.
Alors $\Psi_l(u)$ est un terme clos et $\Psi_r(u)$ une constante. Nous en déduisons que :

$$(\Psi_l(u) \xrightarrow{*}_\mathcal{B} q) \Rightarrow (\Psi_r(u) \to q \in \Delta_\mathcal{C}) \Rightarrow (\Psi_r(u) \xrightarrow{*}_\mathcal{C} q).$$

Donc la propriété (2) est vraie dans le cas de base.

Induction. Soit k un entier. Supposons que la propriété (2) soit vraie pour tout terme u' de $T(\Gamma)$ de hauteur au plus k.

Supposons que u soit de hauteur $k+1$. Alors il existe un symbole f de Γ d'arité n non nulle et $u_1, \ldots, u_n$ termes de $T(\Gamma)$ tels que $u = f(u_1, \ldots, u_n)$. Nous avons alors $\Psi_l(u) = \Psi_l(f)[\Psi_l(u_1), \ldots, \Psi_l(u_n)]$ et $\Psi_r(u) = \Psi_r(f)[\Psi_r(u_1), \ldots, \Psi_r(u_n)]$.

Supposons que $\Psi_l(u) \xrightarrow{*}_{\mathcal{B}} q$. Alors puisque Ψ_l est linéaire, chaque variable de Ψ_l n'apparaît qu'une fois dans Ψ_l. Donc il existe n états $q_1, \ldots, q_n$ de $\mathcal{Q}$ tels que :

$$\Psi_l(u) \xrightarrow{*}_{\mathcal{B}} \Psi_l(f)[q_1, \ldots, q_n] \xrightarrow{*}_{\mathcal{B}} q.$$

Nous en déduisons que la règle $\Psi_r(f)[q_1, \ldots, q_n] \to q$ appartient à $\Delta_{\mathcal{C}}$ et que pour tout entier i de $[n]$, $\Psi_l(u_i) \xrightarrow{*}_{\mathcal{B}} q_i$. Donc pour tout i de $[n]$, $\Psi_r(u_i) \xrightarrow{*}_{\mathcal{C}} q_i$ par hypothèse d'induction. Finalement :

$$\Psi_r(u) \xrightarrow{*}_{\mathcal{C}} \Psi_r(f)[q_1, \ldots, q_n] \to_{\mathcal{C}} q.$$

Donc la propriété (2) est vraie. Nous montrons maintenant que $\mathcal{L}(\mathcal{C}) = \Psi_r(\Psi_l^{-1}(\mathcal{L}'))$.

Soit t un terme de $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$. Alors il existe un terme u de $\Psi_l^{-1}(\mathcal{L}')$ tel que $\Psi_r(u) = t$ et un terme s de $\mathcal{L}'$ tel que $\Psi_l(u) = s$. Puisque s est un terme de $\mathcal{L}'$, il existe un état final q de $\mathcal{Q}'_f$ tel que $s \xrightarrow{*}_{\mathcal{B}} q$. Nous déduisons de la propriété (2) que $\Psi_r(u) \xrightarrow{*}_{\mathcal{C}} q$. Donc t est un terme de $\mathcal{L}(\mathcal{C})$ puisque q est un état final de $\mathcal{C}$. Nous en déduisons que $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$ est inclus dans $\mathcal{L}(\mathcal{C})$.

Soit t un terme de $\mathcal{L}(\mathcal{C})$. Alors il existe un état final q de $\mathcal{Q}'_f$ tel que $t \xrightarrow{*}_{\mathcal{C}} q$. Nous pouvons montrer que par définition des règles de $\mathcal{C}$ il existe un entier n, un contexte C de $\mathcal{C}^n(\Sigma)$ et n états $q_1, \ldots, q_n$ de $\mathcal{Q}'$ tels que :

- $C[q_1, \ldots, q_n] \xrightarrow{*}_{\mathcal{B}} q$, et
- Pour tous termes $s_1, \ldots, s_n$ de $T(\Sigma)$, t appartient à $\Psi_r(\Psi_l^{-1}(C)[s_1, \ldots, s_n])$.

De plus puisque $\mathcal{B}$ est un automate réduit, il existe pour tout entier i de $[n]$, un terme s_i de $T(\Sigma)$ tel que $s_i \xrightarrow{*}_{\mathcal{B}} q_i$. Donc si nous considérons le terme $s = C[s_1, \ldots, s_n]$, nous avons t appartient à $\Psi_r(\Psi_l^{-1}(s))$ et $s \xrightarrow{*}_{\mathcal{B}} q$. Nous en déduisons que t est un terme de $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$ puisque q est un état final de $\mathcal{B}$. Finalement $\mathcal{C}$ reconnaît $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$.

L'automate $\mathcal{C}$ est un automate avec ϵ-transitions mais par le théorème 36, il existe un automate $\mathcal{D} = (\Sigma, \mathcal{Q}', \mathcal{Q}'_f, \Delta_{\mathcal{D}})$ sans ϵ-transition équivalent à $\mathcal{C}$. Finalement $\mathcal{D}$ est un automate reconnaissant $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$ dont l'ensemble d'états $\mathcal{Q}'$ est inclus dans $\mathcal{Q}$. □

Nous donnons maintenant un exemple illustrant la proposition 292.

Exemple 295. *Soient la signature* $\Sigma = \{f/2, g/1, a/0\}$, *le système* $\mathcal{R}$ *constitué de la seule règle* $\textit{règle}_1 \equiv f(f(x,a),y) \to g(x)$ *et le langage* $\mathcal{L} = \{t_n \mid n \in \mathbb{N}\}$ *où :*

$$\begin{aligned} t_0 &= a \\ \forall n \in \mathbb{N},\ t_{n+1} &= f(t_n, a) \end{aligned}$$

Les morphismes Ψ_l et Φ_r sont définis par:

$$\begin{array}{lcl} \psi_l(a) & = & a \\ \psi_l(g) & = & g(x_1) \\ \psi_l(f) & = & f(x_1, x_2) \\ \psi_l(\bullet_1) & = & f(fx_1, a), x_2) \end{array} \qquad \begin{array}{lcl} \psi_r(a) & = & a \\ \psi_r(g) & = & g(x_1) \\ \psi_r(f) & = & f(x_1, x_2) \\ \psi_r(\bullet_1) & = & g(x_1) \end{array}$$

Et le langage $\mathcal{L}$ est reconnu par l'automate $\mathcal{A} = (\Sigma, \{q_1, q_2\}, \{q_1, q_2\}, \Delta_\mathcal{A})$ où $\Delta_\mathcal{A}$ est constitué des règles suivantes:

$$a \to q_1 \qquad f(q_1, q_1) \to q_2 \qquad f(q_2, q_1) \to q_2$$

L'automate $\mathcal{A}$ est réduit. Nous déduisons de la preuve de la proposition 292, l'automate $C = (\Sigma, \{q_1, q_2\}, \{q_1, q_2\}, \Delta_C)$ où Δ_C est constitué des règles suivantes:

$$\begin{array}{lll} a \to q_1 & f(q_1, q_1) \to q_2 & f(q_2, q_1) \to q_2 \\ & g(q_1) \to q_2 & g(q_2) \to q_2 \end{array}$$

Les deux dernières règles sont déduites des relations $\Psi_l(\bullet_1)[q_1, q_1] \xrightarrow{*}_\mathcal{A} q_2$ et $\Psi_l(\bullet_1)[q_2, q_1] \xrightarrow{*}_\mathcal{A} q_2$. L'automate C reconnaît $\Psi_r(\Psi_l^{-1}(\mathcal{L}'))$.

Nous déduisons de la proposition 292, que pour tout entier k, le langage $\mathcal{L}_k$ est reconnu par un automate d'arbres standard $\mathcal{A}_k$ tel que l'ensemble d'états de $\mathcal{A}_{k+1}$ est inclus dans l'ensemble d'états de $\mathcal{A}_k$. Donc puisque le nombre d'automates reconnaissant un langage différent est borné pour un nombre d'états donné, la suite $(\mathcal{L}_k)_{k\in\mathbb{N}}$ est stationnaire à partir d'un indice i. Donc pour tout entier k supérieur ou égal à i, nous avons $\mathcal{L}_k = \mathcal{L}_i$.

Or pour tout entier k, $\mathcal{R}^k(\mathcal{L}_0) \subseteq \mathcal{L}_k$ car $\mathcal{R}(\mathcal{L}_k) \subseteq \mathcal{L}_{k+1}$. Donc $\mathcal{R}^k(\mathcal{L}_0) \subseteq \mathcal{L}_i$ car la suite est croissante. Nous en déduisons que $\mathcal{R}^*(\mathcal{L}_0) \subseteq \mathcal{L}_i$. Finalement $\mathcal{L}_i = \mathcal{R}^*(\mathcal{L}_0) = \mathcal{R}^*(\mathcal{L})$ puisque $\mathcal{L}_i \subseteq \mathcal{R}^*(\mathcal{L}_0)$. Donc $\mathcal{R}^*(\mathcal{L})$ est régulier. Nous en déduisons la proposition suivante.

Proposition 296. *Soit $\mathcal{L}$ un langage régulier. Alors $\mathcal{R}^*(\mathcal{L})$ est régulier si $\mathcal{R}$ est linéaire et tout membre droit de règle de $\mathcal{R}$ est soit une constante, soit une variable, soit un terme de hauteur un ne possédant pas de sous-terme clos.*

Notre résultat est plus restrictif que les résultats de K. Salomaa [76] et J.-L. Coquidé *et al.* [21] puisque nous n'autorisons pas de terme clos à la profondeur un dans les membres droits des règles. En fait l'intérêt de notre méthode d'étude réside dans la manière de marquer l'applications des règles de réécriture (morphismes Ψ_l et Ψ_r).

Remarquons également que la linéarité à gauche du système de réécriture nous est indispensable (exemple 297).

Exemple 297. *Soient la signature $\Sigma = \{a/0, g/1, f/2\}$, le système de réécriture R contenant l'unique règle $f(x, x) \to g(x)$ et le langage:*

$$\mathcal{L} = \{t \in T(\Sigma) \mid t = f(f(\ldots f(a, t_1), \ldots, t_{n-1}), t_n) \text{ où } t_1, \ldots, t_n \in T(\{g, a\})\}.$$

Nous associons à l'unique règle de $\mathcal{R}$ le symbole $\bullet$. Alors les morphisme Ψ_l et Ψ_r sont déterminées par:

$$\begin{array}{lll} \psi_l(a) = a & & \psi_r(a) = a \\ \psi_l(g) = g(x_1) & & \psi_r(g) = g(x_1) \\ \psi_l(f) = f(x_1, x_2) & & \psi_r(f) = f(x_1, x_2) \\ \psi_l(\bullet) = f(x_1, x_1) & & \psi_r(\bullet) = g(x_1) \end{array}$$

Pour tout entier n, nous notons t_n le terme $f(f(\ldots f(a,a),\ldots,g^{n-1}(a)),g^n(a))$ de $\mathcal{L}$.

Soit n un entier. Le terme t_n se dérive en $g^{n+1}(a)$ en $n+1$ applications de $\mathcal{R}$ donc t_n appartient à $\mathcal{R}^*(\mathcal{L})$. Or pour obtenir $g^{n+1}(a)$ à partir de t_n, il faut exactement $n+1$ applications de $\Psi_r(\Psi_l^{-1})$, c'est-à-dire que t_n appartient à $\mathcal{L}_{k+1}$ mais pas à $\mathcal{L}_k$. Nous en déduisons que la suite $(\mathcal{L}_k)_{k\in\mathbb{N}}$ est infinie.

2.3 Problèmes de décision et méthode générale de résolution

Nous nous intéressons maintenant aux questions de décision « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » où $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des réguliers et $\mathcal{R}el(\mathcal{L}_1)$ est l'ensemble image des termes de $\mathcal{L}_1$ par une relation $\mathcal{R}el$.

Les relations $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$ et $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$ ne sont pas en général facilement manipulables car $\mathcal{R}el(\mathcal{L}_1)$ n'est pas toujours régulier. Dans cette section, nous présentons une méthode générale de réduction de ces relations en relations équivalentes ne faisant intervenir que des langages réguliers ceci afin d'obtenir un problème plus simple à résoudre. Nous rappelons tout d'abord la propriété ci-dessous.

Propriété 298. *Soient A et B deux langages sur Σ et Γ respectivement (Γ signature) et Ψ une application de $T(\Sigma)$ dans $T(\Gamma)$. Alors*

$$\Psi(A) \subseteq B \Leftrightarrow A \subseteq \Psi^{-1}(B).$$

Nous supposons que la relation $\mathcal{R}el$ est définie par une conjonction de réécritures et de non réécritures. Par exemple, si $\mathcal{R}el$ est le calcul des formes normales, on a pour tout langage $\mathcal{L}_1$, $\mathcal{R}el(\mathcal{L}_1)$ est l'ensemble des termes t de $T(\Sigma)$ tel qu'il existe un terme s de $\mathcal{L}_1$ satisfaisant $s \to^* t$ et il n'existe pas de terme u de $T(\Sigma)$ tel que $t \to u$. Alors un terme t de $\mathcal{R}el(\mathcal{L}_1)$ est l'image d'un terme s de $\mathcal{L}_1$ par $\mathcal{R}el$ si s se réécrit en t et si la relation est satisfaite. Nous utilisons cette propriété pour tenter de définir une expression différente pour $\mathcal{R}el(\mathcal{L}_1)$. L'idée générale est de marquer les positions où une réécriture s'applique puis de déterminer un langage $\mathcal{R}_c$, dit de contrôle, traduisant les propriétés que doivent vérifier les marques pour qu'on ait bien application de la relation $\mathcal{R}el$. Par exemple, si $\mathcal{R}el$ est l'application d'un pas de réécriture, le langage de contrôle doit vérifier que les termes n'ont qu'une seule marque.

Pour marquer l'application de réécritures, nous considérons l'alphabet Γ et les morphismes Ψ_l et Ψ_r définis sur le système de réécriture $\mathcal{R}$ (section 2.2). Les morphismes précédents ne dépendent pas de la relation $\mathcal{R}el$. Donc pour traduire l'application de $\mathcal{R}el$, nous cherchons à définir un langage de contrôle $\mathcal{R}_c$ sur Γ tel que $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ contrôle l'application de la relation $\mathcal{R}el$. Par exemple, si $\mathcal{R}el$ est la réécriture en un pas, alors $\mathcal{R}_c$ est l'ensemble des

termes de $T(\Gamma)$ ne contenant qu'un symbole $\bullet_i$. Pour être facilement utilisable, le langage $\mathcal{R}_c$ doit être régulier. Si on arrive à déterminer un tel langage, on cherche à vérifier que l'on a

$$\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c) \tag{3}$$

En effet si cette relation est vérifiée et que $\mathcal{R}_c$ est régulier, nous allons montrer que la résolution des questions de décision « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est simplifiée.

Nous donnons maintenant la méthode générale que nous allons utiliser dans la section suivante pour montrer lorsque cela est possible que la relation (3) est vérifiée.

Méthode de preuve de la relation $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$.

Nous montrons tout d'abord que $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c) \subseteq \mathcal{R}el(\mathcal{L}_1)$. Nous considérons un terme t de $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$. Il existe un terme u de $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ tel que $\Psi_r(u) = t$. Alors $\Psi_l(u) \in \mathcal{L}_1$ puisque u appartient à $\Psi_l^{-1}(\mathcal{L}_1)$.

Puisque u appartient à $\mathcal{R}_c$, nous montrons que $\Psi_l(u)$ se dérive en $\Psi_r(u)$ par application de la relation $\mathcal{R}el$, c'est-à-dire que $\Psi_r(u)$ appartient à $\mathcal{R}el(\Psi_l(u))$. Nous en déduisons que $\Psi_r(u)$ appartient à $\mathcal{R}el(\mathcal{L}_1)$ puisque $\Psi_l(u)$ appartient à $\mathcal{L}_1$. Et finalement $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c) \subseteq \mathcal{R}el(\mathcal{L}_1)$ puisque $\Psi_r(u) = t$.

Ensuite nous montrons que $\mathcal{R}el(\mathcal{L}_1) \subseteq \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$. Nous considérons un terme t de $\mathcal{R}el(\mathcal{L}_1)$. Alors il existe un terme s de $\mathcal{L}_1$ tel que s se dérive en t par application de la relation $\mathcal{R}el$. Si $\mathcal{R}_c$ a été bien choisi, nous pouvons montrer qu'il existe un terme u de $\mathcal{R}_c$ tel que $s = \Psi_l(u)$ et $t = \Psi_r(u)$.

Alors $u \in \Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ puisque $u \in \Psi_l^{-1}(s)$. Finalement $t \in \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ car $t = \Psi_r(u)$ ce qui termine la preuve de la relation $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$.

Nous supposons maintenant qu'il existe un langage de contrôle régulier tel que $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$. Alors puisque $\mathcal{R}_c$ est régulier, nous avons $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ régulier (proposition 299).

Proposition 299. *Soit $\mathcal{R}_c$ un langage régulier. Alors $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier.*

Preuve : Soit $\mathcal{R}_c$ un langage régulier. La classe des réguliers étant close par morphismes inverses (propriété 47), $\Psi_l^{-1}(\mathcal{L}_1)$ est régulier puisque Ψ_l est un morphisme et $\mathcal{L}_1$ est régulier. Donc puisque la classe des réguliers est close par intersection (propriété 45), $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier. □

Nous allons maintenant distinguer deux cas suivant que $\mathcal{R}$ est semi-linéaire à droite ou quelconque. Nous montrons que dans tous les cas la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable et que dans le premier cas, la question « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » l'est aussi. Nous commençons par le second cas.

a) $\mathcal{R}$ quelconque

Nous montrons tout d'abord que l'on peut transformer la relation $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$ (proposition 300).

Proposition 300. *Si $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c) = \mathcal{R}el(\mathcal{L}_1)$, alors*

$$\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2 \Leftrightarrow \Psi_l^{-1}(\mathcal{L}_1) \cap R_c \subseteq \Psi_r^{-1}(\mathcal{L}_2).$$

Preuve : Supposons que $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c) = \mathcal{R}el(\mathcal{L}_1)$. Alors

$$\begin{aligned} \mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2 &\Leftrightarrow \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c) \subseteq \mathcal{L}_2 && \text{(par hypothèse)} \\ &\Leftrightarrow \Psi_l^{-1}(\mathcal{L}_1) \cap R_c^1 \subseteq \Psi_r^{-1}(\mathcal{L}_2) && \text{(d'après la propriété 298)} \end{aligned}$$

□

Nous utilisons cette équivalence pour résoudre la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » (proposition 301).

Proposition 301. *Si $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ avec $\mathcal{R}_c$ régulier, alors la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable.*

Preuve : Supposons que $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ avec $\mathcal{R}_c$ régulier. Tout d'abord, par la proposition 300, le problème « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable si et seulement si le problème « $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \subseteq \Psi_r^{-1}(\mathcal{L}_2)$? » est décidable.

Or d'après la proposition 299, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier. De plus $\Psi_r^{-1}(\mathcal{L}_2)$ est régulier puisque la classe des réguliers est close par morphismes inverses (propriété 47).

Donc puisque le problème d'inclusion est décidable pour les automates d'arbres standards (théorème 52), le problème « $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \subseteq \Psi_r^{-1}(\mathcal{L}_2)$? » est décidable. Donc le problème « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable. □

b) $\mathcal{R}$ semi-linéaire à droite

Nous supposons que $\mathcal{R}$ est semi-linéaire à droite. Alors Ψ_r est semi-linéaire puisque tout membre droit de règle est semi-linéaire.

Nous montrons tout d'abord que $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ appartient à la classe des reconnaissables par automates à tests d'égalité entre frères (proposition 302) et nous en déduisons que la question « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable (proposition 303).

Proposition 302. *Si $\mathcal{R}$ est semi-linéaire à droite et $\mathcal{R}_c$ régulier, alors $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ est un langage de la classe RATEF.*

Preuve : Supposons que $\mathcal{R}$ est semi-linéaire à droite et $\mathcal{R}_c$ régulier. Ψ_r est alors semi-linéaire. Or d'après la proposition 299, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier. Donc $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ appartient à la classe *RATEF* puisque l'image d'un régulier par un morphisme semi-linéaire appartient à *RATEF* (propriété 137). □

Proposition 303. *Si $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ avec $\mathcal{R}$ semi-linéaire à droite et $\mathcal{R}_c$ régulier, alors la question « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable.*

Preuve : Supposons que $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ avec $\mathcal{R}$ semi-linéaire à droite et $\mathcal{R}_c$ régulier. Nous déduisons de la proposition 302 que $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ appartient à *RATEF*. Or $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ donc $\mathcal{R}el(\mathcal{L}_1)$ appartient à *RATEF*.

De plus le problème de reconnaissabilité étant décidable pour la classe *RATF* (théorème 194), il l'est aussi pour la classe *RATEF* (propriété 132). On peut donc décider si $\mathcal{R}el(\mathcal{L}_1)$ est régulier. On déduit donc de la décidabilité du problème d'équivalence pour les automates d'arbres standards (théorème 53) que le problème « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable.

□

Calcul des formes normales

Pour terminer cette section, nous considérons le cas où la relation $\mathcal{R}el$ consiste à calculer des formes normales suivant une stratégie particulière.

Il faut d'abord vérifier que la stratégie est bien appliquée puis que le terme obtenu est une forme normale pour la stratégie considérée. La première étape est réalisée comme précédemment avec un langage de contrôle $\mathcal{R}_c$. Pour la seconde, on définit l'ensemble E_s des termes qui permettent à la stratégie de se poursuivre. On en déduit l'ensemble $\mathcal{R}_s^{FN} = \{t \in T(\Gamma) \mid t$ n'entoure pas u pour tout u de $E_s\}$. Finalement si on arrive à déterminer un langage de contrôle et l'ensemble E_s, on aura

$$\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}).$$

La classe de reconnaissables dont fait partie le langage $\mathcal{R}_s^{FN}$ dépend des termes de E_s. Nous distinguons deux cas suivant que E_s contient des termes quelconques ou uniquement des termes linéaires, afin d'étudier les questions « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » et « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? ».

a) E_s quelconque

Proposition 304. *Si $\mathcal{R}_c$ est régulier, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}$ est reconnaissable par AR (automate de réduction) déterministes.*

Preuve : Supposons que $\mathcal{R}_c$ soit régulier. Soit u un terme de E_s. L'ensemble des termes clos entourant u est reconnaissable par AR déterministes et complets d'après la propriété 148. Or la classe des langages reconnus par les AR déterministes est close par complément (propriété 145). Donc l'ensemble des termes n'entourant pas u est reconnaissable par AR déterministes. De plus l'ensemble E_s étant fini, la classe RAR étant close par intersection (propriété 146) et l'intersection d'automates déterministes étant un automate déterministe, $\mathcal{R}_s^{FN}$ est reconnaissable par AR déterministes.

Par la proposition 299, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier puisque $\mathcal{R}_c$ est régulier. Donc $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est reconnaissable par automates d'arbres standards déterministes d'après le théorème 38. De plus tout automate d'arbres standard déterministe est de façon évidente un automate de réduction déterministe. Donc $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est reconnaissable par AR déterministes.

Nous en déduisons que $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}$ est reconnaissable par AR déterministes. □

Proposition 305. *Si $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN})$ avec $\mathcal{R}_c$ régulier, alors la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable.*

Preuve : Supposons que $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN})$ avec $\mathcal{R}_c$ régulier. Tout d'abord, en procédant de la même façon que dans la preuve de la proposition 300, nous pouvons montrer que :

$$\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2 \;\Leftrightarrow\; \Psi_l^{-1}(\mathcal{L}_1) \cap R_c \cap \mathcal{R}_s^{FN} \subseteq \Psi_r^{-1}(\mathcal{L}_2).$$

D'où

$$\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2 \;\Leftrightarrow\; \Psi_l^{-1}(\mathcal{L}_1) \cap R_c \cap \mathcal{R}_s^{FN} \cap \complement\, \Psi_r^{-1}(\mathcal{L}_2) = \emptyset \qquad (4)$$

D'après la proposition 304, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}$ est reconnaissable par AR déterministes puisque $\mathcal{R}_c$ est régulier. De plus $\Psi_r^{-1}(\mathcal{L}_2)$ est régulier puisqu'image par morphisme inverse d'un régulier (propriété 47). Donc $\complement\Psi_r^{-1}(\mathcal{L}_2)$ est régulier puisque la classe des réguliers est

close par complément (propriété 45). Nous en déduisons que $\mathbf{C}\Psi_r^{-1}(\mathcal{L}_2)$ est reconnaissable par AR déterministes. Finalement $\Psi_l^{-1}(\mathcal{L}_1) \cap R_c \cap \mathcal{R}_s^{FN} \cap \mathbf{C}\Psi_r^{-1}(\mathcal{L}_2)$ est reconnaissable par AR déterministes.

Nous déduisons alors de la décidabilité du problème du vide pour les AR déterministes (propriété 147) et de la relation (4) que la question « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable. □

b) E_s contient uniquement des termes linéaires

Proposition 306. *Si E_s contient uniquement des termes linéaires et $\mathcal{R}_c$ est régulier, le langage $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}$ est régulier.*

Preuve : Supposons que E_s ne contienne que des termes linéaires et que $\mathcal{R}_c$ soit régulier. Soit u un terme de E_s. Puisque u est un contexte, l'ensemble des termes clos entourant u est régulier d'après la propriété 57. Or la classe des réguliers est close par complément (propriété 45). Donc l'ensemble des termes n'entourant pas u est régulier. Puisque E_s est fini et la classe des réguliers est close par intersection (propriété 45), $\mathcal{R}_s^{FN}$ est régulier.

De plus par la proposition 299, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier puisque $\mathcal{R}_c$ est régulier. Finalement $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}$ est régulier par clôture par intersection de la classe des réguliers. □

Proposition 307. *Si $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN})$ avec $\mathcal{R}_c$ régulier, $\mathcal{R}$ semi-linéaire à droite et E_s contenant uniquement des termes linéaires alors la question « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable.*

Preuve : Supposons que $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN})$ avec $\mathcal{R}_c$ régulier, $\mathcal{R}$ semi-linéaire à droite et E_s contenant uniquement des termes linéaires. Tout d'abord, $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN}$ est régulier d'après la proposition 306.

De plus $\mathcal{R}$ étant semi-linéaire à droite, on peut montrer de façon similaire à la preuve de la proposition 302 que $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c \cap \mathcal{R}_s^{FN})$ appartient à la classe $RATEF$. Nous en déduisons, en procédant comme dans la preuve de la proposition 303, que la question « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable. □

2.4 Résolution

Nous étudions les problèmes de décision énoncés dans la section précédente pour différentes relations $\mathcal{R}el$. Nous donnons pour chaque relation étudiée, le bon langage de contrôle $\mathcal{R}_c$ et les conditions pour lesquelles on a $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$.

2.4.1 Réécriture en un pas

Pour vérifier qu'une seule réécriture s'applique sur un terme, il faut qu'il possède un unique symbole de Λ. Nous définissons donc le langage de contrôle $\mathcal{R}_c^1$ par l'ensemble des termes de $T(\Gamma)$ qui ne contiennent qu'une et une seule occurrence de symbole de Λ. Clairement $\mathcal{R}_c^1$ est régulier. On peut montrer par la méthode développée dans la section 2.3 que pour tout système de réécriture $\mathcal{R}$,

$$\mathcal{R}(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c^1).$$

Puisque $\mathcal{R}_c^1$ est régulier, on a d'après les propositions 301 et 303, les résultats suivants.

