THÈSE

présentée à

L'UNIVERSITÉ DE LA MÉDITERRANÉE
AIX-MARSEILLE II

École doctorale de Mathématiques et Informatique

par

Dao Thi Bich Hanh

pour obtenir le grade de

DOCTEUR ÈS SCIENCES

spécialité :

INFORMATIQUE

Résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis

---

soutenue le : 4 décembre 2000

devant le jury composé de :

M. Alain Colmerauer	Directeur de thèse
Mme. Nadia Creignou	Examinatrice
M. François Fages	Rapporteur
M. Leszek Pacholski	Rapporteur

À mes parents et à ma sœur

Rapporteurs et membres du jury

Alain Colmerauer Directeur de thèse
Professeur à l'Université de la Méditerranée
Département d'Informatique
Faculté des Sciences de Luminy
163 avenue de Luminy
13288 Marseille Cedex 9
France

Nadia Creignou Examinatrice
Professeur à l'Université de la Méditerranée
Département d'Informatique
Faculté des Sciences de Luminy
163 avenue de Luminy
13288 Marseille Cedex 9
France

François Fages Rapporteur
Directeur de recherche
INRIA Rocquencourt
Domaine de Voluceau
BP 105
78153 Le Chesnay Cedex
France

Leszek Pacholski Rapporteur
Professeur à l'Université de Wroclaw
Instytut Informatyki U. Wr.
Przesmyckiego 20
51-151 Wroclaw
Poland

Remerciements

Je sais qu'il est difficile de remercier toutes les personnes qui m'ont aidée et soutenue pendant ces années de préparation de thèse, mais je vais essayer de faire de mon mieux.

Mon directeur de thèse, Alain Colmerauer, est la première personne que je tiens à remercier chaleureusement, bien que je sois sûre que tous mes remerciements ne peuvent témoigner entièrement ma gratitude envers lui. Il m'a montré un domaine de recherche intéressant et m'a donné son temps ainsi que des conseils précieux pendant toutes ces années. Les discussions avec lui ont été une source abondante d'idées intéressantes. Je suis reconnaissante de sa disponibilité et de son aide. Il a toujours été là, même pendant le week-end, lorsque j'avais des points à discuter. Ses instructions sur le français et sur le style de rédaction ont été et resteront encore très utiles pour moi. Éloignée de ma famille, je remercie également Colette pour les moments familiaux si agréables qu'elle m'a apportés.

Je tiens à remercier Leszek Pacholski, qui m'a fait l'honneur d'être rapporteur de ma thèse. Je lui remercie aussi pour les discussions qu'il a eues, lors de son séjour à Marseille, avec mon directeur de thèse, car elles ont été le point de départ d'une partie de ma recherche.

Je tiens à remercier François Fages pour l'honneur qu'il m'a fait en acceptant de rapporter sur ma thèse. La discussion avec lui a été intéressante. Je le remercie également pour sa réponse rapide à ma demande de documentation.

Je tiens à remercier Nadia Creignou pour avoir aimablement accepté de participer au jury de ma thèse. Elle a pris du temps pour m'écouter et pour discuter avec moi, je la remercie également pour celà.

Je remercie les membres de l'ancienne équipe « Contraintes, Algorithmes et Combinatoire » et tout particulièrement Jean-François Maurras, Victor Chepoï, Noëlle Bleuzen, Touraïvane, Michel Van Caneghem ainsi que les membres de l'équipe « Logique et Complexité » dont je fais maintenant partie, pour les nombreux discussions.

Merci à Guylaine Vincent pour son aide pour mes recherches bibliographiques, et à Sylvie Calabresse et Solange Panattoni pour leur gentillesse et leur aide.

Je remercie Mireille Decorps pour sa gentillesse, pour son aide administrative et pour les activités culturelles qu'elle a organisées et qui ont beaucoup enrichi mes connaissances générales.