Proposition 308. *Pour tout système de réécriture $\mathcal{R}$, la question « $\mathcal{R}(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable.*

Proposition 309. *Pour tout système de réécriture $\mathcal{R}$ semi-linéaire à droite, la question « $\mathcal{R}(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable.*

Une des applications du résultat de la proposition 308 est la suivante. Soit $\mathcal{L}$ un langage régulier. On a

$$\mathcal{R}(\mathcal{L}) \subseteq \mathcal{L} \Leftrightarrow \mathcal{R}^*(\mathcal{L}) = \mathcal{L}.$$

On déduit donc de la proposition 308 que la question classique « $\mathcal{R}^*(\mathcal{L}) = \mathcal{L}$? » est décidable (lemme 310).

Lemme 310. *Pour tout système de réécriture $\mathcal{R}$ et tout langage $\mathcal{L}$ régulier, la question « $\mathcal{R}^*(\mathcal{L}) = \mathcal{L}$? » est décidable.*

2.4.2 Réécriture parallèle

Pour vérifier qu'une réécriture parallèle s'applique sur un terme, il faut que chacune de ses branches possède au plus un symbole de Λ puisque les réécritures s'appliquent sur des positions parallèles. Nous définissons donc le langage de contrôle $\mathcal{R}_c^{\|}$ par l'ensemble des termes de $T(\Gamma)$ qui contiennent au plus une occurrence de symbole de Λ sur chaque branche. Clairement, $\mathcal{R}_c^{\|}$ est régulier. Par la méthode développée dans la section 2.3, nous pouvons montrer que pour tout système de réécriture $\mathcal{R}$,

$$\mathcal{R}_{\|}(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c^{\|}).$$

D'après les propositions 301 et 303, on a les résultats suivants puisque $\mathcal{R}_c^{\|}$ est régulier.

Proposition 311. *Pour tout système de réécriture $\mathcal{R}$, la question « $\mathcal{R}_{\|}(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » est décidable.*

Proposition 312. *Pour tout système de réécriture $\mathcal{R}$ semi-linéaire à droite, la question « $\mathcal{R}_{\|}(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable.*

2.4.3 Réécriture en une passe IO

Dans le cas où $\mathcal{R}el$ est le calcul des formes sententielles selon la réécriture en une passe IO, il n'y a pas de langage de contrôle et on peut montrer que :

$$\Psi_r(\Psi_l^{-1}(\mathcal{L}_1)) \subseteq \mathsf{FS}_{\mathsf{io}}(\mathcal{L}_1).$$

En général, l'inclusion est stricte car certains termes obtenus par réécriture en une passe IO d'un terme ne sont par capturés par $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1))$ (exemple 313).

Exemple 313. *Soient la signature* $\Sigma = \{f/2, g/1, a/0, b/0\}$ *et le système* $\mathcal{R}$ *sur* Σ *constitué des règles suivantes :*

$$\begin{array}{lrcl} règle_1 : & f(x,x) & \to & g(x) \\ règle_2 : & a & \to & b \end{array}$$

Nous avons $f(a,b) \to^{io} g(b)$ *puisque :*

$$f(a,b) \xrightarrow{règle_2} f(b,b) \xrightarrow{règle_1} g(b)$$

Or il n'existe pas de terme u *de* $T(\Gamma)$ *tel que* $\Psi_l(u) = f(a,b)$ *et* $\Psi_r(u) = g(b)$.

On peut montrer que si le système $\mathcal{R}$ est linéaire à gauche, on a

$$\mathsf{FS}_{\mathsf{io}}(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1)).$$

Nous obtenons donc par les propositions 301 et 303, les résultats suivants.

Proposition 314. *Pour tout système de réécriture* $\mathcal{R}$ *linéaire à gauche, la question «* $\mathsf{FS}_{\mathsf{io}}(\mathcal{L}_1) \subseteq \mathcal{L}_2$*? » est décidable.*

Proposition 315. *Pour tout système de réécriture* $\mathcal{R}$ *linéaire à gauche et semi-linéaire à droite, la question «* $\mathsf{FS}_{\mathsf{io}}(\mathcal{L}_1) = \mathcal{L}_2$*? » est décidable.*

2.4.4 Réécriture en une passe OI

Comme dans le cas précédent, lorsque $\mathcal{R}el$ est le calcul des formes sententielles selon la réécriture en une passe OI, il n'y a pas de langage de contrôle et on peut montrer que :

$$\Psi_r(\Psi_l^{-1}(\mathcal{L}_1)) \subseteq \mathsf{FS}_{\mathsf{oi}}(\mathcal{L}_1).$$

En général, l'inclusion est stricte car deux mêmes sous-termes à des positions différentes peuvent être réécrit différemment alors qu'ils ont même image par morphisme (exemple 316).

Exemple 316. *Soient la signature* $\Sigma = \{f/2, g/1, a/0, b/0\}$ *et le système* $\mathcal{R}$ *sur* Σ *constitué des règles suivantes :*

$$\begin{array}{lrcl} règle_1 : & g(x) & \to & f(x,x) \\ règle_2 : & g(x) & \to & a \\ règle_3 : & g(x) & \to & b \end{array}$$

Nous avons $g(g(a)) \to^{oi} f(a,b)$ *puisque :*

$$g(g(a)) \xrightarrow{règle_1} f(g(a),g(a)) \xrightarrow{règle_2} f(a,g(a)) \xrightarrow{règle_3} f(a,b)$$

Or il n'existe pas de terme u *de* $T(\Gamma)$ *tel que* $\Psi_l(u) = g(g(a))$ *et* $\Psi_r(u) = f(a,b)$.

On peut montrer que si le système $\mathcal{R}$ est linéaire à droite, on a

$$\mathsf{FS}_{\mathsf{oi}}(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1)).$$

Nous obtenons donc par les propositions 301 et 303, le résultat suivant.

Proposition 317. *Pour tout système de réécriture $\mathcal{R}$ linéaire à droite, les questions* « $\mathsf{FS}_{\mathsf{oi}}(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *et* « $\mathsf{FS}_{\mathsf{oi}}(\mathcal{L}_1) = \mathcal{L}_2$? » *sont décidables.*

2.4.5 Réécriture en une passe par les feuilles

Dans un premier temps, nous nous intéressons au calcul des formes sententielles des termes de $\mathcal{L}_1$ selon la réécriture en une passe par les feuilles puis à celui des formes normales selon cette même stratégie.

Pour le reste de cette section et pour la section suivante concernant la réécriture en une passe par la racine, nous notons, pour tout terme de $T(\Gamma)$, $l_\pi(t)$ le mot défini par les symboles rencontrés sur la branche π de t. Si π n'est pas une branche de t, alors $l_\pi(t) = \epsilon$.

a) Formes sententielles

Nous rappelons tout d'abord le résultat de Z. Fülöp *et al.*

Théorème 318 (Z. Fülöp *et al.*, [33]). *Pour tout système de réécriture $\mathcal{R}$ linéaire à gauche, la question* « $\mathsf{FS}_{\uparrow}(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *est décidable.*

Nous utilisons maintenant notre méthode d'étude. Nous définissons dans un premier temps notre langage de contrôle.

Pour vérifier qu'une réécriture en une passe par les feuilles s'applique sur un terme, il faut que chacune de ses branches soit constituée, en remontant de sa feuille à la racine, d'une succession de symboles de Λ suivie d'une succession de symboles de Σ. Nous définissons donc le langage de contrôle $R_c^{\uparrow s}$ par l'ensemble des termes t de $T(\Gamma)$ tels que pour toute branche π de t, $l_\pi(t) \in \Sigma^*\Lambda^*\Sigma_0 \cup \Sigma^*\Lambda^*$. Remarquons que les mots de $\Lambda\Sigma_0$ traduisent l'extension $\mathcal{R}_e$ du système $\mathcal{R}$. Nous pouvons montrer que $R_c^{\uparrow s}$ est régulier.

La méthode développée dans la section 2.3, nous permet de montrer que pour tout système de réécriture linéaire à gauche $\mathcal{R}$,

$$\mathsf{FS}_{\uparrow}(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap R_c^{\uparrow s}).$$

Remarque 319. *La restriction linéaire à gauche est indispensable pour les systèmes de réécriture pour les mêmes raisons que dans le cas de la réécriture en une passe IO.*

Nous retrouvons par la proposition 301, le résultat de Z. Fülöp *et al.* (théorème 318) et nous déduisons de la proposition 303, le résultat suivant.

Proposition 320. *Pour tout système de réécriture $\mathcal{R}$ linéaire à gauche et semi-linéaire à droite, la question* « $\mathsf{FS}_{\uparrow}(\mathcal{L}_1) = \mathcal{L}_2$? » *est décidable.*

b) Formes normales

Le résultat de Z. Fülöp *et al.* obtenu dans [33] est le suivant :

Théorème 321 (Z. Fülöp *et al.*, **[33]).** *Pour tout système de réécriture $\mathcal{R}$ linéaire à gauche, la question* « $\mathsf{FN}_\uparrow(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *est décidable.*

Soit $E^{FN}_\uparrow$ l'ensemble des termes ne devant pas apparaître dans un terme afin que celui-ci ne puisse continuer à se réécrire selon la réécriture en une passe par les feuilles.

Pour que la réécriture en une passe par les feuilles d'un terme puisse se poursuivre, il faut qu'une partie gauche de règle apparaisse juste au dessus de constantes ou de parties du terme réécrites, c'est-à-dire juste au dessus des marques (symboles de Λ). Pour tout i de $[n]$, nous notons p_i le nombre de variables distinctes de l_i. Alors nous pouvons montrer que :

$$E^{FN}_\uparrow = \{l_i[u_1, \ldots, u_{p_i}] \mid \begin{array}{l} i \in [n], \\ \forall j \in [p_i], u_j \in \Sigma_0 \text{ ou } \exists k \in [n], y_1, \ldots, y_{p_k} \in \mathcal{X}, u_j = \bullet_k(y_1, \ldots, y_{p_k}), \\ \text{et les variables de } \bigcup_{j \in [p_i]} \mathcal{V}ar(u_j) \text{ sont deux à deux distinctes}\}. \end{array}$$

On peut alors montrer que

$$\mathsf{FN}_\uparrow(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap R_c^{\uparrow s} \cap \mathcal{R}^{FN}_\uparrow)$$

avec $\mathcal{R}^{FN}_\uparrow = \{t \in T(\Gamma) \mid t$ n'entoure pas u pour tout u de $E^{FN}_\uparrow\}$.

Nous retrouvons alors par l'étude faite sur le calcul des formes normales et la proposition 305, le résultat de Z. Fülöp *et al.* (théorème 321).

De plus lorsque $\mathcal{R}$ est linéaire à gauche, les termes de $E^{FN}_\uparrow$ sont linéaires donc d'après la proposition 307, on a :

Proposition 322. *Pour tout système de réécriture $\mathcal{R}$ linéaire à gauche et semi-linéaire à droite, la question* « $\mathsf{FN}_\uparrow(\mathcal{L}_1) = \mathcal{L}_2$? » *est décidable.*

2.4.6 Réécriture en une passe par la racine

Dans un premier temps, nous nous intéressons à l'ensemble des formes sententielles des termes de $\mathcal{L}_1$ selon la réécriture en une passe par la racine puis à celui des formes normales selon cette même stratégie.

a) Formes sententielles

Nous rappelons tout d'abord le résultat de Z. Fülöp *et al.*

Théorème 323 (Z. Fülöp *et al.*, **[33]).** *Pour tout système de réécriture $\mathcal{R}$ linéaire à gauche, la question* « $\mathsf{FS}_\downarrow(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *est décidable.*

Nous utilisons maintenant notre méthode d'étude. Nous définissons dans un premier temps notre langage de contrôle.

Pour vérifier qu'une réécriture en une passe par la racine s'applique sur un terme, il faut que chacune de ses branches soit constituée, en descendant de la racine à sa feuille, d'une succession de symboles de Λ suivie d'une succession de symboles de Σ. Nous définissons donc le langage de contrôle $R_c^{\downarrow s}$ par l'ensemble des termes t de $T(\Gamma)$ tels que pour tout chemin π

de t, $l_\pi(t) \in \Lambda^*\Sigma^*$. Nous pouvons montrer que $R_c^{\downarrow s}$ est régulier et que pour tout système de réécriture linéaire à droite $\mathcal{R}$,

$$\mathsf{FS}_\downarrow(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap R_c^{\downarrow s}).$$

Remarque 324. *La linéarité à droite est indispensable pour les mêmes raisons que dans le cas de la réécriture en une passe OI.*

D'après les propositions 301 et 303, nous avons le résultat suivant.

Proposition 325. *Pour tout système de réécriture $\mathcal{R}$ linéaire à droite, les questions* « $\mathsf{FS}_\downarrow(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *et* « $\mathsf{FS}_\downarrow(\mathcal{L}_1) = \mathcal{L}_2$? » *sont décidables.*

Notre étude permet donc de trouver un résultat différent de celui de Z. Fülöp *et al.* (théorème 323) quant à la restriction sur la linéarité du système $\mathcal{R}$.

b) Formes normales

Le résultat de Z. Fülöp *et al.* obtenu dans [33] est le suivant :

Théorème 326 (Z. Fülöp *et al.*, [33]). *Pour tout système de réécriture $\mathcal{R}$ linéaire à gauche, la question* « $\mathsf{FN}_\downarrow(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *est décidable.*

Soit $E_\downarrow^{FN}$ l'ensemble des termes ne devant pas apparaître dans un terme afin que celui-ci ne puisse continuer à se réécrire selon la réécriture en une passe par la racine.

Pour que la réécriture en une passe par la racine d'un terme puisse se poursuivre, il faut qu'une partie gauche de règle apparaisse au niveau de la racine ou juste en dessous d'une partie du terme réécrite, c'est-à-dire juste en dessous d'une marque (symbole de Λ). Nous en déduisons les deux ensembles suivants permettant de déterminer les formes normales selon la réécriture en une passe pas la racine.

$$E_\downarrow^{FN} = \{\bullet_i[y_1, \ldots, y_{l-1}, l_k[x_1, \ldots, x_{p_k}], y_{l+1}, \ldots, y_{p_i}] \mid i, k \in [n], l \in [p_i] \text{ et } x_1, \ldots, x_{p_k}, y_1, \ldots, y_{l-1}, y_{k+1}, \ldots, y_{p_i} \in \mathcal{X}\}.$$

$$\mathcal{R}_{\downarrow 1}^{FN} = \{t \in T(\Gamma) \mid t \text{ n'est instance d'aucune partie gauche de règles}\}.$$

On peut alors montrer que

$$\mathsf{FN}_\downarrow(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap R_c^{\downarrow s} \cap \mathcal{R}_{\downarrow 1}^{FN} \cap \mathcal{R}_{\downarrow 2}^{FN})$$

avec $\mathcal{R}_{\downarrow 2}^{FN} = \{t \in T(\Gamma) \mid t \text{ n'entoure pas } u \text{ pour tout } u \text{ de } E_\downarrow^{FN}\}$.

Proposition 327. *$\mathcal{R}_{\downarrow 1}^{FN}$ est reconnaissable par automates de réduction déterministes et complets.*

Preuve : Soit i un entier de $[n]$. En nous inspirant de la preuve de la propriété 148 [23], montrant que l'ensemble des termes entourant un terme est reconnaissable par AR déterministes et complets, nous pouvons montrer que l'ensemble des instances closes de l_i est reconnaissable par AR déterministes et complets. Or la classe des langages reconnus par les AR déterministes est close par complément (propriété 145). Donc l'ensemble des termes qui ne sont pas instances closes de l_i est reconnaissable par AR déterministes. De plus la classe RAR étant close par intersection (propriété 146) et l'intersection d'automates déterministes étant un automate déterministe, $\mathcal{R}_{\downarrow 1}^{FN}$ est reconnaissable par AR déterministes.

Finalement, d'après la clôture par complétion de la classe des automates de réduction (propriété 142) et la remarque 143, $\mathcal{R}_{\downarrow 1}^{FN}$ est reconnaissable par AR déterministes et complets.

□

Nous déduisons alors de l'étude faite sur le calcul des formes sententielles et de la proposition 305, le résultat suivant.

Proposition 328. *Pour tout système de réécriture $\mathcal{R}$ linéaire à droite, la question* « $\mathsf{FN}_{\downarrow}(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » *est décidable.*

Notre étude permet donc de trouver un résultat différent de celui de Z. Fülöp *et al.* (théorème 326) quant à la restriction sur la linéarité du système $\mathcal{R}$.

De plus lorsque $\mathcal{R}$ est linéaire à gauche, les termes de $E_{\downarrow}^{FN}$ sont linéaires et on peut montrer que $\mathcal{R}_{\downarrow 1}^{FN}$ est régulier. Donc d'après la proposition 307, on a :

Proposition 329. *Pour tout système de réécriture $\mathcal{R}$ linéaire, la question* « $\mathsf{FN}_{\downarrow}(\mathcal{L}_1) = \mathcal{L}_2$? » *est décidable.*

2.4.7 Bilan et extension

Nous avons regroupé dans le tableau ci-dessous les résultats de Z. Fülöp *et al.* [33] ainsi que ceux énoncés ci-dessus.

	quelconque	semi-lin. droit	lin. droit	lin. gche	lin. gche semi-lin. droit	linéaire
$\mathcal{R}$	$\subseteq$	$\subseteq, =$	$\subseteq, =$	$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathcal{R}_{\|\|}$	$\subseteq$	$\subseteq, =$	$\subseteq, =$	$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathsf{FS}_{\mathsf{io}}$				$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathsf{FS}_{\mathsf{oi}}$			$\subseteq, =$			$\subseteq, =$
$\mathsf{FS}_{\uparrow}$				$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathsf{FN}_{\uparrow}$				$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathsf{FS}_{\downarrow}$			$\subseteq, =$	$\subseteq$	$\subseteq$	$\subseteq, =$
$\mathsf{FN}_{\downarrow}$			$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq, =$

Une possibilité d'extension de ces résultats est de montrer que le problème de reconnaissabilité pour la classe RAR (reconnaissables par automates de réduction) est décidable. Si c'est le cas, les restrictions de semi-linéarité disparaissent dans l'énoncé des résultats de ce chapitre.

En effet, supposons que $\mathcal{R}el(\mathcal{L}_1) = \Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ avec $\mathcal{R}_c$ régulier. Alors $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est régulier d'après la proposition 299. Si le nombre de pas de réécriture nécessaires pour

décrire $\mathcal{R}el$ est borné, le nombre de marques (symboles de Λ) apparaissant dans tout terme de $\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c$ est borné. Nous pouvons alors montrer que $\Psi_r(\Psi_l^{-1}(\mathcal{L}_1) \cap \mathcal{R}_c)$ est un langage de RAR. Donc $\mathcal{R}el(\mathcal{L}_1)$ appartient à RAR.

Nous déduisons alors de la décidabilité du problème de reconnaissabilité pour la classe RAR que la question « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » est décidable.

Cinquième partie

Contraintes ensemblistes

Cette partie, composée de quatre chapitres, est consacrée à l'étude des contraintes ensemblistes.

Dans le premier chapitre, nous rappelons les définitions des contraintes ensemblistes ainsi qu'un état de l'art rapide de leur étude. Dans le second, nous montrons en utilisant la méthode des Arbres-Grilles l'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules contenant pour seul opérateur ensembliste l'union $\cup$. Dans le troisième chapitre, nous montrons que le problème de satisfiabilité pour les systèmes de contraintes ensemblistes positives avec tests d'égalité est décidable en définissant une nouvelle classe d'automates pour laquelle le problème du vide est décidable. Finalement, dans le dernier chapitre, nous étudions les propriétés topologiques de l'ensemble des solutions de certains systèmes de contraintes. Nous déduisons de cette étude que le pouvoir d'expression en terme d'ensemble de solutions des contraintes ensemblistes positives avec symboles de projection est strictement supérieur à celui des contraintes ensemblistes positives et négatives.

Chapitre 1

Définitions - Etat de l'art

Les *contraintes ensemblistes* sont apparues (du moins sous ce nom) pour la première fois dans [41]. Elles y sont mentionnées en fait plus précisément comme "Herbrand Set Constraints" (et donc en français "Contraintes Ensemblistes de Herbrand"), ce qui est bien la terminologie la plus appropriée. En effet, une contrainte ensembliste (basique) est une formule atomique (du premier ordre) interprétée à l'aide d'une sémantique (en l'occurrence, une structure) fixée. Cette structure a pour support l'ensemble des sous-ensembles d'arbres finis (d'où le terme "Herbrand") et prend en compte des opérations et des relations ensemblistes.

Nous rappelons tout d'abord les définitions tant syntaxique que sémantique se trouvant dans [41], et reprises dans la plupart des travaux postérieurs (section 1.1). Puis nous faisons un bref état de l'art sur les résultats obtenus sur les contraintes ensemblistes (section 1.2).

1.1 Définitions

Soient Σ une signature, $\mathcal{X}$ un ensemble de variables dites *ensemblistes* et Σ_B, Σ_{-1}, Σ_n trois signatures dont les éléments sont appelés *opérateurs ensemblistes.* Σ_B est l'ensemble $\{\cup/2, \cap/2, \complement/1\}$ et ses symboles de fonctions sont appelés respectivement union, intersection et complément. Σ_{-1} est l'ensemble des symboles de fonctions unaires f_i^{-1} pour f symbole de Σ d'arité non nulle et i entier de $[Arité(f)]$, appelés *symboles de projection.* Finalement, Σ_n est composé de deux constantes : $\top$ et $\bot$.

Une *expression ensembliste* est un terme construit sur Σ, Σ_B, Σ_{-1}, Σ_n et $\mathcal{X}$, c'est-à-dire un élément de $T(\Sigma \cup \Sigma_B \cup \Sigma_{-1} \cup \Sigma_n, \mathcal{X})$. Par commodité, les symboles $\cup$ et $\cap$ sont utilisés en écriture infixée : par exemple, au lieu de $\cup(f(X,Y), f(a,Z))$, nous écrivons $f(X,Y) \cup f(a,Z)$.

Considérons maintenant l'ensemble des symboles de prédicats $\mathcal{P}$ constitué des prédicats binaires $\subseteq$, $\not\subseteq$ et $=$, appelés respectivement inclusion, non-inclusion et égalité.

Une *contrainte ensembliste* est un atome $exp_1 \subseteq exp_2$ (contrainte positive), $exp_1 \not\subseteq exp_2$ (contrainte négative), ou $exp_1 = exp_2$ (contrainte égalitaire), où exp_1 et exp_2 sont des expressions ensemblistes (exemple 330).

Un *système de contraintes ensemblistes* est une conjonction de contraintes ensemblistes. Pour SC un système de contraintes ensemblistes, on note $Var(SC)$ l'ensemble des variables de SC.

Exemple 330. *Soient les signatures* $\Sigma_1 = \{a/0, f/2\}$, $\Sigma_2 = \{0/0, s/1\}$ *et les variables* X, Y *et ENT. Alors* C_1 *et* C_2 *définis respectivement sur* Σ_1 *et* Σ_2 *sont des contraintes ensemblistes:*

$$\begin{aligned} C_1 &:= a \subseteq f(X,Y) \\ C_2 &:= ENT \subseteq 0 \cup s(ENT) \end{aligned}$$

La sémantique des contraintes ensemblistes, définie ci-dessous, est la structure notée $\mathcal{T}_{\mathrm{SC}}(\Sigma)$, de support l'ensemble des sous-ensembles d'arbres finis, et fixant une sémantique pour les symboles de fonction de Σ_B, Σ_n et Σ_{-1} et les symboles de prédicats de $\mathcal{P}$.

- Le support est $2^{T(\Sigma)}$, l'ensemble des sous-ensembles de $T(\Sigma)$,
- Les éléments et fonctions associés respectivement aux constantes et aux symboles de fonctions d'arité non nulle sont définis comme suit. Soient $E_1, \ldots, E_n \in 2^{T(\Sigma)}$,

$$\begin{aligned} e^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} &= \{e\}, \forall e \in \Sigma_0 \\ f^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}(E_1, \ldots, E_p) &= \{f(t_1, \ldots, t_p) \mid t_1 \in E_1, \ldots, t_p \in E_p\}, \forall f \in \Sigma_{>0} \\ \bot^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} &= \emptyset \\ \top^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} &= 2^{T(\Sigma)} \\ E_1 \cup^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} E_2 &= E_1 \cup E_2 \\ E_1 \cap^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} E_2 &= E_1 \cap E_2 \\ \complement^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} E_1 &= 2^{T(\Sigma)} \setminus E_1 \\ (f_i^{-1})^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}(E_1) &= \{t_i \mid f(t_1, \ldots, t_p) \in E_1\}, \forall f \in \Sigma_p, p > 0, \forall i \in [p] \end{aligned}$$

- Les relations $\subseteq^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}$, $\not\subseteq^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}$ et $=^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}$ associées respectivement aux symboles de prédicats $\subseteq$, $\not\subseteq$ et $=$, sont respectivement l'inclusion, la non-inclusion et l'égalité dans $2^{T(\Sigma)}$. Lorsqu'il n'y a aucune ambiguïté possible, nous écrivons simplement $\subseteq$, $\not\subseteq$ et $=$ au lieu de $\subseteq^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}$, $\not\subseteq^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}$ et $=^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}$.

Une *interprétation ensembliste* $\mathcal{I}$ est une application qui associe à chaque variable une valeur dans $2^{T(\Sigma)}$:

$$\mathcal{I} : \mathcal{X} \to 2^{T(\Sigma)}.$$

Toute interprétation s'étend à l'ensemble des expressions ensemblistes selon la sémantique définie par $\mathcal{T}_{\mathrm{SC}}(\Sigma)$. L'extension $\chi_{\mathcal{I}}$ d'une interprétation $\mathcal{I}$ est définie comme suit :

Pour toute variable X de $\mathcal{X}$ et toute expression ensembliste $e_1, \ldots, e_p$,

$$\begin{aligned} \chi_{\mathcal{I}}(X) &= \mathcal{I}(X) \\ \chi_{\mathcal{I}}(f(e_1, \ldots, e_p)) &= f^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}(\chi_{\mathcal{I}}(e_1), \ldots, \chi_{\mathcal{I}}(e_p)), \forall f \in \Sigma_p, p \in \mathbb{N} \\ \chi_{\mathcal{I}}(\bot) &= \bot^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} \\ \chi_{\mathcal{I}}(\top) &= \top^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} \\ \chi_{\mathcal{I}}(e_1 \cup e_2) &= \chi_{\mathcal{I}}(e_1) \cup^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} \chi_{\mathcal{I}}(e_2) \\ \chi_{\mathcal{I}}(e_1 \cap e_2) &= \chi_{\mathcal{I}}(e_1) \cap^{\mathcal{T}_{\mathrm{SC}}(\Sigma)} \chi_{\mathcal{I}}(e_2) \\ \chi_{\mathcal{I}}(\complement e_1) &= \complement^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}(\chi_{\mathcal{I}}(e_1)) \\ \chi_{\mathcal{I}}(f_i^{-1})(e_1) &= (f_i^{-1})^{\mathcal{T}_{\mathrm{SC}}(\Sigma)}(\chi_{\mathcal{I}}(e_1)) \end{aligned}$$

Une interprétation $\mathcal{I}$ satisfait une contrainte positive $exp_1 \subseteq exp_2$ si $\chi_{\mathcal{I}}(exp_1) \subseteq^{\mathcal{T}_{SC}(\Sigma)} \chi_{\mathcal{I}}(exp_2)$, une contrainte négative $exp_1 \not\subseteq exp_2$ si $\chi_{\mathcal{I}}(exp_1) \not\subseteq^{\mathcal{T}_{SC}(\Sigma)} \chi_{\mathcal{I}}(exp_2)$, une contrainte égalitaire $exp_1 = exp_2$ si $\chi_{\mathcal{I}}(exp_1) =^{\mathcal{T}_{SC}(\Sigma)} \chi_{\mathcal{I}}(exp_2)$.

Une interprétation satisfaisant une contrainte ensembliste C est appelée *solution* de C, et $Sol(C)$ désigne l'ensemble des solutions de C. Une solution d'un système de contraintes ensemblistes est une interprétation satisfaisant toutes les contraintes du système. L'ensemble des solutions d'un système SC est notée $Sol(SC)$ et si $Sol(SC)$ est non vide, SC est dit *satisfiable* (exemple 331).

Remarquons que toute interprétation $\mathcal{I}$ d'un système de contraintes ensemblistes sur n variables $X_1, \ldots, X_n$ est entièrement définie par la donnée du n-uplet $(\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n))$.

Exemple 331. *Nous considérons les contraintes définies dans l'exemple 330. La contrainte C_1 est insatisfiable puisque le singleton $\{a\}$ ne peut être inclus dans un ensemble dont tous les termes ont f pour symbole de tête.*

Par contre la seconde, C_2, possède plusieurs solutions (en fait, une infinité): ainsi, l'interprétation associant à la variable ENT, l'ensemble vide ($\emptyset$) est une solution (puisque le vide est inclus dans tous les ensembles). En fait pour toute solution $\mathcal{I}$, $\mathcal{I}(ENT)$ est inclus dans $\{s^n(0) \mid n \in \mathbb{N}\}$.

L'un des problèmes majeurs étudiés dans les chapitres suivants est le *problème de satisfiabilité* des contraintes ensemblistes énoncé ci-dessous.

Problème de satisfiabilité
Entrée : *Un système SC de contraintes ensemblistes.*
Question : *Est-ce que SC est satisfiable ?*

Maintenant que les définitions principales sur les contraintes ensemblistes ont été données, nous faisons un bref état de l'art sur les résultats obtenus pour celles-ci.

1.2 Etat de l'art

De part les études dont les contraintes ensemblistes ont fait l'objet, on distingue parmi les systèmes de contraintes ensemblistes certaines classes syntaxiques. Nous considérons dans ce chapitre certaines de ces classes et rappelons les résultats obtenus pour le problème de satisfiabilité de celles-ci.

Comme nous l'avons dit précédemment, les définitions tant syntaxique que sémantique concernant les contraintes ensemblistes sont apparues dans [41]. Cependant, cet article restreignait la définition des contraintes ensemblistes en considérant pour symbole de prédicat uniquement l'inclusion et en limitant la forme des expressions ensemblistes pouvant apparaître en partie droite et gauche d'une inclusion. Cette classe de contraintes ensemblistes a été appelée classe des *contraintes définies* et notée (DSC) (Definite Set Constraints).

En 1992, A. Aiken et E. Wimmers ont introduit une nouvelle classe de contraintes appelée *classe des contraintes positives*, notée (PSC) (Positive Set Constraints) [4]. Cette classe est constituée des contraintes ensemblistes positives sans symbole de projection.

Le problème de satisfiabilité des systèmes de contraintes de la classe (PSC) a été montré décidable par plusieurs approches :

- Transformation syntaxique des systèmes de contraintes : A. Aiken et E. Wimmers [4].
- Utilisation d'une nouvelle classe d'automates appelés automates d'ensembles d'arbres : R. Gilleron, M. Tommasi et S. Tison [35].
- Caractérisation logique du problème de satisfiabilité : L. Bachmair, H. Ganzinger et U. Waldmann [8].

Enfin une étude exhaustive de la complexité de la classe (PSC) selon l'arité des symboles de fonctions de la signature a été mené par A. Aiken, D. Kozen, M. Vardi et E. Wimmers [1]. Ces résultats sont résumés dans le tableau ci-dessous (la notation $n+$ est mise pour "n ou plus").

Nombre de symboles			Complexité du problème de satisfiabilité
d'arité 0	d'arité 1	d'arité 2+	
0	0+	0+	trivial
1+	0+	0	*NP*-complet
1+	1	0	*PSPACE*-complet
1+	2+	0	*EXPTIME*-complet
1+	0+	1+	*NEXPTIME*-complet

La classe (PSC) a été ensuite étendue en considérant en plus de la relation d'inclusion, la relation de non-inclusion. La nouvelle classe ainsi définie est la *classe des contraintes positives et négatives* notée (PSNC) (Positive and Negative Set Constraints). Cette classe est constituée des contraintes ensemblistes positives et négatives sans symbole de projection.

Le problème de satisfiabilité des systèmes de contraintes de la classe (PNSC) a également été montré décidable par plusieurs approches :

- Utilisation d'une nouvelle classe d'automates appelés automates d'ensembles d'arbres : R. Gilleron, M. Tommasi et S. Tison [37].
- Utilisation des hypergraphes : A. Aiken, D. Kozen, et E. Wimmers [2].

Enfin, dans la continuité des idées développées par L. Bachmair, H. Ganzinger et U. Waldmann, W. Charatonik et L. Pacholski [17] ont montré que le problème de satisfiabilité des systèmes de contraintes de la classe (PNSC) est *NEXPTIME*-complet.

Dans les deux classes définies ci-dessus, les symboles de projection sont interdits. L'extension de la classe (PNSC) aux contraintes contenant les projections donne la *classe des contraintes positives et négatives avec symboles de projection* notée (PNSCP) (Positive and Negative Set Constraints with Projections). Cette classe est constituée des contraintes ensemblistes positives et négatives avec symboles de projection.

Le problème de satisfiabilité de cette classe a été montré décidable par W. Charatonik et L. Pacholski dans [18].

Finalement nous définissons une dernière classe de contraintes dont l'expressivité est étudiée dans le chapitre 4. La *classe des contraintes positives avec symboles de projection*, notée (PSCP) (Positive Set Constraints with Projections), est la classe constituée des contraintes ensemblistes positives avec symboles de projection.

Chapitre 2

Indécidabilité

Soient Σ une signature et $\mathcal{X}$ un ensemble de variables ensemblistes. Nous rappelons que $\mathcal{P}$ désigne l'ensemble des symboles de prédicats constitué des prédicats binaires $\subseteq$, $\not\subseteq$ et $=$.

La théorie du premier ordre des contraintes ensemblistes est l'ensemble des formules logiques du premier ordre construites sur la signature Σ, l'ensemble de variables $\mathcal{X}$ et l'ensemble de prédicats $\mathcal{P}$. On peut également définir cette théorie comme l'ensemble des formules du premier ordre dont les atomes sont les contraintes ensemblistes. La sémantique de cette théorie est fixée par la structure $\mathcal{T}_{\mathrm{SC}}(\Sigma)$ définie au chapitre précédent.

La théorie du premier ordre des contraintes ensemblistes est indécidable en raison de l'indécidabilité de la théorie monadique des algèbres libres finiment engendrées [83]. Par contre, le fragment existentiel de cette théorie est décidable [36, 2, 18] et plusieurs algorithmes basés sur les automates d'arbres ont été présentés pour la résolution de ce problème. Dans la suite de ce chapitre, nous raffinons le résultat d'indécidabilité en montrant le théorème suivant :

Théorème 332. *Le fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules contenant pour seul opérateur ensembliste l'union $\cup$ est indécidable.*

Nous montrons ce résultat en utilisant la méthode des Arbres-Grilles que nous avons introduite dans la partie **Outils de décidabilité et d'indécidabilité.** Nous rappelons tout d'abord les différents éléments intervenant dans cette méthode.

Soit $\mathcal{M} = (\mathcal{Q}, \{a, b\}, \{a, b, \square\}, \delta, q_s, \square, q_f)$, avec $\mathcal{Q} = \{q_1, \ldots, q_k\}$, une instance de la classe restreinte des machines de Turing déterministes (définition 164). Nous définissons une signature Γ composée des symboles de fonction suivants :

$$\begin{aligned} f &: \textit{ternaire} \\ \bot_0 &: \textit{constante} \\ a_l, b_l, a_r, b_r, \square_r, q_1, \ldots, q_k &: \textit{constante} \end{aligned}$$

sur laquelle on définit un ensemble d'arbres particuliers, appelés G-Arbres (définition 175), représentant les grilles finies.

Définition 175 (G-Arbre). *Un terme clos t d'une signature contenant Γ est un* G-Arbre sur Γ si :

$$t \in T(\Gamma) \qquad \text{(PURETÉ)}$$

Pour tout sous-arbre $f(x, y, z)$ de t :

$$x = \bot_0 \text{ ou } tête(x) = f \qquad \text{(FILS-1)}$$
$$y = \bot_0 \text{ ou } tête(y) = f \qquad \text{(FILS-2)}$$
$$z \in \{a_l, b_l, a_r, b_r, \square_r, q_1, \ldots, q_k\} \qquad \text{(CONSTANTES)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), f(x_2, y_2, z_2), z_3)$ de t :

$$y_1 = x_2 \qquad \text{(EGALITÉ)}$$

Pour tout sous-arbre $f(f(x_1, y_1, z_1), y_2, z_2)$ de t :

$$tête(y_1) = f \qquad \text{(CORPS-1)}$$

Pour tout sous-arbre $f(x_1, f(f(x_2, y_2, z_2), y_3, z_3), z_1)$ de t :

$$tête(x_1) = f \qquad \text{(CORPS-2)}$$

Nous définissons ensuite un ensemble P'_T de contextes de $T(\Gamma, \mathcal{X})$ décelant les arbres qui ne codent pas un calcul de la machine $\mathcal{M}$. Finalement nous montrons que l'on peut caractériser l'ensemble des G-Arbres codant les calculs réussis de la machine $\mathcal{M}$ débutant sur l'entrée vide (proposition 177).

Proposition 177. *Un G-Arbre t sur Γ code un calcul réusi et seulement si de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide si et seulement si :*

- *t entoure l'arbre $f(x_1, f(x_2, f(x_3, \bot_0, \square_r), \square_r), q_s)$* (INIT1)
- *t n'entoure pas l'arbre $f(f(x_1, x_2, q_s), x_3, x_4)$* (INIT2)
- *t n'entoure pas l'arbre $f(x_1, f(x_2, x_3, q_s), x_4)$* (INIT3)
- *t n'entoure aucun des motifs de P'_T* (CALCUL)
- *t entoure l'arbre $f(\bot_0, x_1, q_f)$* (FINAL)

Ces rappels faits nous donnons l'idée de la preuve du théorème 332. Nous réduisons le problème de l'arrêt de la machine $\mathcal{M}$ en la satisfiabilité d'une formule, nommée *stop*, de la théorie du premier ordre des contraintes ensemblistes en utilisant la méthode des Arbres-Grilles :

$$stop := \exists X \left(grille(X) \wedge init(X) \wedge calcul(X) \wedge final(X) \right).$$

Les formules *grille*(X), *init*(X), *calcul*(X) et *final*(X) permettent de vérifier si un arbre satisfait les conditions de la proposition 177 :

- *grille*(X) teste si un arbre est un G-Arbre sur Γ (définition 175),
- *init*(X) teste si un arbre satisfait les conditions relatives à la configuration initiale (conditions INIT1, INIT2 et INIT3 de la proposition 177),

- *calcul*(X) teste si un arbre n'entoure aucun des contextes de P'_T (condition CALCUL de la proposition 177),
- *final*(X) teste si un arbre satisfait la condition sur l'état final (condition FINAL de la proposition 177).

L'idée de la preuve étant donnée, nous commençons par définir un ensemble de formules (définition 333) afin d'alléger les notations. Puis nous définissons la formule *grille*(x) (définition 334) qui teste si un arbre est un G-Arbre sur Γ (proposition 336).

Définition 333. *Soient E et F deux éléments de $2^{T(\Gamma)}$. Les formules vide(E), disjoint(E,F) et sing(E) définies ci-dessous sont satisfiables dans la structure $\mathcal{T}_{SC}(\Gamma)$ si et seulement si respectivement l'ensemble E est l'ensemble vide, les ensembles E et F sont disjoints et l'ensemble E est un singleton:*

$$\begin{aligned} \mathit{vide}(E) &:= E \subseteq a_r \wedge E \subseteq b_r \\ \mathit{disjoint}(E,F) &:= \forall Z \ (Z \subseteq E \wedge Z \subseteq F) \Rightarrow \mathit{vide}(Z) \\ \mathit{sing}(E) &:= \neg \mathit{vide}(E) \wedge \forall Z \ Z \subseteq E \Rightarrow (\mathit{vide}(Z) \vee E \subseteq Z) \end{aligned}$$

Définition 334 (*grille*(X)). *Soient C l'ensemble $\Gamma_0 \backslash \{\bot_0\}$ $\left(C = \bigcup_{c \in \Gamma_0 \backslash \{\bot_0\}} \{c\}\right)$ et grille(X) la formule définie comme suit:*

$$\mathit{grille}(X) := \mathit{sing}(X) \wedge \left(\exists Z \ \mathit{sousarbres}(X,Z) \wedge \mathit{égal}(Z) \wedge \mathit{corps1}(Z) \wedge \mathit{corps2}(Z)\right)$$

où

$$\begin{aligned} \mathit{sousarbres}(X,Z) &:= Z \subseteq f(Z,Z,C) \cup \bot_0 \ \wedge \ X \subseteq Z \\ &\qquad \wedge \ \forall Z' \left(Z' \subseteq f(Z',Z',C) \cup \bot_0 \wedge X \subseteq Z'\right) \Rightarrow Z \subseteq Z' \\ \mathit{égal}(Z) &:= \forall Y_1,Y_2,Y_3,Y_4,Y_5,Y_6,Y_7 \Big(\bigwedge_{i \in [7]} \neg \mathit{vide}(Y_i) \wedge \\ &\qquad f(f(Y_1,Y_2,Y_3),f(Y_4,Y_5,Y_6),Y_7) \subseteq Z \Big) \Rightarrow Y_2 = Y_4 \\ \mathit{corps1}(Z) &:= \forall Y_1,Y_2,Y_3,Y_4 \bigwedge_{c \in \Gamma_0} \mathit{disjoint}(f(f(Y_1,c,Y_2),Y_3,Y_4),\ Z) \\ \mathit{corps2}(Z) &:= \forall Y_1,Y_2,Y_3,Y_4,Y_5,Y_6 \\ &\qquad \bigwedge_{c \in \Gamma_0} \mathit{disjoint}(f(c,f(f(Y_1,Y_2,Y_3),Y_4,Y_5),Y_6),\ Z) \end{aligned}$$

La définition de la formule *grille*(X) étant donné nous allons montrer qu'elle permet bien de tester si un arbre est un G-Arbre sur Γ (proposition 336). Mais avant tout, nous montrons (lemme 335) que la formule *sousarbres*(X,Z) définie l'ensemble des sous-arbres d'un arbre satisfaisant les conditions FILS-1, FILS-2 et CONSTANTES de la définition des G-Arbres (définition 175).

Lemme 335. *Soient T et S deux éléments de $2^{T(\Gamma)}$. Alors $\mathcal{T}_{SC}(\Gamma) \models sing(T) \wedge sousarbres(T,S)$ si et seulement s'il existe un terme t de $T(\Gamma)$ satisfaisant les conditions* FILS-1, FILS-2 *et* CONSTANTES *de la définition des G-Arbres (définition 175) tel que $T = \{t\}$ et S soit l'ensemble des sous-termes de t privé des constantes de C.*

Preuve : Soient T et S deux éléments de $2^{T(\Gamma)}$. Supposons tout d'abord que $\mathcal{T}_{SC}(\Gamma) \models sing(T) \wedge sousarbres(T,S)$. Puisque $\mathcal{T}_{SC}(\Gamma) \models sing(T)$, il existe, d'après la définition 333, un terme t de $T(\Gamma)$ tel que $T = \{t\}$.

D'après la formule $sousarbres(T,S)$, pour tout terme t' de S, soit $t' = \perp_0$, soit il existe t_1, t_2 termes de S et t_3 constante de C tels que $t' = f(t_1,t_2,t_3)$. De plus t appartient à S puisque $\{t\} \subseteq S$. Donc t satisfait les conditions FILS-1, FILS-2 et CONSTANTES de la définition des G-Arbres (définition 175) et S contient l'ensemble E des sous-arbres de t exceptés les éléments de C.

Finalement comme pour tout élément S' de $2^{T(\Gamma)}$, nous avons $\mathcal{T}_{SC}(\Gamma) \models (S' \subseteq f(S',S',C) \cup \perp_0 \wedge \{t\} \subseteq S') \Rightarrow S \subseteq S'$, S est le plus petit ensemble contenant E d'où $S = E$ ce qui termine la première partie de la preuve.

Supposons maintenant qu'il existe un arbre t de $T(\Gamma)$ satisfaisant les conditions FILS-1, FILS-2 et CONSTANTES de la définition des G-Arbres (définition 175) tel que $T = \{t\}$ et S soit l'ensemble des sous-arbres de t privé des constantes de C. Puisque T est un singleton, nous avons $\mathcal{T}_{SC}(\Gamma) \models sing(T)$ d'après la définition 333.

De plus, comme t satisfait les conditions FILS-1, FILS-2 et CONSTANTES de la définition des G-Arbres (définition 175), nous avons pour tout sous-arbre t' de S, soit $t' = \perp_0$, soit il existe t_1,t_2 termes de S et t_3 constante de C tels que $t' = f(t_1,t_2,t_3)$. Donc $\mathcal{T}_{SC}(\Gamma) \models S \subseteq f(S,S,C) \cup \perp_0 \wedge T \subseteq S$. Finalement comme S est le plus petit ensemble vérifiant cette formule, nous avons pour tout élément S' de $2^{T(\Gamma)}$, $\mathcal{T}_{SC}(\Gamma) \models (S' \subseteq f(S',S',C) \cup \perp_0 \wedge \{t\} \subseteq S') \Rightarrow S \subseteq S'$. Nous en déduisons que $\mathcal{T}_{SC}(\Gamma) \models sousarbres(T,S)$. Finalement $\mathcal{T}_{SC}(\Gamma) \models sing(T) \wedge sousarbres(T,S)$. □

Proposition 336. *Soit t un arbre de $T(\Gamma)$ alors t est un G-Arbre sur Γ si et seulement si $\mathcal{T}_{SC}(\Gamma) \models grille(\{t\})$.*

Preuve : Nous avons de façon évidente, les propriétés suivantes pour tout élément S de $2^{T(\Gamma)}$:

- $\mathcal{T}_{SC}(\Gamma) \models corps1(S)$ si et seulement si tout arbre $f(f(t_1,t_2,t_3),t_4,t_5)$ de S satisfait la condition CORPS-1 de la définition 175;
- $\mathcal{T}_{SC}(\Gamma) \models corps2(S)$ si et seulement si tout arbre $f(t_1,f(f(t_2,t_3,t_4),t_5,t_6),t_7)$ de S satisfait la condition CORPS-2 de la définition 175.

De plus $\mathcal{T}_{SC}(\Gamma) \models égal(S)$ si et seulement si tout arbre $f(f(t_1,t_2,t_3),f(t_4,t_5,t_6),t_7)$ de S satisfait la condition EGALITÉ de la définition 175. En effet, supposons que $\mathcal{T}_{SC}(\Gamma) \models égal(S)$. Alors pour tout terme $t_1,\ldots,t_7$ de $T(\Gamma)$:

$$\begin{aligned} f(f(t_1,t_2,t_3),f(t_4,t_5,t_6),t_7) \in S &\Rightarrow f(f(\{t_1\},\{t_2\},\{t_3\}),f(\{t_4\},\{t_5\},\{t_6\}),\{t_7\}) \subseteq S \\ &\Rightarrow \{t_2\} = \{t_4\} \\ &\Rightarrow t_2 = t_4 \end{aligned}$$

Donc tout arbre $f(f(t_1,t_2,t_3),f(t_4,t_5,t_6),t_7)$ de S satisfait la condition EGALITÉ de la définition 175.

Supposons maintenant que tout arbre $f(f(t_1,t_2,t_3),f(t_4,t_5,t_6),t_7)$ de S satisfait la condition EGALITÉ de la définition 175. Soient $Y_1,\ldots,Y_7$ ensembles non vides de $2^{T(\Gamma)}$ tels que $f(f(Y_1,Y_2,Y_3),f(Y_4,Y_5,Y_6),Y_7) \subseteq S$.

Pour tout i de [7], soit t_i un terme de Y_i. Alors $f(f(t_1,t_2,t_3),f(t_4,t_5,t_6),t_7)$ appartient à S. Donc $t_2 = t_4$ par la condition EGALITÉ. Nous en déduisons que Y_2 et Y_4 sont des singletons et que $Y_2 = Y_4$. Donc $\mathcal{T}_{SC}(\Gamma) \models \mathit{égal}(S)$.

Finalement si t est un arbre de $T(\Gamma)$, nous avons $\mathcal{T}_{SC}(\Gamma) \models \mathit{grille}(\{t\})$ si et seulement s'il existe S élément de $2^{T(\Gamma)}$ tel que $\mathcal{T}_{SC}(\Gamma) \models (\mathit{sing}(\{t\}) \wedge \mathit{sousarbres}(\{t\},S) \wedge \mathit{égal}(S) \wedge \mathit{corps1}(S) \wedge \mathit{corps2}(S))$. Nous déduisons donc des propriétés précédentes et du lemme 335 que si t un arbre de $T(\Gamma)$ alors t est un G-Arbre sur Γ si et seulement si $\mathcal{T}_{SC}(\Gamma) \models \mathit{grille}(\{t\})$. □

Maintenant que nous savons reconnaître à partir d'une formule les G-Arbres, nous allons utiliser la proposition 177 pour reconnaître les G-Arbres codant les calculs réussis de la machine $\mathcal{M}$ débutant sur l'entrée vide. Nous définissons pour cela les formules $\mathit{init}(X)$, $\mathit{calcul}(X)$ et $\mathit{final}(X)$ qui nous permettent de vérifier si un arbre satisfait les conditions énoncées par la proposition 177 (définition 339). Les conditions de cette proposition nécessitent de reconnaître des motifs dans un arbre, nous définissons donc tout d'abord la formule $\mathit{match}[p](S)$ (définition 337) qui réalise cette opération (proposition 338).

Définition 337 ($\mathit{match}[p](Z)$). *Pour tout entier n et tout contexte p de $C^n(\Gamma)$, nous définissons la formule suivante :*

$$\mathit{match}[p](Z) := \exists X_1,\ldots,X_n \; p[X_1,\ldots,X_n] \subseteq Z \wedge \neg \mathit{vide}(p[X_1,\ldots,X_n]).$$

Proposition 338. *Pour tout entier n, tout contexte p de $C^n(\Gamma)$ et tout élément S de $2^{T(\Gamma)}$, $\mathcal{T}_{SC}(\Sigma) \models \mathit{match}[p](S)$ si et seulement s'il existe une instance close de p appartenant à S.*

Preuve : Soit n un entier, p un contexte de $C^n(\Gamma)$ et S un élément de $2^{T(\Gamma)}$. Supposons que $\mathcal{T}_{SC}(\Gamma) \models \mathit{match}[p](S)$ alors il existe n ensembles $S_1,\ldots,S_n$ de $2^{T(\Gamma)}$ tels que $\mathcal{T}_{SC}(\Gamma) \models p[S_1,\ldots,S_n] \subseteq S$ et $\mathcal{T}_{SC}(\Gamma) \models \neg \mathit{vide}(p[S_1,\ldots,S_n])$. Donc les ensembles $S_1,\ldots,S_n$ sont non vides et il existe pour tout i de $[n]$, un terme t_i de S_i tel que $p[t_1,\ldots,t_n]$ appartienne à S. Donc $p[t_1,\ldots,t_n]$ est une instance close de p appartenant à S.

Supposons maintenant qu'il existe une instance close de p appartenant à S. Alors il existe $t_1,\ldots,t_n$ arbres de $T(\Gamma)$ tels que $p[t_1,\ldots,t_n]$ appartienne à S. Nous avons donc

$$\mathcal{T}_{SC}(\Gamma) \models (p[\{t_1\},\ldots,\{t_n\}] \subseteq S \wedge \neg \mathit{vide}(p[\{t_1\},\ldots,\{t_n\}])$$

d'où $\mathcal{T}_{SC}(\Gamma) \models \mathit{match}[p](S)$. □

Nous définissons maintenant les formules $\mathit{init}(X)$, $\mathit{calcul}(X)$ et $\mathit{final}(X)$ ainsi que la formule *stop* (définition 339) qui code le problème de l'arrêt sur l'entrée vide des machines de Turing en une formule du fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes. Finalement nous en déduisons la preuve du théorème 332.

Définition 339 (*stop*).

$$
\begin{aligned}
\mathit{init}(Z) &:= \mathit{match}\big[f(x_1, f(x_2, f(x_3, \bot_0, \Box_r), \Box_r), q_s)\big](Z) \\
&\quad \wedge \neg \mathit{match}\big[f(f(x_1, x_2, q_s), x_3, x_4)\big](Z) \\
&\quad \wedge \neg \mathit{match}\big[f(x_1, f(x_2, x_3, q_s), x_4)\big](Z) \\
\mathit{calcul}(Z) &:= \bigwedge_{p \in P'_T} \neg \mathit{match}\,[p](Z) \\
\mathit{final}(Z) &:= \mathit{match}\big[f(\bot_0, x_1, q_f)\big](Z) \\
\mathit{stop} &:= \exists X\ \mathit{sing}(X) \wedge \big(\exists Z\ \mathit{sousarbres}(X, Z) \wedge \mathit{égal}(Z) \wedge \mathit{corps1}(Z) \wedge \mathit{corps2}(Z) \\
&\quad \wedge \mathit{init}(Z) \wedge \mathit{calcul}(Z) \wedge \mathit{final}(Z)\big)
\end{aligned}
$$

Preuve : théorème 332. Nous déduisons de la proposition 338 que pour tout élément S de $2^{T(\Gamma)}$:

- $\mathcal{T}_{\mathrm{SC}}(\Gamma) \models \mathit{init}(S)$ si et seulement si l'ensemble S contient une instance close de l'arbre

$$f(x_1, f(x_2, f(x_3, \bot_0, \Box_r), \Box_r), q_s)$$

  mais aucune instance close des arbres $f(f(x_1, x_2, q_s), x_3, x_4)$ et $f(x_1, f(x_2, x_3, q_s), x_4)$ (conditions INIT1, INIT2 et INIT3 de la proposition 177),

- $\mathcal{T}_{\mathrm{SC}}(\Gamma) \models \mathit{calcul}(S)$ si et seulement si S ne contient aucune instance close des motifs de P'_T (condition CALCUL de la proposition 177),

- $\mathcal{T}_{\mathrm{SC}}(\Gamma) \models \mathit{final}(S)$ si et seulement si S contient une instance close de $f(\bot_0, x_1, q_f)$ (condition FINAL de la proposition 177).