Je remercie Traïan, Léon qui m'ont encouragée, et remercie Bruno, Ian, Nicolas, Pedro, David, Stéphane, Fabienne et les autres doctorants à l'ESIL et au LIM qui m'ont entourée et m'ont apporté leur amitié.

Merci à mes vieux amis Duong, Huyen, Nhoi, Hung qui m'ont soutenue et cru en moi. Les heures pendant lesquelles on s'est parlé et le nombre énorme d'e-mails qu'on s'est échangés m'ont apporté beaucoup de joie et du courage.

Pendant ces années de thèse, j'ai été heureuse de faire la connaissance d'étudiants venant de différents pays et qui sont devenus des amis : Sandra, Kaeko, Mildreth, Marina, Lamia, Monica,

Sonia, Gabrielle, et Cuong, Phuong, Duc, Duy, Dung, Viet Anh, Thu et bien d'autres dont je ne peux pas écrire tous les noms ici. Je vous remercie tous pour m'avoir entourée et soutenue. Merci également à M. Phuc et à Mme. Lan pour leur gentillesse.

Merci à ma famille, mes grands parents, mes oncles, mes tantes, mes cousins et cousines pour l'amour et l'encouragement qu'ils m'ont toujours réservés.

Encore merci, merci à tous !

Résumé

L'objet de cette thèse est la résolution de contraintes dans la théorie complète **T** des arbres éventuellement infinis. Ces contraintes se présentent sous forme de formules générales du premier ordre construites à partir d'un ensemble infini de symboles d'opérations et de l'égalité. Par, résoudre une contrainte p dans **T**, nous entendons : transformer p en une contrainte q équivalente dans **T**, qui est la constante logique *vrai* si p est toujours vraie dans **T**, et la constante *faux* si p est toujours fausse dans **T**. De plus, si p ou sa négation a une base finie de solutions dans un modèle de **T**, alors ces solutions ou non-solutions doivent être explicites dans q.

Après avoir introduit l'algèbre des arbres et montré qu'elle constitue bien un modèle de la théorie des arbres, nous montrons que nos contraintes ont un pouvoir d'expression « quasi-universel » et proposons un algorithme en 11 règles de réécriture de sous-formules pour les résoudre. Nous considérons ensuite la structure obtenue en introduisant dans cette algèbre un prédicat unaire $\mathit{fini}(x)$, exprimant que x est un arbre fini. Nous proposons une extension **T**∪FINI de la théorie **T** admettant cette structure comme modèle. Nous donnons un algorithme pour résoudre les contraintes dans **T**∪FINI en 16 règles de réécriture et concluons ainsi que **T**∪FINI est une théorie complète. Nous donnons également un système de 12 règles de réécriture qui effectue le même type de résolution de contraintes dans la théorie des arbres finis. Nous terminons par des détails sur l'implantation d'algorithmes de résolution de contraintes, dans la théorie **T** des arbres éventuellement infinis et dans la théorie des arbres finis. Les exemples qui nous ont servi à montrer le pouvoir d'expression de nos contraintes nous permettent de produire des benchmarks intéressants.

Mots-clefs Arbres finis, arbres infinis, algèbre des arbres, théorie des arbres, contraintes, résolution de contraintes, règles de réécriture.

Table des matières

Chapitre 1

Introduction

1.1 À propos des contraintes d'arbres

Les arbres éventuellement infinis, jouent un rôle fondamental en informatique. Ils modélisent aussi bien des structures de données que des schémas ou des déroulements de programme. Dès 1976, Gérard Huet a proposé un algorithme pour unifier des termes infinis, c'est-à-dire résoudre des équations dans les arbres [27]. Bruno Courcelle a étudié les propriétés des arbres infinis notamment dans le cadre des schémas récursifs de programme [19, 20]. Alain Colmerauer a modélisé le fonctionnement de Prolog II, III et IV par la résolution d'équations et de diséquations dans les arbres infinis [7, 11, 8, 9, 5]. Il est donc intéressant de résoudre les contraintes les plus générales et les plus naturelles possibles sur les arbres, à notre connaissance, les formules du premier ordre, construites sans restrictions à partir de variables, désignant des arbres, d'opérations de construction, de l'égalité, de la négation, des connecteurs logiques et des quantificateurs.

L'unification dans les termes finis, c'est-à-dire la résolution de conjonctions d'équations portant sur des arbres finis, a d'abord été étudiée par A. Robinson [43]. Des algorithmes avec une meilleur complexité ont ensuite été proposés par M.S. Paterson et M.N. Wegman [41] et A. Martelli et U. Montanari [39]. La résolution de conjonctions d'équations dans les arbres infinis a été étudié par G. Huet [27], par A. Colmerauer [7, 8] et par J. Jaffar [28]. Pour une synthèse à ce sujet, nous renvoyons à l'article de J.-P. Jouannaud et C. Kirchner [29]. La résolution de conjonctions d'équations et de diséquations portant sur des arbres éventuellement infinis a été étudié par A. Colmerauer [8] et H.-J. Bürckert [6]. Un algorithme incrémental de résolution de conjonctions d'équations et de diséquations portant sur des arbres rationnels a ensuite été proposé par V. Ramachandran et P. Van Hentenryck [42]. La résolution de diséquations quantifiées universellement portant sur des arbres finis a été étudiée par D.A. Smith [45].

En ce qui concerne les contraintes qui sont des formules générales du premier ordre, il existe des algorithmes d'élimination de quantificateurs, pour les transformer en des combinaisons booléennes de contraintes simples. Pour le cas des arbres finis, nous pouvons citer le travail de A. Malcev [38], de K. Kunen [31], de M.J. Maher [36] et de H. Comon [12, 14, 15]. Pour le cas des arbres infinis avec un ensemble fini de symboles de fonction, on peut consulter M.J. Maher [36] et H. Comon [12]. Pour le cas des arbres éventuellement infinis avec un ensemble infini de symbole de fonction, il y a le travail de M.J. Maher [36]. On trouvera une synthèse sur ce sujet dans le papier de H. Comon [13]. Michael J. Maher a axiomatisé tous ces cas de figures par des théories complètes. Entre autres, il a introduit la théorie **T** des arbres (éventuellement) infinis comportant un ensemble infini de symboles.

C'est cette théorie **T** qui a été le point de départ de notre travail. Notre but était de construire un algorithme permettant de résoudre effectivement les contraintes les plus générales dans cette théorie, c'est-à-dire dans tout modèle $\mathcal{M}$ de **T**. Encore fallait-il s'entendre sur ce que signifie « résoudre effectivement ». Pour nous, il s'agit de transformer une contrainte de départ p en une contrainte équivalente dans **T**, qui est la constante logique *vrai* si p est toujours vraie dans **T**, est la constante logique *faux* si p est toujours fausse dans **T**. Nous exigeons même plus : si p ou sa négation a un ensemble fini de solutions dans $\mathcal{M}$, alors ces solutions et ces non-solutions doivent être explicitement visibles dans la formule obtenue. Nous voulions aussi être capable de tester notre algorithme sur des exemples probants et nous avons passé beaucoup de temps à construire de tels exemples et d'une façon générale, à étudier le pouvoir d'expression des contraintes générales d'arbres. Enfin nous nous sommes permis d'étendre nos résultats en adjoignant à l'algèbre des arbres un prédicat contraignant un arbre à être fini. Nous avons aussi étudié le cas des arbres finis.

1.2 Contenu et organisation de la thèse

Cette thèse est organisée en 7 chapitres dont le premier constitue l'introduction et le dernier la conclusion.

Dans le chapitre 2, nous rappelons et précisons les notions essentielles concernant l'algèbre des arbres et la théorie **T** des arbres. Nous montrons que l'algèbre des arbres est bien un modèle de cette théorie, qui fait intervenir une infinité de symboles d'opération et l'égalité. Une grande partie de ce chapitre reprend des notes de cours d'Alain Colmerauer [1].