Nous déduisons de ces propriétés et des propositions 336 et 177 que la formule *stop* est satisfiable dans la structure $\mathcal{T}_{\mathrm{SC}}(\Gamma)$ si et seulement s'il existe un G-Arbre sur Γ satisfaisant les conditions de la proposition 177. Donc la formule *stop* est satisfiable si et seulement s'il existe un calcul réussi de la machine de Turing $\mathcal{M}$ débutant sur l'entrée vide.

De plus la formule *stop* est une formule close dont une forme prénexe appartient au fragment $\exists^*\forall^*$ et dans laquelle seul l'opérateur ensembliste d'union $\cup$ apparaît. Nous déduisons donc de l'indécidabilité du problème de l'arrêt des machines de Turing sur l'entrée vide que le fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules contenant pour seul opérateur ensembliste l'union $\cup$ est indécidable. □

J.M. Talbot [84] a raffiné ce résultat en montrant que l'indécidabilité demeure lorsqu'aucun opérateur ensembliste apparaît dans les formules.

Théorème 340. *Le fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules ne contenant aucun opérateur ensembliste est indécidable.*

Chapitre 3

Décidabilité

Dans ce chapitre, nous enrichissons la syntaxe des contraintes ensemblistes en introduisant de nouveaux opérateurs ensemblistes, dits opérateurs de diagonalisation, permettant de prendre en compte les non linéarités des termes au niveau des termes frères. Nous montrons que le problème de satisfiabilité pour les systèmes de contraintes ensemblistes positives avec tests d'égalité entre frères est décidable. Pour cela, nous nous inspirons de la preuve de R. Gilleron, S. Tison et M. Tommasi de la décidabilité du problème de satisfiabilité pour les systèmes de la classe (PNSC) faisant intervenir une nouvelle classe d'automates appelés automates d'ensembles d'arbres généralisés [37].

Nous commençons par rappeler les travaux de R. Gilleron, S. Tison et M. Tommasi sur les automates d'ensembles d'arbres généralisés (section 3.1). Ces rappels faits, nous étendons les automates d'ensembles d'arbres généralisés en leur ajoutant des tests entre frères et nous étudions les propriétés des nouveaux automates ainsi obtenus (section 3.2). Finalement nous utilisons ces automates pour étudier les contraintes ensemblistes avec tests d'égalité (section 3.3).

3.1 Automates d'ensembles d'arbres généralisés

Les automates d'ensembles d'arbres généralisés introduits par R. Gilleron, S. Tison et M. Tommasi dans [37] sont des automates capables de reconnaître des applications de $T(\Sigma)$ dans E, où Σ est une signature et E un ensemble fini. Dans le cas où $E = \{0,1\}^n$, un tel automate est alors un reconnaisseur de n-uplets de langages d'arbres. Les automates d'ensembles d'arbres généralisés possèdent de bonnes propriétés de clôture et le problème du vide est décidable pour ceux-ci. De ces propriétés, R. Gilleron, S. Tison et M. Tommasi déduisent la décidabilité du problème de satisfiabilité pour les systèmes de contraintes de la classe (PNSC).

Dans cette section, nous présentons tout d'abord la notion d'ensemble d'arbres généralisés (section 3.1.1) puis les automates d'ensembles d'arbres généralisés (section 3.1.2) et les propriétés de ces automates (section 3.1.3). Finalement dans la section 3.1.4, nous rappelons le lien existant entre ces automates et les contraintes de la classe (PNSC).

3.1.1 Ensembles d'arbres généralisés

A et E étant deux ensembles finis, les arbres infinis E-valués sur A^* sont les applications de A^* dans E. Si $E = \{0,1\}^n$, un arbre E-valués sur A^* peut être considéré comme l'application caractéristique d'un n-uplet de langages sur A, c'est-à-dire un mot de A^* appartient à la $i^{\text{ième}}$ composante du n-uplet de langages d'arbres si et seulement si la $i^{\text{ième}}$ composante de son image par l'application vaut un. Ces notions sont étendus en introduisant des applications de $T(\Sigma)$ dans E où Σ est une signature. De telles applications sont appelées ensembles d'arbres généralisés E-valués.

Définition 341. *Soient Σ une signature et E un ensemble fini. Un* Ensemble d'Arbres Généralisé (EAG) *E-valué est une application de $T(\Sigma)$ dans E. L'ensemble des EAG E-valué est noté $\mathcal{G}_E^\Sigma$.*

Si l'ensemble E est l'ensemble $\{0,1\}^n$ pour n un entier non nul et g un EAG de $\mathcal{G}_{\{0,1\}^n}^\Sigma$, g est appelée l'*EAG caractéristique* du n-uplet $\mathcal{L}(g) = (\mathcal{L}_1, \ldots, \mathcal{L}_n)$ de langages d'arbres sur Σ où pour tout i de $[n]$, $\mathcal{L}_i = \{t \in T(\Sigma) \mid g(t)_i = 1\}$ avec $g(t)_i$ la $i^{\text{ième}}$ composante du vecteur $g(t)$.

Rappelons maintenant les opérations définies sur les EAG. Soient E et E' deux ensembles finis et g (respectivement g') un EAG de $\mathcal{G}_E^\Sigma$ (respectivement $\mathcal{G}_{E'}^\Sigma$). L'EAG $g \uparrow g'$ de $\mathcal{G}_{E\times E'}^\Sigma$ est défini par $g \uparrow g'(t) = (g(t), g'(t))$ pour tout terme t de $T(\Sigma)$. Inversement si g est un EAG de $\mathcal{G}_{E\times E'}^\Sigma$ et π la projection de $E \times E'$ dans E alors $\pi(g)$ est l'EAG de $\mathcal{G}_E^\Sigma$ défini par $\pi(g)(t) = \pi(g(t))$ pour tout terme t de $T(\Sigma)$. Finalement si $G \subseteq \mathcal{G}_{E\times E'}^\Sigma$ et $G' \subseteq \mathcal{G}_E^\Sigma$ alors $\pi(G) = \{\pi(g) \mid g \in G\}$ et $\pi^{-1}(G') = \{g \in \mathcal{G}_{E\times E'}^\Sigma \mid \pi(g) \in G'\}$.

3.1.2 Automates d'ensembles d'arbres généralisés (AEAG)

Nous montrons dans cette section comment R. Gilleron, S. Tison et M. Tommasi dans [37] étendent aux EAG la notion d'automates d'arbres infinis vus comme reconnaisseurs d'arbres infinis. Ils définissent pour cela un modèle de reconnaisseurs pour les EAG appelé automate d'ensembles d'arbres généralisé.

Définition 342. *Soient E un ensemble fini dont les éléments sont appelés* étiquettes *et Σ une signature. Un* Automate d'Ensembles d'Arbres Généralisés *(AEAG) sur E est un quadruplet $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ où :*

1. *Σ est une signature;*
2. *$\mathcal{Q}$ est un ensemble fini d'états (symboles de fonction d'arité 0);*
3. *$\Delta \subseteq \bigcup_p \mathcal{Q}^p \times \Sigma_p \times E \times \mathcal{Q}$ est un ensemble de règles dites de* transition*;*
4. *$\Omega \subseteq 2^{\mathcal{Q}}$ est un ensemble d'ensembles d'états dits* ensembles acceptants.

Par la suite, une règle $(q_1, \ldots, q_p, f, l, q)$ de Δ sera aussi notée $f(q_1, \ldots, q_p) \xrightarrow[l]{} q$.

Remarque 343. *Les états d'un AEAG sont des constantes alors que pour les automates d'arbres standards et à contraintes se sont des symboles de fonction unaires.*

Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ un AEAG sur E. Un *calcul* de $\mathcal{A}$ sur un EAG g de $\mathcal{G}_E^\Sigma$ est une application r de $T(\Sigma)$ dans $\mathcal{Q}$ telle que pour tout symbole f de Σ d'arité p et tous termes $t_1, \ldots, t_p$ de $T(\Sigma)$:

$$f(r(t_1), \ldots, r(t_p)) \xrightarrow[g(f(t_1,\ldots,t_p))]{} r(f(t_1, \ldots, t_p)) \in \Delta.$$

Donc une règle $f(q_1, \ldots, q_p) \underset{l}{\rightarrow} q$ peut s'appliquer sur un terme $t = f(t_1, \ldots, t_p)$ dans un calcul r sur l'EAG g si $r(t_1) = q_1, \ldots, r(t_p) = q_p$ et t est étiqueté par l c'est-à-dire $g(t) = l$. Si la règle est appliquée alors $r(t) = q$.

Remarque 344. *Un calcul r dans un automate d'arbres standards ou à contraintes sur un arbre t est une application de $\mathcal{P}os(t)$ dans l'ensemble des états. Tandis qu'un calcul d'un AEAG sur un EAG est une application r de $T(\Sigma)$ dans l'ensemble des états donc pour tout terme t, il existe un état q tel que $r(t) = q$.*

Un calcul r est *réussi* si le rang de r est dans Ω c'est-à-dire $r(T(\Sigma))$ appartient à Ω. Un EAG g de $\mathcal{G}_E^\Sigma$ est *accepté* par $\mathcal{A}$ s'il existe un calcul réussi de $\mathcal{A}$ sur g. L'ensemble des EAG de $\mathcal{G}_E^\Sigma$ acceptés par $\mathcal{A}$ est noté $\mathcal{G}(\mathcal{A})$:

$$\mathcal{G}(\mathcal{A}) = \{g \in \mathcal{G}_E^\Sigma \mid g \text{ accepté par } \mathcal{A}\}.$$

Un ensemble G d'éléments de $\mathcal{G}_E^\Sigma$ ($G \subseteq \mathcal{G}_E^\Sigma$) est *reconnaissable par AEAG* si $G = \mathcal{G}(\mathcal{A})$ pour un AEAG $\mathcal{A}$.

Comme pour les automates d'arbres standards et à contraintes, nous pouvons distinguer différents types d'AEAG.

- $\mathcal{A}$ est *déterministe* si pour tout tuple $(q_1, \ldots, q_p, f, l)$ de $Q^p \times \Sigma_p \times E$, il existe au plus un état q de $\mathcal{Q}$ tel que $f(q_1, \ldots, q_p) \underset{l}{\rightarrow} q \in \Delta$.

- $\mathcal{A}$ est *fortement déterministe* si pour tout tuple $(q_1, \ldots, q_p, f)$ de $Q^p \times \Sigma_p$, il existe au plus une paire (l, q) de $E \times \mathcal{Q}$ tel que $f(q_1, \ldots, q_p) \underset{l}{\rightarrow} q \in \Delta$.

- $\mathcal{A}$ est *complet* si pour tout tuple $(q_1, \ldots, q_p, f, l)$ de $Q^p \times \Sigma_p \times E$, il existe au moins un état q de $\mathcal{Q}$ tel que $f(q_1, \ldots, q_p) \underset{l}{\rightarrow} q \in \Delta$.

- $\mathcal{A}$ est *simple* si Ω est clos par sous-termes c'est-à-dire :

$$\forall \omega, \omega' \, (\omega \in \Omega \wedge \omega' \subseteq \omega) \Rightarrow \omega' \in \Omega.$$

Il existe en général un nombre infini de calculs même dans le cas d'AEAG déterministes (exemple 346). Mais dans le cas d'AEAG fortement déterministe il existe au plus un calcul (exemple 345).

Exemple 345. *Soient* $E = \{0, 1\}$, $\Sigma = \{0/0, s/1, nil/1, cons/2\}$ *et* $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ *l'AEAG sur* E *défini par* $\mathcal{Q} = \{Nat, Liste, Terme\}$, $\Omega = 2^{\mathcal{Q}}$ *et* Δ *constitué des règles suivantes :*

$$\begin{aligned}
0 &\xrightarrow[0]{} Nat \\
s(Nat) &\xrightarrow[0]{} Nat \\
nil &\xrightarrow[1]{} List \\
Cons(Nat, List) &\xrightarrow[1]{} List \\
Cons(q, q') &\xrightarrow[0]{} Term \quad \forall (q, q') \neq (Nat, List) \\
s(q) &\xrightarrow[0]{} Term \quad \forall q \neq Nat
\end{aligned}$$

$\mathcal{A}$ est fortement déterministe, simple mais non complet et $\mathcal{G}(\mathcal{A})$ est un singleton. En effet, il existe un unique calcul r sur un unique EAG g. Le calcul r associe à un entier naturel l'état Nat, à une liste l'état List et aux autres termes l'état Term. Donc g associe à un entier naturel la valeur 0, à une liste la valeur 1 et aux autres termes la valeur 0. Donc $\mathcal{G}(\mathcal{A})$ est l'ensemble des listes d'entiers naturels.

Exemple 346. *Nous prenons les mêmes définitions que dans l'exemple 345 et nous considérons l'AEAG* $\mathcal{A}' = (\Sigma, \mathcal{Q}, \Delta', \Omega)$ *sur* E *où* $\Delta' = \Delta \cup \{nil \xrightarrow[0]{} List, Cons(Nat, List) \xrightarrow[0]{} List\}$.

$\mathcal{A}'$ est déterministe, simple mais non complet et $\mathcal{G}(\mathcal{A}')$ est l'ensemble des sous-ensembles de $\mathcal{G}(\mathcal{A})$. En effet, les calculs réussis peuvent maintenant être définis sur des EAG g où les terme de $\mathcal{G}(\mathcal{A})$ sont étiquetés par 0 *ou* 1.

3.1.3 Propriétés

Nous rappelons l'ensemble des propriétés de la classe des ensembles d'EAG reconnaissables par AEAG et des AEAG exhibés par R. Gilleron, S. Tison et M. Tommasi dans [37]. Dans la suite, E désigne un ensemble d'étiquettes et Σ une signature.

La classe des ensembles d'EAG sur E reconnaissables est close par union et intersection mais pas par complément. Donc si G_1, G_2 sont des ensembles d'EAG de $\mathcal{G}_E^\Sigma$ reconnaissables, alors $G_1 \cup G_2$ et $G_1 \cap G_2$ sont reconnaissables.

Si $E = E_1 \times E_2$, alors la classe des ensembles d'EAG sur E reconnaissables est close par projection, c'est-à-dire si π est la projection de E sur E_1 alors pour tout G inclus dans $\mathcal{G}_E^\Sigma$ reconnaissable, $\pi(G)$ inclus dans $\mathcal{G}_{E_1}^\Sigma$ est reconnaissable. De plus la classe des ensembles d'EAG sur E_1 reconnaissables est close par projection inverse, c'est-à-dire pour tout G' inclus dans $\mathcal{G}_{E_1}^\Sigma$ reconnaissable, $\pi^{-1}(G')$ inclus dans $\mathcal{G}_E^\Sigma$ est reconnaissable.

Finalement si G_1 et G_2 sont des ensembles reconnaissables d'EAG de $\mathcal{G}_{E_1}^\Sigma$ et $\mathcal{G}_{E_2}^\Sigma$ respectivement, alors l'ensemble $G_1 \uparrow G_2 = \{g_1 \uparrow g_2 \mid g_1 \in G_1, g_2 \in G_2\}$ est un ensemble d'EAG de $\mathcal{G}_{E_1 \times E_2}^\Sigma$ reconnaissable.

Considérons maintenant les propriétés de la classe des AEAG sur E:

1. Pour tout AEAG $\mathcal{A}$ sur E, il existe un AEAG complet $\mathcal{A}_c$ sur E tel que $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{A}_c)$.
2. Pour tout AEAG déterministe et complet $\mathcal{A}_{cd}$ sur E, il existe un AEAG $\mathcal{A}$ sur E tel que $\mathcal{G}(\mathcal{A}) = \mathcal{G}_E^\Sigma \setminus \mathcal{G}(\mathcal{A}_{cd})$.
3. La classe des ensembles d'EAG sur E reconnaissables par AEAG n'est pas close par complément.
4. La classe des AEAG sur E n'est pas close par déterminisation.

Finalement les résultats de décision obtenus dans la classe des AEAG sont les suivants:

1. Le problème du vide est décidable pour les AEAG sur E, c'est-à-dire pour un AEAG $\mathcal{A}$ sur E, on peut décider si $\mathcal{G}(\mathcal{A}) = \emptyset$.
2. Les problèmes d'inclusion et d'équivalence sont décidables pour les AEAG sur E déterministes.
3. Soit $\mathcal{A}$ un AEAG sur E déterministe. On peut décider si $\mathcal{G}(\mathcal{A})$ est un singleton ou non.
4. Soient $\mathcal{A}$ un AEAG sur $E = \{0,1\}^n$, n entier non nul, et $I \subseteq [n]$. Alors on peut décider s'il existe un n-uplet $(L_1, \ldots, L_n)$ de $\mathcal{G}(\mathcal{A})$ tel que pour tout i de I, L_i est fini.

3.1.4 Contraintes ensemblistes et AEAG

Nous rappelons maintenant le lien entre les AEAG et les contraintes établi par R. Gilleron, S. Tison et M. Tommasi [37]. Pour tout système de contraintes SC de la classe (PNSC) (respectivement (PSC)) sur n variables $X_1, \ldots, X_n$, il existe un AEAG $\mathcal{A}$ sur $\{0,1\}^n$ déterministe (respectivement déterministe et simple) tel que $\mathcal{G}(\mathcal{A}) = Sol(SC)$. On en déduit le théorème suivant:

Théorème 347 (R. Gilleron, S. Tison et M. Tommasi [37]). *Le problème de satisfiabilité pour les systèmes de contraintes de la classe (PNSC) est décidable.*

3.2 AEAG avec tests entre frères

Dans cette section, nous étendons les travaux de R. Gilleron, S. Tison et M. Tommasi énoncés dans la section précédente pour définir des automates d'ensembles d'arbres généralisés avec tests entre frères et pour étudier les propriétés de ces nouveaux automates.

3.2.1 Définitions

Définition 348. *Soit p un entier. Nous désignons par EC_p l'ensemble des conjonctions de contraintes égalitaires et diségalitaires faisant intervenir uniquement des positions de $[p]$.*

Définition 349. *Soient E un ensemble fini d'étiquettes et Σ une signature. Un* AEAG avec tests entre frères *(AEAGF) sur E est un quadruplet $\mathcal{A} = (\Sigma, Q, \Delta, \Omega)$ où:*

1. *Σ est une signature*

2. $\mathcal{Q}$ *est un ensemble fini d'états;*

3. $\Delta \subseteq \bigcup_p Q^p \times \Sigma_p \times E \times EC_p \times Q$ *est un ensemble de règles dites de transition;*

4. $\Omega \subseteq 2^{\mathcal{Q}}$ *est un ensemble d'ensembles d'états dits ensembles acceptants.*

Par la suite, une règle $(q_1, \ldots, q_p, f, l, c, q)$ de Δ est aussi notée $f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q$ et $f(q_1, \ldots, q_p)$ est appelé *membre gauche* de la règle.

Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ un AEAGF. Un *calcul* (respectivement *calcul partiel* sur $T(\Sigma)_{\leq k}$ avec k entier) de $\mathcal{A}$ sur un EAG g de $\mathcal{G}_E^\Sigma$ est une application r de $T(\Sigma)$ (respectivement $T(\Sigma)_{\leq k}$) dans $\mathcal{Q}$ telle que pour tout symbole f de Σ d'arité p et tous termes $t_1, \ldots, t_p$ de $T(\Sigma)$ (respectivement $T(\Sigma)_{<k}$) :

$r(f(t_1, \ldots, t_p)) = q$ si et seulement s'il existe une contrainte de c de EC_p telle que $f(r(t_1), \ldots, r(t_p)) \xrightarrow[g(f(t_1,\ldots,t_p))]{[c]} r(f(t_1, \ldots, t_p))$ appartienne à Δ et $f(t_1, \ldots, t_p) \vDash c$.

Donc une règle $f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q$ peut s'appliquer sur un terme $t = f(t_1, \ldots, t_p)$ dans un calcul r sur l'EAG g si $r(t_1) = q_1, \ldots, r(t_p) = q_p$, t est étiqueté par l $(g(t) = l)$ et $f(t_1, \ldots, t_p \vDash c$. Si la règle est appliquée alors $r(t) = q$.

Un calcul (respectivement un calcul partiel sur $T(\Sigma)_{\leq k}$ avec k entier) r est *réussi* si le rang de r est dans Ω, c'est-à-dire $r(T(\Sigma))$ (respectivement $r(T(\Sigma)_{\leq k})$) appartient à Ω. Un EAG g de $\mathcal{G}_E^\Sigma$ est *accepté* par $\mathcal{A}$ s'il existe un calcul réussi de $\mathcal{A}$ sur g. L'ensemble des EAG de $\mathcal{G}_E^\Sigma$ acceptés par $\mathcal{A}$ est noté $\mathcal{G}(\mathcal{A})$:

$$\mathcal{G}(\mathcal{A}) = \{g \in \mathcal{G}_E^\Sigma \mid g \text{ accepté par } \mathcal{A}\}.$$

Un ensemble $G \in \mathcal{G}_E^\Sigma$ est *reconnaissable par AEAGF* si $G = \mathcal{G}(\mathcal{A})$ pour un AEAGF $\mathcal{A}$.

Comme pour les AEAG, nous définissons différents types d'AEAGF.

- $\mathcal{A}$ est *déterministe* si pour tout tuple $(q_1, \ldots, q_p, f, l)$ de $Q^p \times \Sigma_p \times E$ et tout tuple $(t_1, \ldots, t_p)$ de termes de $T(\Sigma)$, il existe au plus une règle $f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q$ de Δ telle que $f(t_1, \ldots, t_p) \vDash c$.

- $\mathcal{A}$ est *fortement déterministe* si pour tout tuple $(q_1, \ldots, q_p, f)$ de $Q^p \times \Sigma_p$ et tout tuple $(t_1, \ldots, t_p)$ de termes de $T(\Sigma)$, il existe au plus une règle $f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q$ de Δ telle que $f(t_1, \ldots, t_p) \vDash c$.

- $\mathcal{A}$ est *complet* si pour tout tuple $(q_1, \ldots, q_p, f, l)$ de $Q^p \times \Sigma_p \times E$ et tout tuple $(t_1, \ldots, t_p)$ de termes de $T(\Sigma)$, il existe au moins une règle $f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q$ de Δ telle que $f(t_1, \ldots, t_p) \vDash c$.

- $\mathcal{A}$ est *simple* si Ω est clos par sous-termes c'est-à-dire :

$$\forall \omega, \omega' \, (\omega \in \Omega \wedge \omega' \subseteq \omega) \Rightarrow \omega' \in \Omega.$$

De façon évidente, un AEAGF est déterministe si et seulement si pour toute paire de règles $f(q_1,\ldots,q_p) \xrightarrow[l]{[c]} q$ et $f(q_1,\ldots,q_p) \xrightarrow[l]{[c']} q'$ avec $q \neq q'$, la contrainte $c \wedge c'$ est insatisfiable. Et un AEAGF est complet si et seulement si pour tout tuple $(q_1,\ldots,q_p,f,l)$ de $Q^p \times \Sigma_p \times E$ la disjonction des expressions c_i des règles $f(q_1,\ldots,q_p) \xrightarrow[l]{[c_i]} q'_i$ de Δ est équivalente à la contrainte nulle ($\top$).

Les AEAG étant des cas particuliers des AEAGF, il existe en général un nombre infini de calculs même dans le cas d'AEAGF déterministes (exemple 346). Mais dans le cas d'AEAGF fortement déterministes, il existe au plus un calcul (exemple 345).

Dans le cas où $E = \{0,1\}^n$ avec n entier et $\mathcal{A}$ est un AEAGF sur E, nous notons $\mathcal{L}(\mathcal{A}) = \{\mathcal{L}(g) \mid g \in \mathcal{G}(\mathcal{A})\}$. Dans la suite, nous confondons $\mathcal{L}(\mathcal{A})$ et $\mathcal{G}(\mathcal{A})$.

3.2.2 Propriétés de clôture

Nous étudions maintenant les propriétés de clôture de la classe des ensembles d'EAG reconnaissables par AEAGF et des AEAGF. Nous montrons que ces propriétés sont similaires à celles des AEAG (section 3.1.3).

Comme pour les automates d'arbres à tests entre frères, tout AEAGF est équivalent à un AEAGF à contraintes pleines (construction : on transforme chaque expression de contraintes de chaque règle en une disjonction de contraintes pleines équivalente et on divise la règle suivant chaque membre de la disjonction). Par souci de simplicité, nous utilisons dans certaines des preuves (en particulier celle de la décidabilité du problème du vide) des automates à contraintes pleines. Remarquons que pour tout tuple $(t_1,\ldots,t_p)$ de termes de $T(\Sigma)$, il existe une seule contrainte pleine c telle que $f(t_1,\ldots,t_p) \vDash c$ avec f symbole de Σ_p.

Proposition 350. *1. Pour tout AEAGF $\mathcal{A}$ sur E, il existe un AEAGF complet $\mathcal{A}_c$ sur E tel que $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{A}_c)$.*

2. Pour tout AEAGF déterministe et complet $\mathcal{A}_{cd}$ sur E, il existe un AEAGF $\mathcal{A}$ sur E tel que $\mathcal{G}(\mathcal{A}) = \mathcal{G}_E^{\Sigma} \setminus \mathcal{G}(\mathcal{A}_{cd})$.

3. La classe des ensembles d'EAG sur E reconnaissables par AEAGF n'est pas close par complément.

4. La classe des AEAGF sur E n'est pas close par déterminisation.

Preuve : 1. Soit $\mathcal{A} = (\Sigma, Q, \Delta, \Omega)$ un AEAGF sur E à contraintes pleines. Nous considérons l'AEAGF sur E, $\mathcal{A}_c = (\Sigma, Q_c, \Delta_c, \Omega_c)$ défini par :

- $Q_c = Q \cup \{q\}$ où q est un nouvel état ($q \notin Q$),
- $\Delta_c = \Delta \cup \{f(q_1,\ldots,q_p) \xrightarrow[l]{[c]} q \mid (q_1,\ldots,q_p,f,l,c,q) \in \bigcup_p Q_c^p \times \Sigma_p \times E \times EC'_p \times Q_c$ et $\{(q_1,\ldots,q_p,f,l)\} \times EC'_p \times Q_c \cap \Delta = \emptyset\}$,
- $\Omega_c = \Omega$.

De façon évidente, $\mathcal{A}_c$ est complet et $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{A}_c)$.

2. Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ un AEAGF déterministe et complet sur E. Nous considérons l'AEAGF sur E, $\mathcal{A}' = (\Sigma, \mathcal{Q}', \Delta', \Omega')$ défini par :

- $\mathcal{Q}' = \mathcal{Q}$,
- $\Delta' = \Delta$,
- $\Omega' = 2^{\mathcal{Q}} \setminus \Omega$.

Nous déduisons du déterminisme et de la complétude de $\mathcal{A}$ que $\mathcal{G}(\mathcal{A}') = \mathcal{G}_E^\Sigma \setminus \mathcal{G}(\mathcal{A})$.

3. La preuve de cette propriété dans le cas de reconnaissables par AEAG fait intervenir une signature dont les symboles de fonction sont d'arité maximum un [37]. Pour une telle signature, la classe des AEAGF est égale à la classe des AEAG. Nous en déduisons que la preuve de la propriété 3 dans le cas des reconnaissables par AEAGF est la même que dans le cas des reconnaissables par AEAG.