Le chapitre 3 est consacré au pouvoir d'expressions de contraintes générales d'arbres. Nous examinons tout d'abord des contraintes faisant intervenir de longues suites alternées de quantificateurs $\exists\forall\exists\forall\ldots$. Nous montrons comment exprimer les positions gagnantes d'un jeux à deux adversaires par de telles contraintes et appliquons nos résultats à trois exemples. Nous nous intéressons ensuite à la famille de contraintes la plus expressive que nous connaissons et qui fait intervenir le nombre immensément grand $\delta(k)$ obtenu en évaluant de haut en bas une tour de puissances de 2 de hauteur k. Avec des éléments de cette famille, de taille linéaire en k, nous contraignons une variable à être égale à un arbre fini, dont le nombre de nœuds est $\delta(k)$, et nous exprimons le résultat d'une machine Prolog exécutant au plus $\delta(k)$ instructions. En remplaçant la machine Prolog par une machine de Turing nous retrouvons le résultat de complexité de Sergei Vorobyov [48]. Cette dernière partie a été fortement inspirée par le travail de Pawel Mielniczuk [40] et tout ce chapitre est le résultat d'un travail en forte collaboration avec mon directeur de thèse.

Dans le chapitre 4, nous proposons un algorithme de résolution de contraintes dans la théorie des arbres **T**, exprimé sous forme d'un système de 11 règles de réécriture de sous-formules. La formule résolue q obtenue à partir d'une formule p est équivalente à p dans **T** et ne fait pas intervenir d'autres variables libres que celles de p. Cette formule q est la constante logique *vrai* si p est toujours vraie dans **T** et est la constante logique *faux* si p est toujours fausse dans **T**. Plus précisément, si p ou sa négation ont un ensemble fini de solutions dans un modèle de **T**, alors ces solutions ou non-solutions sont explicitement visibles dans q. Nous traitons ensuite le cas particulier où l'on s'intéresse uniquement à décider si p est toujours vrai ou si p est toujours faux. Nous montrons aussi comment éliminer des parties redondantes au cours de la résolu-

1. Éléments de programmation par contraintes, Notes de cours d'Alain Colmerauer, Diplôme d'Études Approfondies d'Informatique, Université de la Méditerranée, Octobre 2000 (en perpétuelle évolution).

tion. Enfin nous estimons une borne supérieure pour la taille de la formule q obtenue. Nous montrons que cette taille est bornée par quelque chose qui est proche d'une tour de puissances de 2 (avec associations de haut en bas) dont la hauteur est la profondeur maximale d'imbrication des négations (si on n'utilise que la quantification existentielle, la négation et le connecteur "et").

Dans le chapitre 5, nous considérons la structure obtenue en introduisant dans l'algèbre des arbres un prédicat unaire $\mathit{fini}(x)$, qui exprime que x est un arbre fini. Nous proposons une extension **T**∪FINI de **T** admettant cette structure comme modèle. Nous donnons un système de 16 règles de réécriture qui résout des contraintes générales dans cette structure et produit des formules résolues ayant les propriétés précédemment citées. De mêmes que les 11 règles précédentes confirmaient que **T** était une théorie complète, ces 16 règles montre que **T**∪FINI est aussi une théorie complète. Nous donnons également un système de 12 règles de réécriture qui, dans l'algèbre des arbres finis, effectue le même type de résolution de contraintes.

Le chapitre 6 présente des détails sur l'implantation de nos algorithmes de résolution de contraintes dans l'algèbre des arbres et l'algèbre des arbres finis. Les exemples qui nous ont servi à montrer le pouvoir d'expression de nos contraintes dans le chapitre 3, nous permettent de produire des benchmarks intéressants.

Chapitre 2

Arbres

Sommaire

Ce chapitre est consacré aux arbres et à l'algèbre des arbres, à la logique du premier ordre et à la théorie des arbres de Michael Maher. Il se termine en montrant que l'algèbre des arbres est bien un modèle de cette théorie des arbres.

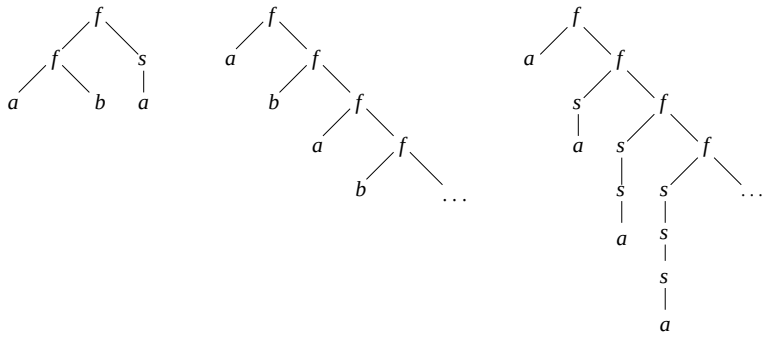
2.1 Algèbre des arbres

2.1.1 Qu'est-ce qu'un arbre?

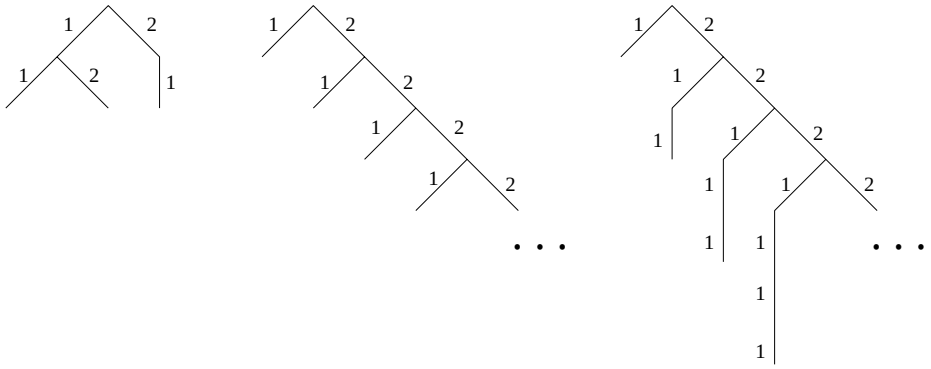
On se donne un ensemble E dont les éléments sont appelés *étiquettes.* À chaque étiquette est associée un entier non négatif n, son *arité.*

Les arbres qui nous intéressent sont formés de nœuds étiquetés par les éléments de E. Les branches qui partent d'un nœud donné sont ordonnées et leur nombre est égal à l'arité de l'étiquette de ce nœud. Voici trois arbres avec l'ensemble E contient entre autres les symboles

a, b, s, f d'arités respectives $0, 0, 1, 2$.



On notera qu'un arbre (que l'on dessine toujours à l'envers) peut comporter un ensemble infini de nœuds, mais que l'embranchement de chacun de ses nœuds est toujours fini. Pour aboutir à une définition formelle de nos arbres il faut pouvoir préciser ce qu'est un nœud. Pour ceci on numérote de 1 à n et de gauche à droite les branches qui connectent un nœud à ses n fils. Sur les exemples précédents cela donne :



La position p d'un nœud c, étiqueté e, est alors la suite d'entiers positifs que l'on rencontre pour aller à c en partant de la racine de l'arbre. En assimilant c au couple (p, e), les trois arbres précédents se représentent par les ensembles de nœuds:

$$\{(\varepsilon, f),\ (1, f),\ (2, s),\ (11, a),\ (12, b),\ (21, b)\},$$

$$\{(\varepsilon, f),\ (1, a),\ (2, f),\ (21, b),\ (22, f),\ (221, a),\ (222, f),\ (2221, b), \ldots\},$$

$$\left\{ \begin{array}{l} (\varepsilon, f),\ (1, a),\ (2, f),\ (21, s),\ (22, f),\ (211, a),\ (221, s),\ (222, f), \\ (2211, a),\ (2221, s),\ (2222, f),\ (22111, a),\ (22211, a),\ (222111, a),\ (2221111, a), \ldots \end{array} \right\},$$

où le signe ε désigne la suite vide.