4. Supposons que la classe des AEAGF sur E soit close par déterminisation. Alors si $\mathcal{A}$ est un AEAGF, il existe un AEAGF déterministe à contraintes pleines $\mathcal{A}_d$ tel que $\mathcal{G}(\mathcal{A}_d) = \mathcal{G}(\mathcal{A})$. D'après la propriété 1, il existe un AEAGF complet $\mathcal{A}_c$ sur E tel que $\mathcal{G}(\mathcal{A}_c) = \mathcal{G}(\mathcal{A}_d)$. Par construction cet automate est aussi déterministe puisque toutes les contraintes des règles sont pleines. Donc d'après la propriété 2, il existe un AEAGF $\mathcal{A}'$ tel que $\mathcal{G}(\mathcal{A}') = \mathcal{G}_E^\Sigma \setminus \mathcal{G}(\mathcal{A}_c)$ et donc la classe des ensembles d'EAG sur E reconnaissables par AEAGF est close par complément ce qui contredit la propriété 3. Nous en déduisons que la classe des AEAGF sur E n'est pas close par déterminisation. □

Comme la classe des ensembles d'EAG reconnaissables par AEAG, la classe des ensembles reconnaissables d'EAG par AEAGF est close par union et intersection.

Proposition 351 (Union, intersection). *La classe des ensembles d'EAG sur E reconnaissables par AEAGF est close par union et intersection. Donc si G_1 et G_2 sont deux ensembles d'EAG de $\mathcal{G}_E^\Sigma$ reconnaissables par AEAGF, alors $G_1 \cup G_2$ et $G_1 \cap G_2$ sont reconnaissables par AEAGF.*

Preuve : Soient G_1 et G_2 deux ensembles d'EAG de $\mathcal{G}_E^\Sigma$ reconnaissables par AEAGF, et $\mathcal{A}_1 = (\Sigma, \mathcal{Q}_1, \Delta_1, \Omega_1)$ et $\mathcal{A}_2 = (\Sigma, \mathcal{Q}_2, \Delta_2, \Omega_2)$ deux AEAGF sur E à contraintes pleines reconnaissant respectivement G_1 et G_2. Sans perte de généralité, nous pouvons supposer que $\mathcal{Q}_1$ et $\mathcal{Q}_2$ sont disjoints.

Soit l'AEAGF $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ avec $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, $\Delta = \Delta_1 \cup \Delta_2$ et $\Omega = \Omega_1 \cup \Omega_2$. Alors de façon immédiate, nous avons $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{A}_1) \cup \mathcal{G}(\mathcal{A}_2) = G_1 \cup G_2$ donc la classe des ensembles d'EAG sur E reconnaissables par AEAG est close par union.

Montrons maintenant que $G_1 \cap G_2$ est reconnaissable par AEAGF. D'après la propriété 1 de la proposition 350, on peut supposer que $\mathcal{A}_1$ et $\mathcal{A}_2$ sont complets. Nous désignons alors par π_1 (respectivement π_2) la projection de $\mathcal{Q}_1 \times \mathcal{Q}_2$ sur $\mathcal{Q}_1$ (respectivement $\mathcal{Q}_2$) et nous considérons l'AEAGF $\mathcal{A}' = (\Sigma, \mathcal{Q}', \Delta', \Omega')$ défini par :

- $\mathcal{Q}' = \mathcal{Q}_1 \times \mathcal{Q}_2$,
- $(f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q \in \Delta') \Leftrightarrow (\forall i \in [2], f(\pi_i(q_1), \ldots, \pi_i(q_p)) \xrightarrow[l]{[c]} \pi_i(q) \in \Delta_i)$,

- $\Omega' = \{\omega \in 2^{\mathcal{Q}} \mid \forall i \in [2], \pi_i(\omega) \in \Omega_i\}$.

Alors nous pouvons montrer que si r est un calcul réussi de $\mathcal{A}'$, les applications :

$$\begin{array}{rcl} r_1 : T(\Sigma) & \to & \mathcal{Q}_1 \\ t & \to & \pi_1(r(t)) \end{array} \qquad \begin{array}{rcl} r_2 : T(\Sigma) & \to & \mathcal{Q}_2 \\ t & \to & \pi_2(r(t)) \end{array}$$

sont des calculs réussis de $\mathcal{A}_1$ et $\mathcal{A}_2$ respectivement. De plus si r_1 et r_2 sont des calculs réussis de $\mathcal{A}_1$ et $\mathcal{A}_2$ respectivement, l'application :

$$\begin{array}{rcl} r : T(\Sigma) & \to & \mathcal{Q}' \\ t & \to & (r_1(t), r_2(t)) \end{array}$$

est un calcul réussi de $\mathcal{A}'$. Nous en déduisons que $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{A}_1) \cap \mathcal{G}(\mathcal{A}_2) = G_1 \cap G_2$ donc la classe des ensembles d'EAG sur E reconnaissables par AEAG est close par intersection. □

Nous montrons maintenant que la classe des ensembles d'EAG sur E reconnaissables par AEAGF est close par projection et projection inverse (corollaire 353). Pour cela, nous montrons le lemme suivant.

Lemme 352. *Soient E_1 et E_2 deux ensembles finis d'étiquettes, $\mathcal{G} \subseteq \mathcal{G}^{\Sigma}_{E_1}$ un ensemble reconnaissable par AEAGF et $R \subseteq E_1 \times E_2$. Alors l'ensemble $R(\mathcal{G}) = \{g' \in \mathcal{G}^{\Sigma}_{E_2} \mid \exists g \in \mathcal{G}, \forall t \in T(\Sigma), (g(t), g'(t)) \in R\}$ est reconnaissable par AEAGF.*

Preuve : Soit $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ un AEAGF tel que $\mathcal{G}(\mathcal{A}) = \mathcal{G}$ et $\mathcal{A}' = (\Sigma, \mathcal{Q}', \Delta', \Omega')$ l'AEAGF sur E_2 défini par:

- $\mathcal{Q}' = \mathcal{Q}$,
- $\Delta' = \{f(q_1, \ldots, q_p) \xrightarrow[l']{[c]} q \mid \exists l \in E_1, f(q_1, \ldots, q_p) \xrightarrow[l]{[c]} q \in \Delta \text{ et } (l, l') \in R\}$,
- $\Omega' = \Omega$.

Nous montrons tout d'abord que $R(\mathcal{G}) = \mathcal{G}(\mathcal{A}')$.

$\supseteq$ Soient g' un EAG de $\mathcal{G}(\mathcal{A}')$ et r' un calcul réussi de $\mathcal{A}'$ sur g'. Nous montrons sur la structure des termes qu'il existe un EAG g de $\mathcal{G}^{\Sigma}_{E_1}$ tel que pour tout terme t de $T(\Sigma)$, $(g(t), g'(t))$ appartienne à R, et tel que r' soit un calcul réussi de $\mathcal{A}$ sur g.

Soit a une constante. Puisque r' est un calcul de $\mathcal{A}'$ sur g', on a $a \xrightarrow[g'(a)]{} r'(a) \in \Delta'$. Donc d'après la définition de Δ', il existe une étiquette l_a de E_1 telle que $a \xrightarrow[l_a]{} r'(a)$ soit une règle de Δ et $(l_a, g'(a))$ appartienne à R. Nous prenons alors $g(a) = l_a$.

Soit t un terme de $T(\Sigma)$ de hauteur non nulle. Alors il existe f symbole de Σ d'arité n non nulle et $t_1, \ldots, t_n$ termes de $T(\Sigma)$ tels que $t = f(t_1, \ldots, t_n)$. Puisque r' est un calcul de $\mathcal{A}'$ sur g', il existe une règle $f(r'(t_1), \ldots, r'(t_p)) \xrightarrow[g'(t)]{[c]} r'(t)$ dans Δ'. Donc d'après la définition de Δ', il existe une étiquette l_t de E_1 telle que $f(q_1, \ldots, q_p) \xrightarrow[l_t]{[c]} r'(t)$ soit une règle de Δ et $(l_t, g'(t))$ appartienne à R. Nous prenons alors $g(t) = l_t$.

De façon évidente, g appartient à $\mathcal{G}^{\Sigma}_{E_1}$ et r' est un calcul réussi de $\mathcal{A}$ sur g puisque $\Omega' = \Omega$. Donc g appartient à $\mathcal{G}(\mathcal{A})$ donc à $\mathcal{G}$. De plus pour tout terme t de $T(\Sigma)$, $(g(t), g'(t))$ appartient à R donc g' appartient à $R(\mathcal{G})$ et $\mathcal{G}(\mathcal{A}') \subseteq R(\mathcal{G})$.

$\subseteq$ Soient g' un EAG de $R(\mathcal{G})$. Alors il existe un EAG g de $\mathcal{G}$ tel que pour tout terme t de $T(\Sigma)$, $(g(t), g'(t))$ appartienne à R. Soit r un calcul réussi de $\mathcal{A}$ sur g. Nous montrons sur la structure des termes que r est un calcul réussi de $\mathcal{A}'$ sur g'.

Soit a une constante. Puisque r est un calcul de $\mathcal{A}$ sur g, la règle $a \xrightarrow[g(a)]{} r(a)$ appartient à Δ. De plus $(g(a), g'(a))$ appartient à R donc d'après la définition de Δ', $a \xrightarrow[g'(a)]{} r(a)$ est une règle de Δ'.

Soit t un terme de $T(\Sigma)$ de hauteur non nulle. Alors il existe f symbole de Σ d'arité n non nulle et $t_1, \ldots, t_n$ termes de $T(\Sigma)$ tels que $t = f(t_1, \ldots, t_n)$. Puisque r est un calcul de $\mathcal{A}$ sur g, il existe une règle $f(r(t_1), \ldots, r(t_p)) \xrightarrow[g(t)]{[c]} r(t)$ dans Δ. De plus $(g(t), g'(t))$ appartient à R donc d'après la définition de Δ', $f(q_1, \ldots, q_p) \xrightarrow[g'(t)]{[c]} r(t)$ est une règle de Δ'.

De façon évidente, r est un calcul réussi de $\mathcal{A}'$ sur g' puisque $\Omega' = \Omega$. Donc g' appartient à $\mathcal{G}(\mathcal{A}')$ et $R(\mathcal{G}) \subseteq \mathcal{G}(\mathcal{A}')$.

Finalement $R(\mathcal{G}) = \mathcal{G}(\mathcal{A}')$ donc $R(\mathcal{G})$ est reconnaissable par AEAGF. □

Corollaire 353. *1. La classe des ensembles d'EAG reconnaissables par AEAGF est close par projection et projection inverse.*

2. Soient $G_1 \in \mathcal{G}^{\Sigma}_{E_1}$ et $G_2 \in \mathcal{G}^{\Sigma}_{E_2}$ reconnaissables par AEAGF. Alors l'ensemble $G_1 \uparrow G_2 = \{g_1 \uparrow g_2 \mid g_1 \in G_1, g_2 \in G_2\}$ est un ensemble d'EAG de $\mathcal{G}^{\Sigma}_{E_1 \times E_2}$ reconnaissable par AEAGF.

Preuve : 1. Soient E et E' deux ensembles finis d'étiquettes, et π la projection de $E \times E'$ sur E. Alors en posant $E_1 = E \times E'$, $E_2 = E$ et $R = \pi$, on déduit du lemme 352 que la classe des ensembles d'EAG est reconnaissable par AEAGF est close par projection.

De façon similaire, en posant $E_1 = E$, $E_2 = E \times E'$ et $R = \pi^{-1}$, on déduit du lemme 352 que la classe des ensembles d'EAG est reconnaissable par AEAGF est close par projection inverse ce qui termine le point 1 du corollaire 353.

2. Soient E_1 et E_2 deux ensembles finis d'étiquettes, et $G_1 \subseteq \mathcal{G}^{\Sigma}_{E_1}$ et $G_2 \subseteq \mathcal{G}^{\Sigma}_{E_2}$ reconnaissables par AEAGF. Alors si π_1 (respectivement π_2) est la projection de $E_1 \times E_2$ sur E_1 (respectivement E_2), on a $G_1 \uparrow G_2 = \pi_1^{-1}(G_1) \cap \pi_2^{-1}(G_2)$. Nous déduisons donc de la clôture par projection inverse (point 1) et par intersection (proposition 351) des reconnaissables par AEAGF que $G_1 \uparrow G_2$ est reconnaissable par AEAGF. □

3.2.3 Décision du vide

Théorème 354. *Le problème du vide est décidable pour les AEAGF simples c'est-à-dire pour un AEAGF simple $\mathcal{A}$, on peut décider si $\mathcal{G}(\mathcal{A}) = \emptyset$.*

De façon évidente, les étiquettes des EAG n'ont aucune incidence dans le problème du vide pour les AEAGF simples donc nous considérons dans la suite des AEAGF simples sans

étiquette. L'ensemble des règles de transition d'un tel automate $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ est une partie de $\bigcup_p Q^p \times \Sigma_p \times EC_p \times Q$.

Pour savoir quelles règles peuvent être appliquées, il est suffisant de compter pour chaque état q les termes ayant q pour image dans un calcul en se bornant à l'arité maximum de la signature Σ [11]. Nous allons mémoriser quels sont les états atteints à chaque moment d'un calcul. Donc nous définissons pour chaque calcul partiel sa frontière F (ensemble des états atteints par les termes de hauteur maximum) et son intérieur I (multi-ensemble des états atteints par les termes de hauteur strictement inférieure à la hauteur maximum).

Nous construisons alors l'arbre des calculs (c'est-à-dire l'arbre des choix des règles appliquées dans la construction d'un calcul). Nous étiquetons chaque nœud de cet arbre avec son couple (I, F) correspondant. Nous montrons alors que l'on peut trouver sur un chemin de l'arbre des calculs, une étiquette (I', F') sous une étiquette (I, F) vérifiant $I =_m I'$ ($=_m$ étant l'égalité multi-ensemblistes) et $F' \subseteq F$ si et seulement s'il existe un calcul réussi.

Le nombre de couple (I, F) étant borné par $(p+1)^{card(\mathcal{Q})} \times 2^{card(\mathcal{Q})}$ où p est l'arité maximale de Σ, nous construisons l'arbre des calculs jusqu'à cette hauteur. Nous pouvons alors décider si l'ensemble des EAG acceptés par $\mathcal{A}$ est vide ou non. Nous en déduisons que le problème du vide est décidable pour les AEAGF simples.

L'idée de la preuve étant donnée, nous rappelons quelques définitions sur les multi-ensembles.

Un *multi-ensemble* A d'états de $\mathcal{Q}$ est une application de $\mathcal{Q}$ dans $\mathbb{N}$. Nous définissons les opérations suivantes sur les multi-ensembles. Soient A et A' deux multi-ensembles d'états de $\mathcal{Q}$.

Somme : $\forall q \in Q \quad (A + A')(q) = A(q) + A'(q)$

Différence : $\forall q \in Q \quad (A - A')(q) = max(0, A(q) - A'(q))$

Pour les relations d'inclusion et d'égalité des multi-ensembles, nous définissons deux types de relations :

Relations multi-ensemblistes :
$$\begin{cases} A \subseteq_m A' & \Leftrightarrow & \forall q \in Q \quad A(q) \leq A'(q) \\ A =_m A' & \Leftrightarrow & (A \subseteq_m A') \wedge (A' \subseteq_m A) \end{cases}$$

Relations ensemblistes :
$$\begin{cases} A \subseteq_s A' & \Leftrightarrow & (\forall q \in Q \quad A(q) > 0 \Rightarrow A'(q) > 0) \\ A =_s A' & \Leftrightarrow & (A \subseteq_s A') \wedge (A' \subseteq_s A) \end{cases}$$

Par la suite, nous comptons les termes atteignant chaque état en se bornant à l'arité maximum de la signature Σ. Nous associons donc à tout multi-ensemble un multi-ensemble dont les images sont bornées par p. Soient $\mathcal{M}ul$ l'ensemble des multi-ensembles et $\mathcal{M}ul_p$ l'ensemble des multi-ensembles de rang $\{0, 1, \ldots, k\}$ avec k entier. Nous notons $\max_k$ l'application de $\mathcal{M}ul$ dans $\mathcal{M}ul_k$ définie par :

$$\begin{array}{rccl} \forall A \in \mathcal{M}ul, \ \max_k(A) : Q & \rightarrow & \{0, 1, \ldots, k\} \\ q & \mapsto & min(k, A(q)) \end{array}$$

Nous définissons maintenant les notions d'intérieur et de frontière associées à tout calcul partiel (définition 355). Soient $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ un AEAG simple et p l'arité maximum de la signature Σ.

Définition 355. *Soient j un entier et r un calcul partiel de $\mathcal{A}$ sur $T_{\Sigma\leq j}$. Nous notons pour tous entiers i de $[j+1]$ et $i' < j$, I_i* (intérieur) *et $F_{i'}$* (frontière) *les multi-ensembles suivants:*

$$\begin{aligned} I_i &=_m max_p(r(T(\Sigma)_{<i})) \\ F_{i'} &=_m max_1(r(T(\Sigma)_{=i'})) \end{aligned}$$

Proposition 356. *Soient j un entier et r un calcul partiel de $\mathcal{A}$ sur $T_{\Sigma\leq j}$. Alors les frontières et intérieurs satisfont les propriétés suivantes:*

$$\forall i \in [j] \quad I_i \subseteq_m I_{i+1} \quad (\textit{la suite } (I_i)_{i\leq j} \textit{ est croissante}) \tag{1}$$

$$\forall i < j \quad F_i \subseteq_m I_{i+1} \tag{2}$$

Preuve : Soient j un entier et r un calcul partiel de $\mathcal{A}$ sur $T_{\Sigma\leq j}$.

Propriété (1). Soit i un entier de $[j]$. Nous avons $r(T(\Sigma)_{<i}) \subseteq r(T(\Sigma)_{<i+1})$ donc pour tout état q de $\mathcal{Q}$, nous avons $\{t \in T(\Sigma)_{<i} \mid r(t) = q\} \subseteq \{t \in T(\Sigma)_{<i+1} \mid r(t) = q\}$. Donc $\max_p(r(T(\Sigma)_{<i})) \subseteq_m \max_p(r(T(\Sigma)_{<i+1}))$ d'où $I_i \subseteq_m I_{i+1}$.

Propriété (2). Soit i un entier tel que $i < j$. Pour tout q de $\mathcal{Q}$, $\{t \in T(\Sigma)_{=i} \mid r(t) = q\} \subseteq \{t \in T(\Sigma)_{<i+1} \mid r(t) = q\}$. Donc $F_i \subseteq_m I_{i+1}$. □

Nous montrons maintenant qu'il existe un calcul réussi sur $\mathcal{A}$ si et seulement s'il existe i et j entiers avec $i < j$, et r un calcul partiel de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$ (proposition 359). La condition est montrée nécessaire directement dans la preuve de la proposition. Par contre, pour montrer qu'elle est suffisante nous montrons les lemmes 357 et 358.

Lemme 357. *Soient i et j entiers avec $i < j$ et r un calcul partiel de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$. Alors $I_{i+1} =_m I_{j+1}$.*

Preuve : Supposons qu'il existe i et j entiers avec $i < j$ et r un calcul partiel de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$. Nous avons alors:

$$\begin{aligned} I_{j+1} &=_m max_p(r(T_{\Sigma<j+1})) \\ &=_m max_p(r(T_{\Sigma<j}) + r(T_{\Sigma=j})) \\ &=_m max_p(I_j + r(T_{\Sigma=j})) \\ &=_m max_p(I_i + r(T_{\Sigma=j})) \end{aligned} \tag{3}$$

Soit q un état de $\mathcal{Q}$. Il existe un entier k de $\{0, 1, \ldots, p\}$ tel que $I_{i+1}(q) = k$. D'après la propriété 1 de la proposition 356 et les hypothèses, nous avons:

$$\begin{cases} I_i(q) \leq I_{i+1}(q) \leq I_j(q) \\ I_i(q) = I_j(q) \end{cases}$$

Nous en déduisons que $I_i(q) = I_{i+1}(q) = I_j(q) = k$. Montrons que $I_{j+1}(q) = I_{i+1}(q)$ en distinguant deux cas suivant la valeur de k ($k < p$ ou $k = p$).

Cas $k < p$: Nous avons la suite d'implications suivante:

$$\begin{aligned} I_i(q) = I_{i+1}(q) < p &\Rightarrow r(T(\Sigma)_{=i})(q) = 0 \\ &\Rightarrow F_i(q) = 0 \\ &\Rightarrow F_j(q) = 0 \qquad \text{car } F_j \subseteq_s F_i \\ &\Rightarrow r(T(\Sigma)_{=j})(q) = 0 \\ &\Rightarrow I_{j+1}(q) = I_j(q) \\ &\Rightarrow I_{j+1}(q) = I_{i+1}(q). \end{aligned}$$

Cas $k = p$: D'après la propriété (3), $I_{j+1}(q) = p$. De plus $I_{i+1}(q) = p$ donc $I_{j+1}(q) = I_{i+1}(q)$.

Finalement $I_{i+1} =_m I_{j+1}$. □

Lemme 358. *Soient i et j entiers avec $i < j$ et r un calcul partiel r de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$. Alors r s'étend sur $T(\Sigma)$ de sorte que pour tout entier $k \geq j$:*

$$F_k \subseteq_s F_{k+(i-j)} \quad et \quad I_k =_m I_{k+(i-j)}.$$

Preuve : Supposons qu'il existe i et j entiers avec $i < j$ et r un calcul partiel r de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$. Nous prouvons par induction sur la valeur de k que r s'étend sur $T(\Sigma)$ de sorte que pour tout entier $k \geq j$:

$$F_k \subseteq_s F_{k+(i-j)} \quad \text{et} \quad I_k =_m I_{k+(i-j)}.$$

Considérons le cas $k = j+1$. D'après le lemme 357, $I_{i+1} =_m I_{j+1}$. Etendons maintenant r sur $T(\Sigma)_{=j+1}$ de sorte que $F_{j+1} \subseteq_s F_{i+1}$.

Construction de r sur $T(\Sigma)_{=j+1}$: Nous rappelons que $F_{j+1} =_m \max_1(r(T(\Sigma)_{=j+1}))$.

Les frontières et intérieurs nous permettent d'imiter à la hauteur $j+1$ une partie suffisante de ce qui a été fait à la hauteur $i+1$. En fait, si $f(t_1, \ldots, t_n)$ est un terme de $T(\Sigma)_{=j+1}$, nous définissons son image en appliquant une règle appliquée à la hauteur $i+1$. Pour cela nous allons tout d'abord déterminer un terme $f(u_1, \ldots, u_n)$ de $T(\Sigma)_{=i+1}$ tel que $(t_i)_{i\in[n]}$ et $(u_i)_{i\in[n]}$ satisfassent la même contrainte pleine.

Soient $f(t_1, \ldots, t_n)$ un terme de $T(\Sigma)_{=j+1}$ et c la contrainte pleine de EC'_n telle que $f(t_1, \ldots, t_n) \vDash c$. Soit $(E_k)_{k\in I}$, I ensemble, une partition de $[n]$ regroupant les termes égaux de la famille $(t_k)_{k\in[n]}$ c'est-à-dire :

$$\forall k \in I, \forall l, l' \in E_k, \quad t_l = t_{l'} \tag{4}$$

$$\forall k, k' \in I, k \neq k', \forall l \in E_k, \forall l' \in E_{k'}, \quad t_l \neq t_{l'} \tag{5}$$

Pour tout k de I et tous indices l, l' de E_k, $r(t_l) = r(t_{l'})$ puisque $t_l = t_{l'}$. Soit $(S_k)_{k\in J}$, J ensemble, une partition de I regroupant les termes ayant même état image c'est-à-dire :

$$\forall k \in J, \forall l, l' \in S_k, \forall m \in E_l, \forall m' \in E_{l'} \quad r(t_m) = r(t_{m'}) \tag{6}$$

$$\forall k, k' \in J, k \neq k', \forall l \in S_k, \forall l', \in S_{k'}, \forall m \in E_l, \forall m' \in E_{l'} \quad r(t_m) \neq r(t_{m'}) \tag{7}$$

Pour tout k de J, nous notons q_k l'état de $\mathcal{Q}$ tel que $\forall l \in S_k$, $\forall m \in E_l$, $r(t_m) = q_k$. Il existe $(t_l^k)_{k\in J, l\in S_k}$ termes de $T(\Sigma)_{<i+1}$ distincts deux à deux tels que :

- $\forall k \in J, \forall l \in S_k, r(t_l^k) = q_k$ puisque $card(S_k) \leq I_{j+1}(q_k)$ et $I_{j+1}(q_k) = I_{i+1}(q_k)$.
- $\exists k \in J, \exists l \in S_k, t_l^k \in T(\Sigma)_{=i}$ puisqu'il existe $m \in [n]$ tel que $t_m \in T(\Sigma)_{=j}$ et car $F_j \subseteq_s F_i$ par hypothèse.

Soient $u_1, \ldots, u_n$ termes de $T(\Sigma)_{<i+1}$ définis par $\forall k \in J, \forall l \in S_k, \forall l' \in E_l, u_{l'} = t_l^k$. Soit $s_1, \ldots, s_n$ états de $\mathcal{Q}$ défini par quelquesoit m appartenant à $[n]$, $r(t_m) = s_m$. Alors $f(u_1, \ldots, u_n) \vDash c$ et pour tout entier m de $[n]$, $r(u_m) = s_m$. Puisque r est un calcul

partiel sur $T(\Sigma)_{\leq j}$, il existe une règle $f(s_1,\ldots,s_n) \xrightarrow{[c']} q$ de Δ telle que $f(u_1,\ldots,u_n) \vDash c'$ et $q = r(f(u_1,\ldots,u_n))$. Comme c et c' sont des contraintes pleines, nous avons $c = c'$ puisque $f(u_1,\ldots,u_n) \vDash c$ et $f(u_1,\ldots,u_n) \vDash c'$. Finalement la règle $f(s_1,\ldots,s_n) \xrightarrow{[c]} q$ appartient à Δ.

On a $f(t_1,\ldots,t_n) \vDash c$. De plus pour tout entier m de $[n]$, $r(t_m) = s_m$, donc nous pouvons appliquer la règle précédente au terme $f(t_1,\ldots,t_n)$. Nous définissons donc $r(f(t_1,\ldots,t_n))$ par $r(f(t_1,\ldots,t_n)) = q$.

Notons que par définition $F_{i+1}(q) = 1$ puisque $f(u_1,\ldots,u_n)$ est un terme de $T(\Sigma)_{=i+1}$.

Preuve de $F_{j+1} \subseteq_s F_{i+1}$: Par construction de $r(T(\Sigma)_{=j+1})$, nous avons pour tout q de $\mathcal{Q}$, $F_{j+1}(q) = 1 \Rightarrow F_{i+1}(q) = 1$. Donc $F_{j+1} \subseteq_s F_{i+1}$.

Finalement nous avons étendu r sur $T(\Sigma)_{=j+1}$ de sorte que $F_{j+1} \subseteq_s F_{i+1}$. De plus $I_{i+1} =_m I_{j+1}$ donc la propriété énoncée en début de preuve est vraie pour $k = j+1$.

Soit un entier k tel que $k \geq j$. Supposons que le calcul r puisse être étendu sur $T(\Sigma)_{\leq k}$ de sorte que pour tout entier k' tel que $k \geq k' \geq j$:

$$F_{k'} \subseteq_s F_{k'+(i-j)} \quad \text{et} \quad I_{k'} =_m I_{k'+(i-j)}.$$

De la même façon que dans le cas $k = j+1$, nous étendons r sur $T(\Sigma)_{=k+1}$ de sorte que :

$$F_{k+1} \subseteq_s F_{k+1+(i-j)} \quad \text{et} \quad I_{k+1} =_m I_{k+1+(i-j)}.$$

□

Proposition 359. *Il existe un calcul réussi de $\mathcal{A}$ si et seulement s'il existe i et j entiers avec $i < j$ et un calcul partiel réussi de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$.*

Preuve : Supposons qu'il existe r' un calcul réussi de $\mathcal{A}$. Alors il existe un entier l tel que pour tout entier $k \geq l$, $I_k =_m I_l$ puisque la suite $(I_k)_{k\in\mathbb{N}}$ est croissante (propriété (1) de la proposition 356) et car le nombre de multi-ensembles de $\mathcal{M}ul_p$ (applications de $\mathcal{Q}$ dans $\{0,1,\ldots,p\}$) est borné par $(p+1)^{card(\mathcal{Q})}$. De plus il existe deux entiers $i, j \geq l$, $i < j$, tels que $F_j \subseteq_s F_i$ puisque le nombre de multi-ensembles de $\mathcal{M}ul_1$ est borné par $2^{card(\mathcal{Q})}$.

On note r le calcul partiel de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ défini par quelquesoit t terme de $T(\Sigma)_{\leq j}$, $r(t) = r'(t)$. Les frontières et intérieurs de r et r' étant égaux, on a par abus de notation $F_j \subseteq_s F_i$ et $I_j =_m I_i$ pour le calcul partiel r de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$.

De plus r' étant un calcul réussi, il existe un ensemble ω de Ω tel que $r'(T(\Sigma)) = \omega$. $\mathcal{A}$ est simple et $r(T(\Sigma)_{\leq j}) \subseteq \omega$ donc r est un calcul partiel réussi de $\mathcal{A}$.

Supposons maintenant qu'il existe i et j entiers avec $i < j$ et un calcul partiel réussi r de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$. D'après le lemme 358, il existe r calcul de $\mathcal{A}$ tel que pour tout entier $k \geq j$:

$$F_k \subseteq_s F_{k+(i-j)} \quad \text{et} \quad I_k =_m I_{k+(i-j)}.$$

De plus r étant un calcul partiel réussi de $\mathcal{A}$, il existe $\omega \in \Omega$ tel que $r(T(\Sigma)_{\leq j}) = \omega$. De façon évidente $r'(T(\Sigma)) = r(T(\Sigma)_{\leq j}$ donc r est un calcul réussi de $\mathcal{A}$. □

Nous montrons maintenant que la propriété sur les calculs partiels énoncée dans le lemme précédent est décidable (lemme 360) et nous en déduisons le théorème 354.

Lemme 360. *Le problème "il existe i et j entiers avec $i < j$ et un calcul partiel réussi r de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$" est décidable.*

Preuve : Nous notons T l'arbre des calculs de $\mathcal{A}$ c'est-à-dire l'arbre défini par :

- Les étiquettes de T sont des éléments de $\mathcal{M}ul_p \times \mathcal{M}ul_1$ (les éléments de $\mathcal{M}ul_p$ et $\mathcal{M}ul_1$ sont des multi-ensembles de $\mathcal{Q}$);
- La racine de T est étiqueté par (R, R) où R est l'application qui à tout état de $\mathcal{Q}$ associe la valeur 0 ;
- Pour tout calcul partiel r sur $T(\Sigma)_{\leq 0}$, T possède un nœud étiqueté par (I_0, F_0) dont le père est la racine de T ;
- Pour tout entier k non nul et tout calcul partiel r sur $T(\Sigma)_{\leq k}$, T possède un nœud étiqueté par (I_k, F_k) dont le père est le nœud étiqueté par (I_{k-1}, F_{k-1}).

Sur chaque chemin de T, le nombre de nœuds distincts est borné par $(p+1)^{card(\mathcal{Q})} \times 2^{card(\mathcal{Q})}$. Donc pour décider du problème "il existe i et j entiers avec $i < j$ et un calcul partiel réussi r de $\mathcal{A}$ sur $T(\Sigma)_{\leq j}$ tels que $F_j \subseteq_s F_i$ et $I_j =_m I_i$", il suffit de construire l'arbre des calculs jusqu'à la hauteur $(p+1)^{card(\mathcal{Q})}.2^{card(\mathcal{Q})}$. □

Le théorème 354 se déduit directement de la proposition 359 et du lemme 360. Donc le problème du vide est décidable pour les AEAGF simples.

3.3 Contraintes ensemblistes avec tests d'égalité

Dans cette section, nous définissons une extension de la classe (PSC) et nous montrons que le problème de satisfiabilité pour les systèmes de contrainte de cette classe reste décidable en utilisant les AEAGF.

Soit $\mathcal{X}$ un ensemble de variables ensemblistes. La classe des contraintes ensemblistes peut être étendue en considérant des opérateurs imposant des égalités entre les fils des expressions ensemblistes.

Formellement soit $\Delta^f_{i=j}$ une famille d'opérateurs ensemblistes appelés *opérateurs de diagonalisation*, où f est un symbole de fonction d'arité p non nulle et $1 \leq i, j \leq p$. Les expressions ensemblistes que nous considérons maintenant sont construites avec les symboles de fonction habituels et les opérateurs de diagonalisation.

La sémantique associée aux opérateurs de diagonalisation est définie comme suit. Soit $E \in 2^{T(\Sigma)}$,

$$(\Delta^f_{i=j})^{\mathcal{T}_{SC}(\Sigma)}(E) = \{f(t_1, \ldots, t_p) \mid f(t_1, \ldots, t_p) \in E, t_i = t_j\} \ \forall f \in \Sigma_p, 1 \leq i, j \leq p.$$

Pour toute interprétation $\mathcal{I}$ des variables, on a pour toute expression ensembliste e :

$$\mathcal{I}(\Delta^f_{i=j}(e)) = (\Delta^f_{i=j})^{\mathcal{T}_{SC}(\Sigma)}(\chi_{\mathcal{I}}(e))$$

où $\chi_{\mathcal{I}}$ est l'extension de $\mathcal{I}$ à l'ensemble des expressions ensemblistes. Finalement nous définissons la classe (PSC) avec tests d'égalité comme la classe des contraintes positives sans symbole de projection mais avec opérateur de diagonalisation. La suite de cette section est dédiée à la preuve du théorème suivant :

Théorème 361. *Le problème de satisfiabilité pour les systèmes de contraintes de la classe (PSC) avec tests d'égalité est décidable.*

La preuve de ce théorème s'inspire fortement de la preuve de la décidabilité de la satisfiabilité pour les systèmes de contraintes de la classe (PNSC) de R. Gilleron, S. Tison et M. Tommasi [37].

Nous considérons un système de contraintes SC de la classe (PSC) avec tests d'égalité et nous construisons un AEAGF simple et déterministe $\mathcal{A}$ tel que $\mathcal{G}(\mathcal{A}) = \mathcal{S}ol(SC)$. Nous déduisons alors le théorème 361 de la décidabilité du problème du vide pour les AEAGF simples (théorème 354).

Lemme 362. *Soit SC un système de contraintes ensemblistes de la classe (PSC) avec tests d'égalité contenant n variables $X_1, \ldots, X_n$. Alors il existe un AEAGF simple et déterministe $\mathcal{A}$ sur $\{0,1\}^n$ tel que $\mathcal{G}(\mathcal{A}) = \mathcal{S}ol(SC)$.*

Preuve : Dans un premier temps, nous montrons que l'on peut réduire, sans perte de généralité, le problème à une contrainte ensembliste unique et nous associons à cette contrainte un AEAGF $\mathcal{A}$. Nous montrons ensuite que $\mathcal{A}$ est simple et déterministe (lemme 365) puis que $\mathcal{G}(\mathcal{A}) = \mathcal{S}ol(SC)$ (lemmes 366, 367 et 368).

Soit $SC = \bigwedge_{i\in[n]} C_i$ un système de contraintes ensemblistes de la classe (PSC) avec tests d'égalité contenant n variables $X_1, \ldots, X_n$. Une solution de SC satisfait toutes les contraintes C_i. Supposons que, pour tout i de $[n]$, il existe un AEAGF simple et déterministe $\mathcal{A}_i$ tel que $\mathcal{G}(\mathcal{A}_i) = \mathcal{S}ol(C_i)$. Soit i appartenant à $[n]$. Toutes les variables $X_1, \ldots, X_n$ n'apparaissent pas forcément dans C_i mais la preuve de la clôture par projection inverse de la classe des ensembles d'EAG reconnaissables par AEAGF (corollaire 353), nous permet de construire un AEAGF déterministe et simple $\mathcal{A}_i^n$ sur $\{0,1\}^n$ satisfaisant : $\mathcal{G}(\mathcal{A}_i^n)$ est l'ensemble des n-uplets de langages d'arbres $(L_1, \ldots, L_n)$ dont la restriction aux variables de C_i est solution de C_i. Nous déduisons alors de la preuve de la clôture par intersection de la classe des ensembles d'EAG reconnaissables par AEAGF (proposition 351), qu'il existe un AEAGF simple et déterministe $\mathcal{A}$ sur $\{0,1\}^n$ tel que $\mathcal{G}(\mathcal{A}) = \mathcal{S}ol(SC)$. Nous pouvons donc restreindre notre preuve au cas d'une unique contrainte ensembliste SC sur n variables $X_1, \ldots, X_n$.

Nous définissons maintenant l'ensemble $\mathcal{E}(exp)$ des *sous-expressions* d'une expression ensemblistes exp (ensemble des expressions apparaissant dans exp) par :

$$\begin{aligned}
\mathcal{E}(\top) &= \{\top\} \\
\mathcal{E}(\bot) &= \{\bot\} \\
\mathcal{E}(X) &= \{X\},\ \forall X \in \mathcal{X} \\
\mathcal{E}(f(e_1, \ldots, e_p)) &= \{f(e_1, \ldots, e_p))\} \cup (\cup_i \mathcal{E}(e_i)) \\
\mathcal{E}(\complement e) &= \mathcal{E}(e) \\
\mathcal{E}(\cup_i e_i) &= \cup_i \mathcal{E}(e_i) \\
\mathcal{E}(\cap_i e_i) &= \cup_i \mathcal{E}(e_i) \\
\mathcal{E}(\Delta^f_{i=j}(e)) &= \{\Delta^f_{i=j}(e)\} \cup \mathcal{E}(e)
\end{aligned}$$

Si $SC := exp \subseteq exp'$ est une contrainte ensembliste, alors l'ensemble des expressions apparaissant dans SC est $\mathcal{E}(SC) = \mathcal{E}(exp) \cup \mathcal{E}(exp')$.

Remarque 363. *Pour une contrainte ensembliste SC, tout élément de $\mathcal{E}(SC)$ est soit l'un des symboles $\top$ ou $\bot$, soit de la forme $f(e_1, \ldots, e_p)$ ou $\Delta^f_{i=j}(e)$ avec f appartenant à Σ_p et $e, e_1, \ldots, e_p$ expressions ensemblistes.*

Soit φ une *valuation booléenne* c'est-à-dire une application de $\mathcal{E}(SC)$ dans les booléens $\mathcal{B} = \{0, 1\}$ telle que $\varphi(\top) = 1$ et $\varphi(\bot) = 0$. Cette valuation s'étend facilement à n'importe quelle expression ensembliste *exp* (respectivement contrainte ensembliste SC) construite sur les éléments de $\mathcal{E}(exp)$ (respectivement $\mathcal{E}(SC)$) et les opérateurs ensemblistes (les opérateurs $\cup$, $\cap$, $\complement$, $\subseteq$ et $\not\subseteq$ sont respectivement interprétés par $\vee$, $\wedge$, $\neg$, $\Rightarrow$ et $\neg \Rightarrow$) (exemple 364).

Exemple 364. *Soient la signature $\Sigma = \{g/1, a/0\}$ et une variable ensembliste X. Alors l'ensemble des expressions apparaissant dans la contrainte $SC := a \cup f(X) \subseteq X$ est:*

$$\mathcal{E}(SC) = \{a, X, f(X)\}.$$

Donc pour les valuations booléennes φ_1 et φ_2 définies par:

$$\begin{array}{lll} \varphi_1(a) = 1 & \varphi_1(X) = 1 & \varphi_1(f(X)) = 0 \\ \varphi_2(a) = 1 & \varphi_2(X) = 0 & \varphi_2(f(X)) = 1 \end{array}$$

nous avons $\varphi_1(SC) = 0$ et $\varphi_2(SC) = 1$.

Nous considérons l'AEAGF $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, \Omega)$ où :

- $\mathcal{Q} = \{\varphi : \mathcal{E}(SC) \to \mathcal{B} \mid \varphi(SC) = 1\}$,
- $\Omega = 2^{\mathcal{Q}}$,
- Δ est défini comme suit : $f(\varphi_1, \ldots, \varphi_p) \xrightarrow[l]{[c]} \varphi \in \Delta$ où $(\varphi_i)_{i \in [p]}, \varphi$ sont des états de $\mathcal{Q}$, f un symbole de Σ_p, c une contrainte pleine de EC'_p et $l = (l_1, \ldots, l_n)$ une étiquette de $\{0, 1\}^n$ et satisfaisant :

$$(\varphi_i)_{i \in [p]} \text{ satisfait les contraintes d'égalité de } c \tag{8}$$

$$\forall i \in [n],\ \varphi(X_i) = l_i \tag{9}$$

$$\forall e \in \mathcal{E}(SC)\ (e \notin \mathcal{X} \cup \{\top\}) \Rightarrow \left[(\varphi(e) = 1) \Leftrightarrow \left(\begin{array}{c} (e = f(e_1, \ldots, e_p) \text{ et } \forall i \in [p],\ \varphi_i(e_i) = 1) \\ \text{ou} \\ (e = \Delta^f_{i=j}(e') \text{ et } i \approx_c j \text{ et } \varphi(e') = 1) \end{array} \right) \right] \tag{10}$$

□

Nous montrons dans les lemmes suivants que notre construction est correcte, c'est-à-dire que $\mathcal{A}$ est simple et déterministe (lemme 365) et que $\mathcal{L}(\mathcal{A}) = Sol(SC)$ (lemme 368).

Lemme 365. *L'AEAGF $\mathcal{A}$ est simple et déterministe.*

Preuve : Soient $f(\varphi_1, \ldots, \varphi_p) \xrightarrow[l]{[c]} \varphi$ et $f(\varphi_1, \ldots, \varphi_p) \xrightarrow[l]{[c']} \varphi'$ deux règles de $\mathcal{A}$ telles que $(c, \varphi) \neq (c', \varphi')$. Si $c \neq c'$ alors la contrainte $c \wedge c'$ est insatisfiable puisque les contraintes sont pleines. Nous considérons donc le cas où $c = c'$ et $\varphi \neq \varphi'$. Il existe donc une expression e de $\mathcal{E}(SC)$ n'appartenant pas à $\mathcal{X} \cup \{\top\}$ telle que $\varphi(e) \neq \varphi'(e)$ (e n'est pas une variable d'après la condition (9)). Nous considérons le cas où $\varphi(e) = 1$ et $\varphi'(e) = 0$, le cas où $\varphi(e) = 0$ et $\varphi'(e) = 1$ étant symétrique. D'après la condition (10) et la remarque 363, nous avons :

$$(e = f(e_1, \ldots, e_p) \text{ et } \forall i \in [p],\ \varphi(e_i) = 1)$$
$$\text{ou}$$
$$(e = \Delta^f_{i=j}(e') \text{ et } i \approx_c j \text{ et } \varphi(e') = 1).$$

Nous montrons par induction sur la hauteur k de l'expression e que nous obtenons une contradiction.

Base. e est de hauteur 0 donc il existe a appartenant à Σ_0 tel que $e = a$. Nous avons alors $\varphi(a) = \varphi'(a)$ d'après la condition (10). Nous obtenons donc une contradiction.

Induction. Soit un entier k. Supposons que nous obtenons une contradiction pour toute contrainte e' de hauteur k telle que $\varphi(e') = 1$ et $\varphi'(e') = 0$.

Supposons que e soit de hauteur $k+1$. Nous distinguons deux cas suivant la structure de e.

Premier cas : $e = f(e_1, \ldots, e_p)$ et pour tout i de $[p]$, $\varphi(e_i) = 1$.

Par hypothèse d'induction, nous obtenons une contradiction s'il existe un entier i de $[p]$ tel que $\varphi'(e_i) = 0$. Donc pour tout i de $[p]$, $\varphi'(e_i) = 1$. Nous en déduisons que $\varphi'(e) = 1$ par la condition (10). Ceci contredit l'hypothèse $\varphi'(e) = 0$.

Second cas : $e = \Delta^f_{i=j}(e')$, $i \approx_c j$ et $\varphi(e') = 1$.

Nous avons $e = \Delta^f_{i=j}(e')$ et $i \approx_c j$ donc d'après la définition des règles (condition (10)), on a $\varphi'(e') = 0$ puisque $\varphi'(e) = 0$. Donc $\varphi(e') = 1$ et $\varphi'(e') = 0$ avec e' de hauteur k. Nous pouvons alors montrer qu'il existe une expression e'' de $\mathcal{E}(SC)$ de hauteur inférieure ou égale à k telle que soit $\varphi(e'') = 1$ et $\varphi'(e'') = 0$, soit $\varphi(e'') = 0$ et $\varphi'(e'') = 1$. Nous obtenons donc une contradiction par hypothèse d'induction.

Finalement, on ne peut avoir $c = c'$ et $\varphi \neq \varphi'$. Donc $\mathcal{A}$ est déterministe. De plus par construction $\mathcal{A}$ est simple donc $\mathcal{A}$ est simple et déterministe. □

Nous montrons maintenant que $\mathcal{L}(\mathcal{A}) = Sol(SC)$ (lemme 368). Tout d'abord, nous montrons que $\mathcal{L}(\mathcal{A}) \subseteq Sol(SC)$. Soient $(\mathcal{L}_1, \ldots, \mathcal{L}_n)$ un n-uplet de langages d'arbres reconnu par $\mathcal{A}$, g son EAG caractéristique (g est un EAG de $\mathcal{G}^{\Sigma}_{\{0,1\}^n}$ et $\mathcal{L}(g) = (\mathcal{L}_1, \ldots, \mathcal{L}_n)$) et r un calcul réussi de $\mathcal{A}$ sur g. Alors soit $\mathcal{I}_r$ l'interprétation telle que $\mathcal{I}_r(X_i) = \mathcal{L}_i$ pour tout i de $[n]$. Nous

confondons $\mathcal{I}_r$ avec son extension à l'ensemble des expressions ensemblistes. Nous caractérisons l'interprétation $\mathcal{I}_r$ (lemme 366) et nous en déduisons dans la preuve du lemme 368 que $(\mathcal{L}_1, \ldots, \mathcal{L}_n) \subseteq \mathcal{S}ol(SC)$.

Soit $\mathcal{E}'(exp)$ l'ensemble de toutes les expressions ensemblistes apparaissant dans une expression ensembliste exp:

$$\begin{aligned}
\mathcal{E}'(\top) &= \mathcal{E}(\top) \\
\mathcal{E}'(\bot) &= \mathcal{E}(\bot) \\
\mathcal{E}'(X) &= \mathcal{E}(X),\ \forall X \in \mathcal{X} \\
\mathcal{E}'(f(e_1, \ldots, e_p)) &= \mathcal{E}(f(e_1, \ldots, e_p)) \\
\mathcal{E}'(\complement e) &= \mathcal{E}(\complement e) \cup \{\complement e\} \\
\mathcal{E}'(\cup_i e_i) &= \mathcal{E}(\cup_i e_i) \cup \{\cup_i e_i\} \\
\mathcal{E}'(\cap_i e_i) &= \mathcal{E}(\cap_i e_i) \cup \{\cap_i e_i\} \\
\mathcal{E}'(\Delta^f_{i=j}(e)) &= \mathcal{E}(\Delta^f_{i=j}(e))
\end{aligned}$$

et $\mathcal{E}'(SC)$ l'ensemble de toutes les expressions ensemblistes apparaissant dans SC.

Lemme 366. $\forall e \in \mathcal{E}'(SC), \forall t \in T(\Sigma),\ (t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1)$.

Preuve : Nous montrons la propriété

$$\forall e \in \mathcal{E}'(SC), \forall t \in T(\Sigma), (t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1) \tag{11}$$

par induction sur la hauteur de e.

Base. Soient t un terme de $T(\Sigma)$ et e une expression ensembliste de $\mathcal{E}'(SC)$ de hauteur 0. Alors e est soit une constante a, soit l'un des symboles $\top$ ou $\bot$, soit une variable X_i.

Tout d'abord, il existe un symbole de fonction f d'arité p, p entier, et $t_1, \ldots, t_p$ termes de $T(\Sigma)$ tels que $t = f(t_1, \ldots, t_p)$. Puisque r est un calcul de $\mathcal{A}$ sur g, il existe une règle $f(r(t_1), \ldots, r(t_p)) \xrightarrow[g(t)]{[c]} r(t)$ appartenant à Δ. Par définition des règles de $\mathcal{A}$, nous avons:

$$(r(t)(e) = 1) \Leftrightarrow \left(\begin{array}{c} (e = f(e_1, \ldots, e_p) \text{ et } \forall i \in [p],\ \varphi(e_i) = 1) \\ \text{ou} \\ (e = \Delta^f_{i=j}(e') \text{ et } i \approx_c j \text{ et } \varphi(e') = 1) \end{array} \right).$$

Si $e = a$, nous avons $\mathcal{I}_r(e) = \{a\}$ donc $(t \in \mathcal{I}_r(e)) \Leftrightarrow (t = a)$. Or par définition des règles, $(r(t)(e) = 1) \Leftrightarrow (a = f)$. Finalement $(t \in \mathcal{I}_r(a)) \Leftrightarrow (r(t)(a) = 1)$.

Si $e = \top$ alors par définitions $\mathcal{I}_r(e) = T(\Sigma)$ et pout tout état φ de $\mathcal{Q}$, $\varphi(e) = 1$. Donc $(t \in \mathcal{I}_r(\top)) \Leftrightarrow (r(t)(\top) = 1)$.

Si $e = \bot$ alors par définition $\mathcal{I}_r(e) = \emptyset$ et pour tout état φ de $\mathcal{X}$, $\varphi(e) = 0$. Donc $(t \in \mathcal{I}_r(\bot)) \Leftrightarrow (r(t)(\bot) = 1)$.

Si e est une variable X_i de SC alors par définition $\mathcal{I}_r(X_i) = \mathcal{L}_i$. Or par la condition (9), nous avons $r(t)(X_i) = g(t)_i$. Nous en déduisons que $(t \in \mathcal{I}_r(X_i)) \Leftrightarrow (r(t)(X_i) = 1)$ puisque $\mathcal{L}_i = \{t \in T(\Sigma) \mid g(t)_i = 1\}$.

Finalement pour toute expression de $\mathcal{E}'(SC)$ de hauteur 0, nous avons:

$$\forall t \in T(\Sigma), (t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1).$$

Induction. Soi k entier. Supposons que la propriété (11) soit vraie pour toutes les expressions de $\mathcal{E}'(SC)$ de hauteur inférieure ou égale à k. Soient e une expression de $\mathcal{E}'(SC)$ de hauteur $k+1$ et t un terme de $T(\Sigma)$.

Tout d'abord, il existe f symbole de Σ d'arité p non nulle et $t_1, \ldots, t_p$ termes de $T(\Sigma)$ tels que $t = f(t_1, \ldots, t_p)$. Puisque r est un calcul de $\mathcal{A}$ sur g, il existe une contrainte pleine c de EC'_p telle que $f(r(t_1), \ldots, r(t_p)) \xrightarrow[g(t)]{[c]} r(t)$ appartienne à Δ et $f(t_1, \ldots, t_p) \vDash c$.

Nous distinguons maintenant plusieurs cas suivant la structure de e.

e **est de la forme** $f(e_1, \ldots, e_p)$. Nous avons :

$$\begin{aligned} t \in \mathcal{I}_r(e) &\Leftrightarrow t \in f(\mathcal{I}_r(e_1), \ldots, \mathcal{I}_r(e_p)) \\ &\Leftrightarrow t \in \{f(t_1, \ldots, t_p) \mid \forall i \in [p], t_i \in \mathcal{I}_r(e_i)\} \\ &\Leftrightarrow t \in \{f(t_1, \ldots, t_p) \mid \forall i \in [p], r(t_i)(e_i) = 1\} \quad \text{par induction} \end{aligned}$$

Supposons que t appartienne à $\mathcal{I}_r(e)$. Donc pour tout i de $[p], r(t_i)(e_i) = 1$. Or d'après la condition (10),

$$(\forall i \in [p], (r(t_i)(e_i) = 1)) \Rightarrow (r(t)(e) = 1)$$

puisque $e = f(e_1, \ldots, e_p)$. Donc $r(t)(e) = 1$.
Supposons maintenant que $r(t)(e) = 1$. Nous déduisons de la condition (10), que pour tout i de $[p], r(t_i)(e_i) = 1$ puisque $e = f(e_1, \ldots, e_p)$. Donc t appartient à $\mathcal{I}_r(e)$. Finalement $(t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1)$.

e **est de la forme** $\Delta^f_{i=j}(e')$. Nous avons :

$$\begin{aligned} t \in \mathcal{I}_r(e) &\Leftrightarrow t \in \{f(t_1, \ldots, t_p) \in \mathcal{I}_r(e') \mid t_i = t_j\} \\ &\Leftrightarrow t \in \{f(t_1, \ldots, t_p) \mid r(f(t_1, \ldots, t_p))(e') = 1, t_i = t_j\} \quad \text{par induction} \end{aligned}$$

Supposons que t appartienne à $\mathcal{I}_r(e)$. Alors $t_i = t_j$ donc $i \approx_c j$. D'après la condition (10),

$$(r(t)(e') = 1) \Rightarrow (r(t)(e) = 1)$$

puisque $e = \Delta^f_{i=j}(e')$ et $i \approx_c j$. Donc $r(t)(e) = 1$.
Supposons maintenant que $r(t)(e) = 1$. Nous déduisons de la condition (10) que $i \approx_c j$ et $r(t)(e') = 1$ puisque $e = \Delta^f_{i=j}(e')$. Donc $t_i = t_j$, d'où t appartient à $\mathcal{I}_r(e)$. Finalement $(t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1)$.

e **est de la forme** $e_1 \cup e_2$. Nous avons :

$$\begin{aligned} t \in \mathcal{I}_r(e_1 \cup e_2) &\Leftrightarrow t \in \mathcal{I}_r(e_1) \cup \mathcal{I}_r(e_2) \\ &\Leftrightarrow (r(t)(e_1) = 1 \text{ et } r(t)(e_2) = 1) \quad \text{par induction} \\ &\Leftrightarrow r(t)(e_1 \cup e_2) = 1. \end{aligned}$$

e **est de la forme** $e_1 \cap e_2$ **ou** $e = \complement e$. Nous montrons de la même manière que pour e de la forme $e_1 \cup e_2$ que $(t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1)$.

Finalement la propriété (11) est vraie pour toute expression de $\mathcal{E}'(SC)$ de hauteur inférieure ou égale à $k+1$ ce qui termine l'induction.

Finalement $\forall e \in \mathcal{E}'(SC), \forall t \in T(\Sigma), (t \in \mathcal{I}_r(e)) \Leftrightarrow (r(t)(e) = 1)$. □

Nous effectuons maintenant l'opération inverse à la précédente, c'est-à-dire que nous associons un calcul de $\mathcal{A}$ à une solution de la contrainte SC (lemme 367). Nous en déduisons dans la preuve du lemme 368 que $Sol(SC) \subseteq \mathcal{L}(\mathcal{A})$.

Lemme 367. *Soient $\mathcal{I}$ une solution de la contrainte SC et g l'EAG caractéristique de cette solution ($\mathcal{L}(g) = (\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n))$. Alors l'application r de $T(\Sigma)$ dans Q définie par $r(t) = \varphi$ avec*

$$\forall e \in \mathcal{E}(SC), (\varphi(e) = 1) \Leftrightarrow (t \in \mathcal{I}(e))$$

est un calcul de $\mathcal{A}$ sur g.

Preuve : Soient $\mathcal{I}$ une solution de la contrainte SC, g l'AEG caractéristique de cette solution et r l'application définie par :

$$\begin{array}{rcl} r : T(\Sigma) & \rightarrow & Q \\ t & \mapsto & \varphi \text{ tel que } \forall e \in \mathcal{E}(SC), (\varphi(e) = 1) \Leftrightarrow (t \in \mathcal{I}(e)). \end{array}$$

Nous allons montrer que r est un calcul de $\mathcal{A}$ sur g c'est-à-dire que r est compatible avec les règles de Δ. Dans ce but, nous considérons tout d'abord les équivalences suivantes :

$$\begin{array}{lrll} \forall t \in T(\Sigma), \forall i \in [n], & r(t)(X_i) = 1 & \Leftrightarrow & t \in \mathcal{I}(X_i) \quad \text{par définition de } r \\ & & \Leftrightarrow & g(t)_i = 1 \quad \text{par définition de } \mathcal{L}(g). \end{array}$$

Nous en déduisons que pour tout terme t de $T(\Sigma)$ et tout i de $[n]$, $r(t)(X_i) = g(t)_i$. Donc la condition (9) est satisfaite.

Soit t un terme $T(\Sigma)$. Montrons par raisonnement sur la structure de t que l'application r satisfait la propriété suivante :

Si $t = f(t_1, \ldots, t_p)$ alors il existe une contrainte pleine c de EC'_p telle que la règle $f(r(t_1), \ldots, r(t_p)) \xrightarrow[g(t)]{[c]} r(t)$ appartienne à Δ et $f(t_1, \ldots, t_p) \vDash c$.

Constantes. Il existe une constante a de Σ telle que $t = a$. Par définition de r, nous avons :

$$\forall e \in \mathcal{E}(SC), (r(a)(e) = 1) \Leftrightarrow (a \in \mathcal{I}(e)).$$

Or nous pouvons montrer que pour toute expression ensembliste e de $\mathcal{E}(SC)$ différente d'une variable, du symbole $\top$ et de a, nous avons a n'appartient pas à $\mathcal{I}(e)$. Nous déduisons alors de la remarque 363 que :

$$\forall e \in \mathcal{E}(SC)\ (e \notin \mathcal{X} \cup \{\top\}) \Rightarrow (r(a)(e) = 1 \Leftrightarrow e = a).$$

Donc la règle $a \xrightarrow{g(a)} r(a)$ appartient à Δ puisque $r(a)$ satisfait les conditions (9) et (10) de la définition de Δ.

Cas général. Supposons que t soit de hauteur non nulle. Alors il existe un symbole f de Σ d'arité p non nulle et $t_1, \ldots, t_p$ termes de $T(\Sigma)$ tels que $t = f(t_1, \ldots, t_p)$.

Soit c la contrainte pleine de EC'_p telle que $f(t_1, \ldots, t_n) \vDash c$. Alors $(r(t_i))_{i \in [p]}$ satisfait les égalités de c. De plus pour toute expression e de $\mathcal{E}(SC) \setminus (\mathcal{X} \cup \{\top\})$, nous avons par

la remarque 363 :

$$\begin{array}{rcl}(r(t)(e)=1) & \Leftrightarrow & (t \in \mathcal{I}(e)) \\ & \Leftrightarrow & \left(\begin{array}{c}(e=f(e_1,\ldots,e_p) \text{ et } \forall i \in [p], t_i \in \mathcal{I}(e_i)) \\ \text{ou} \\ (e=\Delta^f_{i=j}(e') \text{ et } t_i=t_j \text{ et } t \in \mathcal{I}(e'))\end{array}\right) \\ & \Leftrightarrow & \left(\begin{array}{c}(e=f(e_1,\ldots,e_p) \text{ et } \forall i \in [p], r(t_i)(e_i)=1) \\ \text{ou} \\ (e=\Delta^f_{i=j}(e') \text{ et } i \approx_c j \text{ et } r(t)(e')=1)\end{array}\right).\end{array}$$

Nous en déduisons que la règle $f(r(t_1),\ldots,r(t_p)) \xrightarrow[g(t)]{[c]} r(t)$ appartient à Δ puisque $r(t)$ et c satisfont les conditions (8), (9) et (10) de la définition de Δ.

Donc r est compatible avec les règles de Δ. Finalement r est un calcul de $\mathcal{A}$ sur g. □

Lemme 368. $\mathcal{G}(\mathcal{A}) = \mathcal{S}ol(SC)$.

Preuve : Nous avons vu dans le début de la preuve du lemme 362 que l'on peut réduire sans perte de généralité le problème à une contrainte ensembliste unique. Nous supposons donc que $SC := exp \subseteq exp'$.

Montrons tout d'abord que $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{S}ol(SC)$. Nous considérons $(\mathcal{L}_1,\ldots,\mathcal{L}_n)$ un n-uplet reconnu par $\mathcal{A}$. Soient r un calcul de $\mathcal{A}$ sur g, l'EAG caractéristique de $(\mathcal{L}_1,\ldots,\mathcal{L}_n)$, et $\mathcal{I}_r$ l'interprétation des variables telle que $\mathcal{I}_r(X_i) = \mathcal{L}_i$ pour tout i de $[n]$. Soit t un terme de $T(\Sigma)$ appartenant à $\mathcal{I}_r(exp)$. Par le lemme 366, $r(t)(exp) = 1$ puisque exp est une expression de $\mathcal{E}'(SC)$. D'après la définition de $\mathcal{Q}$, chaque état φ de $r(T(\Sigma))$ satisfait $\varphi(SC) = 1$, donc $r(t)(exp') = 1$. Donc t appartient à $\mathcal{I}_r(exp')$ puisque exp' appartient à $\mathcal{E}'(SC)$ (lemme 366). Nous en déduisons que $\mathcal{I}_r$ satisfait la contrainte SC. Donc $(\mathcal{L}_1,\ldots,\mathcal{L}_n)$ est solution de SC.

Montrons maintenant que $\mathcal{S}ol(SC) \subseteq \mathcal{L}(\mathcal{A})$. Nous considérons $(\mathcal{L}_1,\ldots,\mathcal{L}_n)$ une solution de SC. Soit $\mathcal{I}$ l'interprétation des variables associée à cette solution ($\mathcal{I}(X_i) = \mathcal{L}_i$ pour tout i de $[n]$). Par le lemme 367, l'application r de $T(\Sigma)$ dans $\mathcal{Q}$ définie par $r(t) = \varphi$ avec

$$\forall e \in \mathcal{E}(SC)(\varphi(e) = 1) \Leftrightarrow (t \in \mathcal{I}(e))$$

est un calcul de $\mathcal{A}$ sur g, l'EAG caractéristique de $(\mathcal{L}_1,\ldots,\mathcal{L}_n)$. Comme l'automate $\mathcal{A}$ est simple (lemme 365), $(\mathcal{L}_1,\ldots,\mathcal{L}_n)$ est reconnu par $\mathcal{A}$.

Donc $\mathcal{L}(\mathcal{A}) = \mathcal{S}ol(SC)$, or $\mathcal{G}(\mathcal{A}) = \mathcal{L}(\mathcal{A})$ d'où $\mathcal{G}(\mathcal{A}) = \mathcal{S}ol(SC)$. □

La preuve du lemme 362 est maintenant terminée. Nous déduisons facilement de ce lemme et du théorème 354, la décidabilité du problème de satisfiabilité pour les systèmes de contraintes de la classe (PSC) avec tests d'égalité (théorème 361).

Pour terminer ce chapitre, nous rappelons le résultat de W. Charatonik et L. Pacholski généralisant notre résultat :

Théorème 369 (W. Charatonik et L. Pacholski [17]). *Le problème de satisfiabilité pour les systèmes de contraintes positives et négatives avec tests d'égalité est décidable.*

Chapitre 4

Topologie - Expressivité

Dans ce chapitre, nous étudions les propriétés topologiques de l'ensemble des solutions des systèmes de certaines classes de contraintes ensemblistes (sections 4.1 et 4.2). Par la suite, cette étude nous permet de comparer le pouvoir d'expression en terme d'ensembles de solutions de certaines classes de contraintes (section 4.3).

Soient Σ la signature et $\mathcal{X}$ l'ensemble de variables ensemblistes sur lesquels les contraintes ensemblistes de ce chapitre sont définies.

4.1 Etude de la classe (PSC)

M. Tommasi dans [87] montre en utilisant les automates d'arbres ensemblistes que l'ensemble des solutions d'un système de contraintes ensemblistes de la classe (PSC) est compact, donc fermé. Nous donnons dans cette section une preuve directe du fait que cet ensemble de solutions est fermé.

Nous définissons pour cela un ensemble d'applications continues (lemme 370) et nous montrons que l'ensemble des solutions d'un système de contraintes ensemblistes positives est l'image inverse du singleton constitué de l'ensemble vide par une composée de ces fonctions (proposition 372).

Tout d'abord, nous associons une application à chacun des symboles de $\Sigma \cup \{\cup, \cap, \complement\}$:

$$\forall f \in \Sigma_p, F_f: \begin{array}{ccc} (2^{T(\Sigma)})^p & \to & 2^{T(\Sigma)} \\ (X_1, \ldots, X_p) & \to & f(X_1, \ldots, X_p) \end{array} \qquad F_\cup: \begin{array}{ccc} (2^{T(\Sigma)})^2 & \to & 2^{T(\Sigma)} \\ (X_1, X_2) & \to & X_1 \cup X_2 \end{array}$$

$$F_\complement: \begin{array}{ccc} 2^{T(\Sigma)} & \to & 2^{T(\Sigma)} \\ X & \to & T(\Sigma) \setminus X \end{array} \qquad F_\cap: \begin{array}{ccc} (2^{T(\Sigma)})^2 & \to & 2^{T(\Sigma)} \\ (X_1, X_2) & \to & X_1 \cap X_2 \end{array}$$

et nous définissons pour tout entier non nul et tout i de $[n]$ une application F_i^n appelée projection :

$$F_i^n: \begin{array}{ccc} (2^{T(\Sigma)})^n & \to & 2^{T(\Sigma)} \\ (X_1, \ldots, X_n) & \to & X_i \end{array}$$

Lemme 370. *Les applications $F_i^n, F_\cup, F_\cap, F_\complement$ et F_f sont continues pour tout i de $[n]$, n strictement positif, et pour tout symbole f de Σ.*

Preuve : Nous prouvons ici uniquement que $F_\cap$ est continue, les autres preuves se faisant de façon similaire. Nous prouvons que pour tout ouvert O de $2^{T(\Sigma)}$, $F_\cap^{-1}(O)$ est un ouvert de $\left(2^{T(\Sigma)}\right)^2$.

Soient O un ouvert de $2^{T(\Sigma)}$ et x un élément de $F_\cap^{-1}(O)$. Alors $F_\cap(x)$ appartient à O, donc il existe $r > 0$ tel que $\mathcal{B}_{d_1}(F_\cap(x), r) \subseteq O$ puisque O est un ouvert. Montrons que $\mathcal{B}_{d_2}(x, r) \subseteq F_\cap^{-1}(O)$. Soit y un élément de $\mathcal{B}_{d_2}(x, r)$.

Dans le cas où $d_2(x, y) = 0$, nous avons $x = y$ donc $F_\cap(y) = F_\cap(x)$, d'où $F_\cap(y) \in \mathcal{B}_{d_1}(F_\cap(x), r)$.

Etudions maintenant le cas où $d_2(x, y) > 0$. Il existe X, X', Y, Y' éléments de $2^{T(\Sigma)}$ tels que $x = (X, X')$ et $y = (Y, Y')$. Soient

$$\begin{aligned} n &= min(\{Haut(t) \mid t \in [(X \Delta Y) \cup (X' \Delta Y')]\}) \\ m &= min(\{Haut(t) \mid t \in [(X \cap X') \Delta (Y \cap Y')]\}) \end{aligned}$$

Alors $d_2(x, y) = 2^{-n}$ et $d_1(F_\cap(x), F_\cap(y)) = 2^{-m}$.

Soit t un terme de $[(X \cap X') \Delta (Y \cap Y')]$. Alors si t appartient à $(X \cap X')$, t appartient à X et X' et t n'appartient pas à la fois à Y et à Y'. Nous en déduisons que t est un terme de $[(X \Delta Y) \cup (X' \Delta Y')]$. Nous obtenons la même conclusion si nous supposons que t appartient à $(Y \cap Y')$, donc :

$$[(X \cap X') \Delta (Y \cap Y')] \subseteq [(X \Delta Y) \cup (X' \Delta Y')].$$

Nous en déduisons que $n \leq m$ ce qui nous mène aux implications suivantes :

$$\begin{aligned} n \leq m &\Rightarrow 2^{-m} \leq 2^{-n} \\ &\Rightarrow d_1(F_\cap(x), F_\cap(y)) \leq d_2(x, y) \\ &\Rightarrow d_1(F_\cap(x), F_\cap(y)) < r && \text{car } d_2(x, y) < r \\ &\Rightarrow F_\cap(y) \in \mathcal{B}_{d_1}(F_\cap(x), r) \\ &\Rightarrow F_\cap(y) \in O && \text{car } \mathcal{B}_{d_1}(F_\cap(x), r) \subseteq O \\ &\Rightarrow y \in F_\cap^{-1}(O). \end{aligned}$$

Finalement $\mathcal{B}_{d_2}(x, r) \subseteq F_\cap^{-1}(O)$ et donc $F_\cap^{-1}(O)$ est un ouvert de $\left(2^{T(\Sigma)}\right)^2$. Donc l'image inverse par $F_\cap$ de tout ouvert de $2^{T(\Sigma)}$ est un ouvert de $\left(2^{T(\Sigma)}\right)^2$ donc l'application $F_\cap$ est continue. □

Nous montrons maintenant que l'on peut associer à toute expression ensembliste e sur n variables $X_1, \ldots, X_n$, une application $M_e : \left(2^{T(\Sigma)}\right)^n \to 2^{T(\Sigma)}$ telle que pour toute interprétation $\mathcal{I}$ des variables de e, on ait $M_e(\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n)) = \mathcal{I}(e)$ (lemme 371).

Soit e une expression ensembliste sur n variables $X_1, \ldots, X_n$. On associe à e, l'application $M_e : \left(2^{T(\Sigma)}\right)^n \to 2^{T(\Sigma)}$ définie par :

- Si $e = X_i$, i appartenant à $[n]$, alors $M_e = F_i^n$,
- Si $e = f(e_1, \ldots, e_p)$ avec f symbole de Σ_p et $e_1, \ldots, e_p$ expressions ensemblistes, alors $M_e = F_f \circ (M_{e_1}, \ldots, M_{e_p})$,
- Si $e = e_1 \cup e_2$ avec e_1, e_2 expressions ensemblistes, alors $M_e = F_\cup \circ (M_{e_1}, M_{e_2})$,
- Si $e = e_1 \cap e_2$ avec e_1, e_2 expressions ensemblistes, alors $M_e = F_\cap \circ (M_{e_1}, M_{e_2})$, et

- Si $e = \complement e_1$ avec e_1 expression ensembliste, alors $M_e = F_\complement \circ M_{e_1}$.

Lemme 371. *Soit e une expression ensembliste sur n variables $X_1, \ldots, X_n$. Alors pour toute interprétation $\mathcal{I}$ des variables de e, nous avons $M_e(\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n)) = \mathcal{I}(e)$.*

La preuve de cette proposition se fait par induction sur la structure de l'expression ensembliste e. Nous pouvons maintenant prouver le résultat que nous annoncions en début de cette section.

Proposition 372. *L'ensemble des solutions de tout système de contraintes ensemblistes de la classe (PSC) est fermé.*

Preuve : Soit $SC = \bigwedge_{i=1}^{k} e_i \subseteq e'_i$ un système de contraintes ensemblistes de la classe (PSC). Soient $X_1, \ldots, X_n$ les variables apparaissant dans SC. Pour tout entier i de $[k]$, nous définissons une application ϕ_i comme suit :

$$\begin{array}{rccl} \phi_i: & \left(2^{T(\Sigma)}\right)^n & \to & 2^{T(\Sigma)} \\ & (Y_1, \ldots, Y_n) & \to & M_{e_i}(Y_1, \ldots, Y_n) \setminus M_{e'_i}(Y_1, \ldots, Y_n) \end{array}$$

Soit i appartenant à $[k]$. Alors :

$$\begin{array}{rcll} Sol(e_i \subseteq e'_i) & = & \{\text{interprétation } \mathcal{I} \mid \mathcal{I}(e_i) \setminus \mathcal{I}(e'_i) = \emptyset\} & \\ & = & \{\text{interprétation } \mathcal{I} \mid \phi_i(\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n)) = \emptyset\} & \text{(proposition 371)} \\ & = & \{(Y_1, \ldots, Y_n) \in \left(2^{T(\Sigma)}\right)^n \mid \phi_i(Y_1, \ldots, Y_n) = \emptyset\} & \\ & = & \phi_i^{-1}(\{\emptyset\}). & \end{array}$$

Remarquons que nous utilisons la notation $\{\emptyset\}$ pour désigner l'ensemble vide comme élément de $\left(2^{T(\Sigma)}\right)^n$ (à différencier de l'ensemble vide partie de $\left(2^{T(\Sigma)}\right)^n$).

Puisque toute suite dans $\{\emptyset\}$ converge, $\{\emptyset\}$ est un compact de $\left(2^{T(\Sigma)}\right)^n$. Donc $\{\emptyset\}$ est un fermé.

De plus, d'après les définitions de M_{e_i} et de $M_{e'_i}$, ϕ_i est une composée des applications $F_\cup, F_\cap, F_\complement, F_f$ et F_i^n. Donc puisque ces applications sont continues (lemme 370), ϕ_i est continue. Nous en déduisons que $\phi_i^{-1}(\{\emptyset\})$ est un fermé de $\left(2^{T(\Sigma)}\right)^n$ car image réciproque d'un fermé de $2^{T(\Sigma)}$.

Finalement $Sol(SC) = \bigcap_{i=1}^{k} \phi_i^{-1}(\{\emptyset\})$ est un fermé de $(2^{T(\Sigma)})^n$ car intersection de fermés de $\left(2^{T(\Sigma)}\right)^n$. □

Remarquons que ce résultat reste valable pour les systèmes de contraintes possédant un nombre dénombrable de contraintes ensemblistes positives.

4.2 Etude de la classe (PNSC)

Proposition 373. *Soit SC un système de contraintes ensemblistes de la classe (PNSC) sur n variables. Alors $Sol(SC)$ est l'intersection d'un fermé et d'un ouvert.*

Preuve : Soit $SC = \bigwedge_{i=1}^{k} e_i \subseteq e'_i \wedge \bigwedge_{i=k+1}^{l} e_i \not\subseteq e'_i$ un système de contraintes ensemblistes de la classe (PNSC). Nous avons :

$$Sol(SC) = \left(\bigcap_{i=1}^{k} Sol(e_i \subseteq e'_i) \right) \cap \left(\bigcap_{i=k+1}^{l} Sol(e_i \not\subseteq e'_i) \right).$$

Le système $\bigwedge_{i\in[k]} e_i \subseteq e'_i$ est un système de contraintes ensemblistes de la classe (PSC) donc son ensemble de solutions est un fermé de $\left(2^{T(\Sigma)}\right)^n$ (proposition 372).

Montrons maintenant que l'ensemble des solutions du système $\bigwedge_{i=k+1}^{l} e_i \not\subseteq e'_i$ est ouvert. Soit i un entier tel que $k+1 \leq i \leq l$. Nous avons $Sol(e_i \not\subseteq e'_i) = \complement_{\left(2^{T(\Sigma)}\right)^n} Sol(e_i \subseteq e'_i)$. Or $Sol(e_i \subseteq e'_i)$ est un fermé de $\left(2^{T(\Sigma)}\right)^n$ puisqu'il est l'ensemble de solutions d'une contrainte ensembliste de la classe (PSC) (proposition 372). Nous en déduisons que $Sol(e_i \not\subseteq e'_i)$ est un ouvert de $\left(2^{T(\Sigma)}\right)^n$ car complément d'un fermé. Or toute intersection finie d'ouverts est un ouvert donc $\bigcap_{i=k+1}^{l} Sol(e_i \not\subseteq e'_i)$ est un ouvert de $\left(2^{T(\Sigma)}\right)^n$. Finalement l'ensemble $Sol(SC)$ est l'intersection d'un fermé et d'un ouvert. □

4.3 Expressivité

Nous comparons maintenant l'expressivité de certaines classes de contraintes ensemblistes relativement aux ensembles de solutions (4.3.2). Nous donnons tout d'abord (section 4.3.1) les définitions relatives à l'expressivité.

4.3.1 Définitions

Soient (S1) et (S2) deux classes de contraintes ensemblistes. La classe (S1) est dite *plus expressive* que la classe (S2) en terme d'ensembles de solutions si et seulement si pour tout système $SC2$ de contraintes de la classe (S2), il existe un système $SC1$ de contraintes de la classe (S1) tel que $Sol(SC1) = Sol(SC2)$. (S1) est *strictement plus expressive* que (S2) si (S1) est plus expressive que (S2) et (S2) n'est pas plus expressive que (S1).

Finalement, les classes (S1) et (S2) sont dites *équivalentes* si et seulement si (S1) est plus expressive que (S2) et (S2) est plus expressive que (S1).

4.3.2 Etude

Pour notre étude, nous distinguons trois cas suivant la structure de la signature Σ :

- **Constante :** Σ ne contient que des constantes.
- **Unaire :** Σ contient au moins une constante et un symbole de fonction unaire mais aucun symbole de fonction d'arité au moins deux.
- **Général :** Σ contient au moins une constante et un symbole de fonction d'arité au moins deux.

Dans tous les cas, la classe (PNSC) (respectivement (PNSCP)) est plus expressive que la classe (PSC) (respectivement (PSCP)) puisque toute contrainte de la classe (PSC) (respectivement (PSCP)) est une contrainte de la classe (PNSC) (respectivement (PNSCP)).

Constante

Supposons que Σ ne contienne que des constantes. Alors l'ensemble des symboles de projection et vide donc les classes (PSC) et (PSCP) sont égales (c'est-à-dire qu'elles contiennent les mêmes contraintes ensemblistes) et il en est de même pour les classes (PNSC) et (PNSCP).

Si de plus Σ ne contient qu'une constante a, nous avons pour toute contrainte ensembliste négative $e_1 \not\subseteq e_2$ en terme d'ensemble de solutions :

$$\begin{aligned} e_1 \not\subseteq e_2 &\Leftrightarrow e_1 \cap \complement e_2 \not\subseteq \bot \\ &\Leftrightarrow e_1 \cap \complement e_2 = a. \end{aligned}$$

Donc si Σ ne contient qu'une constante, les classes (PSC), (PSCP), (PNSC) et (PNSCP) sont équivalentes.

Si Σ contient au moins deux constantes. Soit $\Sigma = \{a/0, b/0\}$. Le système SC, défini ci-dessous sur la variable ensembliste X, de la classe (PNSC) a pour ensemble de solutions $Sol(SC) = \{\{a\}, \{b\}\}$.

$$\begin{cases} X & \not\subseteq & \bot \\ a \cup b & \not\subseteq & X \end{cases}$$

Nous conjecturons qu'il n'existe pas de système de la classe (PSC) sur la seule variable X ayant même ensemble de solutions que le système SC donc que la classe (PNSC) est strictement plus expressive que la classe (PSC).

Unaire

Supposons que Σ contienne au moins une constante et un symbole de fonction unaire mais aucun symbole de fonction d'arité au moins deux.

M. Tommasi dans [87] montre qu'avec une telle signature la classe (PNSC) est strictement plus expressive que la classe (PSC).

Cette propriété se montre facilement par l'absurde. Considérons la signature $\Sigma = \{f/1, a/0\}$ et la contrainte négative $SC := X \not\subseteq \emptyset$ où X est une variable ensembliste. Alors

$$Sol(SC) = \{\mathcal{L} \in 2^{T(\Sigma)} \mid \mathcal{L} \neq \emptyset\}.$$

Soit la suite $(X_p)_{p\in\mathbb{N}}$ dans $Sol(SC)$ définie par

$$\forall p \in \mathbb{N}, X_p = \{f^p(a)\}.$$

La suite $(X_p)_{p\in\mathbb{N}}$ converge vers $\emptyset$ car pour tout entier p, $d_1(X_p, \emptyset) = 2^{-p}$. Supposons que $Sol(SC)$ est solution d'un système de contraintes de la classe (PSC). Alors $Sol(SC)$ est un fermé d'après la proposition 372. Nous en déduisons que $\emptyset$ appartient à $Sol(SC)$ (théorème 29). Or $\emptyset$ n'est pas solution de SC donc $Sol(SC)$ n'est pas solution d'un système de la classe (PSC). Finalement la classe (PNSC) est strictement plus expressive que la classe (PSC).

Ce même résultat est obtenu par A. Aiken, D. Kozen et E. Wimmers dans [2] en utilisant un théorème de compacité.

Général

Supposons que Σ contienne au moins une constante et un symbole de fonction d'arité au moins deux.

Nous pouvons montrer que l'expressivité de la classe (PSC) est strictement augmentée par les contraintes ensemblistes de la forme $X \subseteq f_i^{-1}(Y)$ où f est un symbole d'arité au moins deux (projection positive). Nous en déduisons que la classe (PSCP) est strictement plus expressive que la classe (PSC).

Montrons maintenant que la classe (PSCP) est strictement plus expressive que la classe (PNSC) (proposition 375). Tout d'abord, nous rappelons le résultat suivant de L. Bachmair, H. Ganzinger et U. Waldmann [8].

Proposition 374. *Si Σ contient au moins une constante et au moins un symbole de fonction d'arité au moins deux, alors la classe (PSCP) est plus expressive que la classe (PNSC).*

Preuve : Soit $e_1 \not\subseteq e_2$ une contrainte négative sans symbole de projection. Cette contrainte est de façon évidente équivalente en terme d'ensembles de solutions à la contrainte $e_1 \cap \complement e_2 \not\subseteq \emptyset$.

D'après les hypothèses, il existe dans la signature Σ une constante a et un symbole de fonction f d'arité p supérieure ou égale à deux. Alors la contrainte $e_1 \cap \complement e_2 \not\subseteq \bot$ est équivalente à la contrainte $a \subseteq f_p^{-1}(f(e_1 \cap \complement e_2, a, \ldots, a))$ en terme d'ensembles de solutions puisque :

$$\begin{aligned} e_1 \cap \complement e_2 \not\subseteq \bot &\Leftrightarrow \exists t \in T(\Sigma), t \in e_1 \cap \complement e_2 \\ &\Leftrightarrow \exists t \in T(\Sigma), f(t, a, \ldots, a) \in f(e_1 \cap \complement e_2, a, \ldots, a) \\ &\Leftrightarrow a \subseteq f_p^{-1}(f(e_1 \cap \complement e_2, a, \ldots, a)). \end{aligned}$$

Donc toute contrainte ensembliste négative sans symbole de projection est équivalente en terme d'ensembles de solutions à une contrainte ensembliste positive avec projection. Nous en déduisons que la classe (PSCP) est plus expressive que la classe (PNSC). □

Proposition 375. *Si Σ contient au moins une constante et au moins un symbole de fonction d'arité au moins deux, alors la classe (PSCP) est strictement plus expressive que la classe (PNSC).*

Preuve : Nous montrons sur un exemple qu'il existe un système SC de contraintes ensemblistes de la classe (PSCP) tel que $Sol(SC)$ ne soit solution qu'aucun système de contraintes de la classe (PNSC).

Soient la signature $\Sigma = \{f/2, a/0\}$, deux variables ensemblistes X et Y, et le système de contraintes SC de la classe (PSCP) défini par :

$$\begin{cases} a \cup f(Y, a) = Y \\ Y \subseteq f_1^{-1}(X) \\ X \subseteq f(Y, Y) \end{cases}$$

Considérons maintenant la suite $(t_n)_{n \in \mathbb{N}}$ définie par :

$$t_0 = a \quad \text{et} \quad \forall n \in \mathbb{N},\ t_{n+1} = f(t_n, a).$$

Alors pour toute solution $\mathcal{I}$ du système $\mathcal{I}(Y) = \{t_n \mid n \in \mathbb{N}\}$.

Supposons que l'ensemble des solutions de SC soit solution d'un système de contraintes de la classe (PNSC). Alors $Sol(SC)$ est l'intersection d'un fermé Fer et d'un ouvert Ouv d'après la proposition 373 :

$$Sol(SC) = Fer \cap Ouv.$$

Considérons l'ensemble $S_0 = (\{f(t_n, t_n) \mid n \in \mathbb{N}\}, \{t_n \mid n \in \mathbb{N}\})$. S_0 est solution du système SC et est donc inclus dans Ouv. Donc il existe $r > 0$ tel que $\mathcal{B}(S_0, r) \subseteq Ouv$. Il existe alors un entier non nul n_0 tel que $\mathcal{B}(S_0, 2^{-(n_0-1)}) \subseteq Ouv$. Nous en déduisons que la boule fermée $BF = \mathcal{B}_F(S_0, 2^{-n_0})$ est incluse dans l'ensemble Ouv. Finalement l'ensemble $Fer \cap BF$ est fermé et inclus dans $Sol(SC)$.

Considérons maintenant la suite $(S_p)_{p\in\mathbb{N}}$ définie comme suit :

$$\forall p \in \mathbb{N}, S_p = (\{f(t_n, t_n) \mid n \leq n_0\} \cup \{f(t_n, t_{n+p}) \mid n > n_0\}, \{t_n \mid n \in \mathbb{N}\}).$$

Montrons que $(S_p)_{p\in\mathbb{N}}$ est une suite dans $Fer \cap BF$. Pour tout entier p, S_p est solution de SC donc S_p appartient à l'ensemble Fer. De plus pour p entier non nul, $d_2(S_0, S_p) = 2^{-(n_0+2)}$ puisque $Haut(f(t_{n_0+1}, t_{n_0+1})) = n_0 + 2$ et

$$S_0 \Delta S_p = \{f(t_n, t_n) \mid n > n_0\} \cup \{f(t_n, t_{n+p}) \mid n > n_0\}.$$

Donc S_p appartient à BF et finalement $(S_p)_{p\in\mathbb{N}}$ est une suite dans le fermé $Fer \cap BF$. Montrons maintenant que $(S_p)_{p\in\mathbb{N}}$ converge vers $l = (\{f(t_n, t_n) \mid n \leq n_0\}, \{t_n \mid n \in \mathbb{N}\})$. Soit p entier.

$$S_p \Delta l = \{f(t_n, t_{n+p}) \mid n > n_0\}.$$

Donc $d_2(X_p, l) = 2^{-(n_0+p+2)}$ puisque $Haut(f(t_{n_0+1}, t_{n_0+p+1})) = n_0 + p + 2$. Nous en déduisons que $\lim_{p\to\infty} d(S_p, l) = 0$ et donc que $(X_p)_{p\in\mathbb{N}}$ converge vers l.

Puisque $(S_p)_{p\in\mathbb{N}}$ est une suite convergente dans $Fer \cap BF$ et que $Fer \cap BF$ est fermé, la limite l de $(S_p)_{p\in\mathbb{N}}$ appartient à $Fer \cap BF$ (théorème 29). Puisque $Fer \cap BF \subseteq Sol(SC)$, on a l appartient à $Sol(SC)$. Or l n'est pas solution du système SC puisque la seconde contrainte n'est pas satisfaite :

$$\{t_n \mid n \in \mathbb{N}\} \not\subseteq f_1^{-1}(l) = \{t_n \mid n \leq n_0\}.$$

Nous obtenons donc une contradiction. Nous en déduisons que l'ensemble $Sol(SC)$ n'est pas solution d'un système de contraintes de la classe (PNSC). Donc nous obtenons par la proposition 374 que la classe (PSCP) est strictement plus expressive que la classe (PNSC).

□

Conclusion

Nous allons conclure cette thèse en rappelant les points essentiels qui y sont développés. Nos travaux ont été effectués dans les domaines des automates d'arbres à contraintes, de la réécriture et des contraintes ensemblistes. Malgré les différences de ces domaines, nous avons développé un outil de preuves d'indécidabilité qui leur est commun : la méthode des Arbres-Grilles. Cette méthode est une adaptation de la méthode de la grille aux arbres. En fait nous définissons un codage des calculs réussis d'une machine de Turing débutant sur l'entrée vide en un ensemble d'arbres. Cette adaptation nous a permis de montrer les trois résultats suivants :

- Le problème du vide pour les automates à tests d'égalité entre cousins est indécidable (résultat précedemment établi par M. Tommasi dans son rapport de D.E.A. [86]).
- L'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre de la réécriture en un pas pour les systèmes de réécriture linéaires, minces et convergents.
- L'indécidabilité du fragment $\exists^*\forall^*$ de la théorie du premier ordre des contraintes ensemblistes pour les formules contenant pour seul opérateur ensembliste l'union $\cup$.

Ensuite, à partir de ces résultats, nous avons essayé de diminuer la taille de la frontière existant entre la décidabilité et l'indécidabilité de ces différents problèmes. Nous avons ainsi défini une nouvelle classe d'automates à tests d'égalité entre frères et cousins pour laquelle le problème du vide est décidable et nous avons montré:

- La décidabilité du fragment existentiel de la théorie du premier ordre de la réécriture en un pas, munie de contraintes d'égalité et de contraintes d'appartenance à des langages reconnaissables par automates à tests entre frères, pour les systèmes de réécriture dont toutes les variables sont à profondeur un.
- La décidabilité du problème de satisfiabilité pour les systèmes de contraintes ensemblistes positives avec tests d'égalité.

Dans cette thèse, nous nous sommes aussi intéressés à des problèmes plus spécifiques à chacun de nos domaines d'étude.

Pour les automates à contraintes, nous nous sommes posés le problème de reconnaissabilité de ceux-ci. Nous avons montré que ce problème est décidable pour la classe des langages reconnaissables par automates à tests entre frères, c'est-à-dire que pour tout langage $\mathcal{L}$ reconnu par un automate à tests entre frères, on peut décider si $\mathcal{L}$ est régulier ou non. Par contre, nous avons montré que ce problème est indécidable pour les langages reconnaissables par automates à tests d'égailité entre cousins.

Pour la réécriture, nous nous sommes intéressés à l'étude des applications sur des langages réguliers de relations basées sur des systèmes de réécriture. Par exemple, nous avons étudié le problème de décidabilité « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? » pour $\mathcal{L}_1$ et $\mathcal{L}_2$ langages réguliers et $\mathcal{R}el$ relation basée sur des systèmes de réécriture. Pour cela, nous avons eu l'idée de marquer à l'aide de morphismes les positions où des réécritures s'appliquent et d'utiliser un langage de contrôle pour vérifier la bonne application de $\mathcal{R}el$. Notre méthode nous a permis d'étudier les questions « $\mathcal{R}el(\mathcal{L}_1) \subseteq \mathcal{L}_2$? », « $\mathcal{R}el(\mathcal{L}_1) = \mathcal{L}_2$? » et « $\mathcal{R}^*(\mathcal{L})$ régulier? » pour $\mathcal{R}$ système de réécriture et $\mathcal{L}$ langage régulier. Nous retrouvons ainsi des résultats de décidabilité déjà obtenus mais nous en obtenons de nouveaux par utilisation en particulier des languages reconnaissables par automates à tests entre frères et par la décidabilité du problème de reconnaissabilité pour la classe de ces langages.

Pour les contraintes ensemblistes, nous avons étudié les propriétés topologiques de l'ensemble des solutions de certains systèmes de contraintes et le pouvoir d'expression de différentes classe de contraintes. Nous avons en particulier montré que le pouvoir d'expression en terme d'ensembles de solutions des contraintes ensemblistes positives avec symboles de projection est strictement supérieur à celui des contraintes ensemblistes positives et négatives sans symbole de projection.

La frontière entre décidabilité et indécidabilité du problème du vide pour les automates à contraintes, des formules de la théorie du premier ordre de la réécriture en un pas et des formules de la théorie du premier ordre des contraintes ensemblistes est très fine. Donc l'établissement de nouveaux résultats pour ces problèmes nécessitent des preuves très techniques. En particulier, il reste un problème ouvert très intéressant, celui du fragment existentiel de la théorie de la réécriture en un pas.

Concernant le problème de reconnaissabilité des automates à contraintes, il faudrait chercher les applications possibles du résultat de décidabilité obtenu pour les langages reconnaissables par automates à tests entre frères. Il serait également intéressant de voir si la classe des langages pour laquelle le problème de reconnaissabilité est décidable peut s'étendre.

Dans le domaine de la réécriture, nous avons défini une méthode utilisant des morphismes pour marquer les applications des règles d'un système de réécriture. Cette découverte étant récente, il faudrait voir les domaines d'applications de celle-ci. Par exemple, cette méthode s'applique-t-elle aux stratégies de réduction (Outermost, Leftmost-Outermost, Parallel-Outermost)?

Index

Bibliographie

[1] AIKEN, A., KOZEN, D., VARDI, M., AND WIMMERS, E. The complexity of set constraints. In Börger et al. [13], pp. 1–17. Techn. Report 93-1352, Cornell University.

[2] AIKEN, A., KOZEN, D., AND WIMMERS, E. Decidability of systems of set constraints with negative constraints. *Information and Computation 122*, 1 (Oct. 1995), 30–44.

[3] AIKEN, A., AND MURPHY, B. R. Implementing regular tree expressions. In *Proceedings of the ACM conf. on Functional Programming Languages and Computer Architecture* (1991), pp. 427–447.

[4] AIKEN, A., AND WIMMERS, E. Solving Systems of Set Constraints. In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science* (22–25 June 1992), IEEE Computer Society Press, pp. 329–340.

[5] BAADER, F., AND NIPKOW, T. *Term Rewriting and All That.* Cambridge University Press, 1998.

[6] BAADER, F., AND SCHULZ, K. Unification in the union of disjoint equational theories. In *Proceedings of the* 11^{th} *International Conference on Automated Deduction* (New York, June 1992), D. Kapur, Ed., vol. 607 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 50–65.

[7] BAADER, F., AND SIEKMANN, J. *Handbook of Logic in Artificial Intelligence and Logic Programming.* Oxford University Press, 1993, ch. Unification Theory.

[8] BACHMAIR, L., GANZINGER, H., AND WALDMANN, U. Set constraints are the monadic class. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science* (19–23 June 1993), IEEE Computer Society Press, pp. 75–83.

[9] BOGAERT, B. *Automates d'arbres avec tests d'égalités.* PhD thesis, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, 1990.

[10] BOGAERT, B., AND TISON, S. Automata with equality tests. Tech. Rep. IT. 207, Laboratoire d'Informatique Fondamentale de Lille, 1991.

[11] BOGAERT, B., AND TISON, S. Equality and disequality constraints on direct subterms in tree automata. In Enjalbert et al. [31], pp. 161–171.

[12] BÖRGER, E., GRÄDEL, E., AND GUREVICH, Y. *The Classical Decision Problem.* Springer-Verlag, 1997.

[13] BÖRGER, E., GUREVICH, Y., AND MEINKE, K., Eds. *Proceedings of Computer Science Logic* (1993), vol. 832 of *Lecture Notes in Computer Science.*

[14] CARON, A.-C. *Structures et Décision en Réécriture.* PhD thesis, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, Feb. 1993.

[15] CARON, A.-C., COMON, H., COQUIDÉ, J.-L., DAUCHET, M., AND JACQUEMARD, F. Pumping, cleaning and symbolic constraints solving. In *Proceedings, International Colloquium Automata Languages and Programming* (1994), vol. 820 of *Lecture Notes in Computer Science*, pp. 436–449.

[16] CARON, A.-C., COQUIDÉ, J.-L., AND DAUCHET, M. Reduction properties and automata with constraints. *Journal of Symbolic Computation 20* (1995), 215–233.

[17] CHARATONIK, W., AND PACHOLSKI, L. Negative set constraints with equality. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science* (4–7 July 1994), IEEE Computer Society Press, pp. 128–136.

[18] CHARATONIK, W., AND PACHOLSKI, L. Set constraints with projections are in NEXPTIME. In *Proceedings of the* 35^{th} *Symp. Foundations of Computer Science* (1994), pp. 642–653.

[19] COMON, H. Completion of rewrite systems with membership constraints. In *Proc. 19th Int. Coll. On Automata, Languages and Programming* (Vienna, 1992), W. Kuich, Ed., vol. 623 of *Lecture Notes in Computer Science*, Springer-Verlag.

[20] COMON, H., DAUCHET, M., GILLERON, R., LUGIEZ, D., TISON, S., AND TOMMASI, M. Tree automata techniques and applications, 1998. Available through WWW from http://l3ux02.univ-lille3.fr/tata.

[21] COQUIDE, J.-L., DAUCHET, M., GILLERON, R., AND VAGVOLGYI, S. Bottom-up tree pushdown automata: Classification and connection with rewrite systems. *Theorical Computer Science 127* (1994), 69–98.

[22] DAUCHET, M. Simulation of turing machines by a left-linear rewrite rule. In *Proceedings, Third International Conference on Rewriting Techniques and Applications* (1989), N. Dershowitz, Ed., vol. 355, Springer-Verlag, pp. 109–120.

[23] DAUCHET, M., CARON, A.-C., AND COQUIDÉ, J.-L. Reduction properties and automata with constraints. *Journal of Symbolic Computation 20* (1995), 215–233.

[24] DAUCHET, M., AND TISON, S. The theory of ground rewrite systems is decidable. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* (4–7 June 1990), IEEE Computer Society Press, pp. 242–248.

[25] DERSHOWITZ, N. Termination of linear rewriting systems. In *Automata, Languages and Programming* (1981), S. Even and O. Kariv, Eds., vol. 115 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 448–458.

[26] DERSHOWITZ, N., AND JOUANNAUD, J. *Handbook of Theoretical Computer Science*, vol. B. Elsevier, 1990, ch. Rewrite Systems, pp. 243–320.

[27] DUGUNDJI, J. *Topology.* Allyn and Bacon, Inc., 1976.

[28] ENGELFRIET, J. Bottom-up and top-down tree transformations. a comparision. *Mathematical System Theory 9* (1975), 198–231.

[29] ENGELFRIET, J., AND SCHMIDT, E. IO and OI I. *Journal of Comput. and Syst. Sci. 15* (1977), 328–353.

[30] ENGELFRIET, J., AND SCHMIDT, E. IO and OI II. *Journal of Comput. and Syst. Sci. 16* (1978), 67–99.

[31] ENJALBERT, P., FINKEL, A., AND WAGNER, K. W., Eds. *9th Annual Symposium on Theoretical Aspects of Computer Science* (1992), vol. 577 of *Lecture Notes in Computer Science.*

[32] ENJALBERT, P., FINKEL, A., AND WAGNER, K. W., Eds. *10th Annual Symposium on Theoretical Aspects of Computer Science* (25–27 Feb. 1993), vol. 665 of *Lecture Notes in Computer Science.*

[33] FÜLÖP, Z., JURVANEN, E., STEINBY, M., AND VÁGVÖLGYI, S. On one-pass term rewriting. In *Mathematical Foundations of Computer Science* (1998).

[34] GANZINGER, H., Ed. *Proceedings. Seventh International Conference on Rewriting Techniques and Applications* (1996), vol. 1103 of *Lecture Notes in Computer Science.*

[35] GILLERON, R., TISON, S., AND TOMMASI, M. Solving System of Set Constraints using Tree Automata. In Enjalbert et al. [32], pp. 505–514.

[36] GILLERON, R., TISON, S., AND TOMMASI, M. Solving systems of set constraints with negated subset relationships. In *Proceedings of the* 34^{th} *Symp. on Foundations of Computer Science* (1993), pp. 372–380. Full version in the LIFL Tech. Rep. IT-247.

[37] GILLERON, R., TISON, S., AND TOMMASI, M. Set constraints and automata. Tech. Rep. IT. 292, Laboratoire d'Informatique Fondamentale de Lille, 1996.

[38] GOLDFARB, W. The undecidability of the second-order unification problem. *Theorical Computer Science 13* (1981), 225–230.

[39] GUREVICH, Y. The decision problem for standard classes. *Journal of Symbolic Computation 41* (1976), 460–464.

[40] HEINTZE, N. *Set Based Program Analysis.* PhD thesis, Carnegie Mellon University, 1992.

[41] HEINTZE, N., AND JAFFAR, J. A Decision Procedure for a Class of Set Constraints. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* (4–7 June 1990), IEEE Computer Society Press, pp. 42–51.

[42] HEINTZE, N., AND JAFFAR, J. A finite presentation theorem for approximating logic programs. In *Proceedings of the 17th ACM Symp. on Principles of Programming Languages* (1990), pp. 197–209. Full version in the IBM tech. rep. RC 16089 (#71415).

[43] HOPCROFT, J., AND ULLMAN, J. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 1979.

[44] HUET, G. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM 27* (1980), 797–821.

[45] J. NIEHREN, M. P. E. P. R. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *International Conference On Automated Deduction* (July 1997), vol. 1249 of *Lecture Notes in Computer Science*, pp. 34–48.

[46] JACQUEMARD, F. *Automates d'arbres et réécriture de termes.* PhD thesis, Université de Paris XI, 1996.

[47] JACQUEMARD, F. Decidable approximations of term rewriting systems. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications* (1996), H. Ganzinger, Ed., vol. 1103 of *Lecture Notes in Computer Science*, pp. 362–376.

[48] JAFFAR, J. Minimal and complete word equation. *Journal of the ACM 37*, 1 (1990), 47–85.

[49] JONES, N., AND MUCHNICK, S. Flow Analysis and Optimization of LISP-like Structures. In *Proceedings of the 6th ACM Symposium on Principles of Programming Languages* (1979), pp. 244–246.

[50] KAHR, A., MOORE, E., AND ULLMAN, J. Entscheidungsproblem reduced to the $\forall\exists\forall$ case. In *Proceedings of the National Academy of Sciences* (USA, 1962), vol. 48, pp. 365–377.

[51] KNUTH, D., AND BENDIX, P. Simple word problems in universal algebra. In *Computational Problems in Abstract Algebra* (1970), J. Leech, Ed., Pergamon Press, pp. 263–297.

[52] KOŚCIELSKI, A., AND PACHOLSKI, L. Complexity of makanin's algorithm. *Journal of the ACM 43*, 4 (1996).

[53] KOZEN, D. Rational spaces and set constraints. In *Proceedings of the 6th International Joint Conference on Theory and Practice of Software Development* (1995), vol. 915 of *Lecture Notes in Computer Science*, pp. 42–61.

[54] KUCHEROV, G., AND TAJINE, M. Decidability of regularity and related properties of ground normal form languages. In *Proceedings of 3rd International Workshop on Conditional Rewriting Systems* (1992), pp. 150–156.

[55] LENTIN, A. In *Automata, Languages and Programming*, M. Nivat, Ed. Elsevier Science, 1992, ch. Equations in Free Monoids.

[56] LENTIN, A., AND SCHÜTZENBERGER, M. A combinatorial problem in the theory of free monoids. In *Proceedings of the Conference on Combinatorical Mathematics and its Applications* (University of North Carolina, Chapel Hill, 1969), R. Bose, Ed.

[57] LÉVY, J. Linear second order unification. In *International Conference on Rewriting Techniques and Applications* (1996), vol. 1103 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 332–346.

[58] LEWIS, H. *Unsolvable Classes of Quantificational Formulas*, reading. mass. ed. Addison-Wesley, 1979.

[59] MAKANIN, G. The problem of solvability of equations in a free semigroup. Tech. Rep. 223(2), Soviet Akad. Nauk SSSR, 1977.

[60] MARCINKOWSKI, J. A horn clause that implies an undecidable set of horn clauses. In *Annual Conference of the European Association of Computer Science Logic* (1993), vol. 832 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 223–237.

[61] MARCINKOWSKI, J. Undecidability of the first-order theory of one-step right ground rewriting. In *8th International Conference on Rewriting Techniques and Applications* (Sitges, Spain, June 1997), H. Comon, Ed., vol. 1232 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 241–253.

[62] MARCINKOWSKI, J., AND PACHOLSKI, L. Undecidability of the horn clause implication problem. In *33th IEEE FOCS* (Los Alamitos, 1992), pp. 354–362.

[63] MARKOV, A. The theory of algorithms. *Trudy Mat. Inst. Steklov* (1954). English Translation in Israel Program for Scientific Translations, Jerusalem, 1968.

[64] MISHRA, P. Towards a Theory of Types in PROLOG. In *Proceedings of the 1st IEEE Symposium on Logic Programming* (Atlantic City, 1984), pp. 456–461.

[65] MONGY, J. *Transformation de noyaux reconnaissables d'arbres. Forêts RATEG.* PhD thesis, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, 1981.

[66] NIEHREN, J., PINKAL, M., AND RUHRBERG, P. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *Proceedings of the International Conference on Automated Deduction* (Townsville, Australia, July 1997), vol. 1249 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 34–48.

[67] NIEHREN, J., AND TREINEN, R. Context unification and rewriting constraints. Presented at CCL2 Workshop, 1998.

[68] PÉCUCHET, J. *Equations avec Constantes et Algorithme de Makanin.* PhD thesis, Laboratoire Informatique Université de Rouen, France, 1981.

[69] PÉCUCHET, J. Solution principale et rang d'un système d'equations avec constantes dans le monoïde libre. *Discrete Mathematics 48* (1984), 253–274.

[70] PLAISTED, D.-A. Semantic confluence tests and completion method. *Information and Control 65* (1985), 182–215.

[71] PLOTKIN, G. Bluding in equational theories. *Machine Intelligence 7* (1972), 73–90.

[72] POST, E. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society 52* (1946), 264–268.

[73] POST, E. L. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic 13* (1947), 1–11.

[74] REYNOLDS, J. Automatic Computation of Data Set Definition. *Information Processing 68* (1969), 456–461.

[75] RICE, H. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society 6* (1953), 373–385.

[76] SALOMAA, K. Deterministic tree pushdown automata and monadic tree rewriting systems. *Journal of Comput. and Syst. Sci. 37* (1988), 367–394.

[77] SCHMIDT-SCHAUSS, M. Unification of stratified second-order terms. Internal Report 12/94, Johann-Wolfgang-Goethe-Universität, Frankfurt, Germany, 1994.

[78] SCHMIDT-SCHAUSS, M. A unification algorithm for distributivity and a multiplicative unit. *Journal of Symbolic Computation 22*, 3 (1997), 315–344.

[79] SCHULZ, K. Makanin's algorithm, two improvements and a generalization. Tech. Rep. CIS–Bericht–91–39, Centrum für Informations and Sprachverarbeitung, Universität München, 1991.

[80] SCHULZ, K. Word unification and transformation of generalized equations. *Journal of Automated Reasoning 11* (1993), 149–184.

[81] SCOTT, D. Some definitional suggestions for automata theory. *Journal of Comput. and Syst. Sci. 1*, 2 (1967), 187–212.

[82] SIEKMANN, J. String unification: Part 1. Internal Report CSM7, Essex University, 1975.

[83] TAITSLIN, M. Some further examples of undecidable theories. *Algebra and Logic 7* (1968), 127–129. Original paper (russian) in Algebra i Logica 6(3):105-111, 1967.

[84] TALBOT, J.-M. *Contraintes Ensemblistes Définies et Co-Définies.* PhD thesis, Université de Lille 1, 1998.

[85] THOMAS, W. *Handbook of Theoretical Computer Science*, vol. B. Elsevier, 1990, ch. Automata on Infinite Objects, pp. 134–191.

[86] TOMMASI, M. Automates d'arbres avec tests d'égalité entre cousins germains. Mémoire de DEA, Univ. Lille I, 1992.

[87] TOMMASI, M. *Automates et contraintes ensemblistes.* PhD thesis, LIFL, 1994.

[88] TREINEN, R. The first-order theory of one-step rewriting by a linear term rewriting system is undecidable (extended abstract), June 1996.

[89] TREINEN, R. The first-order theory of one-step rewriting is undecidable. In *7th International Conference on Rewriting Techniques and Applications* (New Brunswick, NJ, USA, July 1996), H. Ganzinger, Ed., vol. 1103 of *LNCS*, Springer-Verlag, pp. 276–286.

[90] TREINEN, R. The first-order theory of one-step rewriting is undecidable. In Ganzinger [34], pp. 276–286.

[91] TURING, A. On computable numbers, with applications to the entscheidungsproblem. In *Proceedings of the London Mathematical Society* (1936), vol. 42, pp. 230–265.

[92] VÁGVÖLGYI, S., AND GILLERON, R. For a rewrite system it is decidable whether the set of irreducible ground terms is recognizable. *Bulletin of the European Association of Theoretical Computer Science 48* (Oct. 1992), 197–209.

[93] VAM EMDE BOAS, P. Dominos are forever. In *Proceedings of the 1st GTI Wokshop* (Paderborn, 1983), pp. 76–95.

[94] VOROBYOV, S. The first-order theory of one-step rewriting in linear noetherian systems is undecidable. In *8th International Conference on Rewriting Techniques and Applications* (Sitges, Spain, June 1997), H. Comon, Ed., vol. 1232 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 254–268.

[95] WANG, H. Dominos and the ∀∃∀ case of the decision problem. In *Mathematical Theory of Automata* (Brooklyn, 1963), Polytechnic Institute.