Formalisons tout ceci. Une *position* est un mot construit sur les entiers strictement positifs. Le mot vide est noté ε. Soit p une position et e une étiquette. Le couple (p, e) est un *nœud*. Sa *profondeur* est la longueur de p. Une *racine* est un nœud de profondeur nulle. Le *rang* d'un nœud de profondeur non nulle est le dernier entier de sa position. On dit que c est un *père* de c' ou que c' est un *fils* de c, si c et c' sont des nœuds dont les positions sont respectivement de la forme $i_1 \ldots i_k$ et $i_1 \ldots i_k i_{k+1}$, où les i_j sont des entiers strictement positifs et où k peut être nul. On note **N** l'ensemble de tous les nœuds (construit à partir de E).

Définition 2.1 *Un élément c de **N** est* arborescent *dans un sous-ensemble a de **N** si a est non vide et, dans le cas où c est élément de a, on a à la fois:*

1. *la position de c n'est la position d'aucun autre élément de a,*

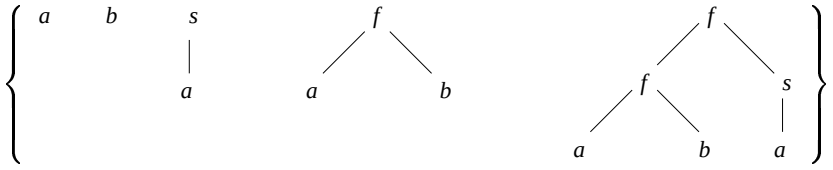
2. *c est un nœud racine ou a un père dans a,*
3. *si l'étiquette de c est d'arité n alors c a n fils dans a de rangs $1, \ldots, n$.*

Un arbre *a est un sous-ensemble de **N** dans lequel tout élément c de **N** est arborescent.*

Dans le dessin d'un arbre figurent d'autres arbres généralement plus petits, ce sont des *sous-arbres*.

Définition 2.2 *Étant donné un arbre a, le* sous-arbre partant du nœud *$(i_1 \ldots i_k, e)$ de a est l'ensemble des nœuds de la forme $(i_{k+1} \ldots i_{k+n}, e')$ avec $(i_1 \ldots i_{k+n}, e')$ élément de a et $n \geq 0$. Un* sous-arbre de a de profondeur k *est un sous-arbre de a partant d'un nœud de profondeur k et un* sous-arbre de a *est un sous-arbre de a de profondeur quelconque.*

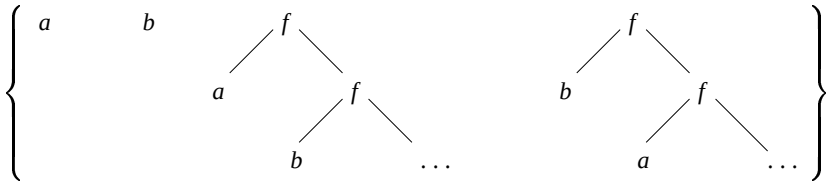
De la définition d'un arbre on conclut qu'un sous-arbre est bien un arbre. Si nous reprenons les trois exemples d'arbres proposés, le premier a pour ensemble de sous-arbres :



c'est-à-dire

$$\{\{(\varepsilon, a)\},$$
$$\{(\varepsilon, b)\},$$
$$\{(\varepsilon, s),\ (1, b)\},$$
$$\{(\varepsilon, f),\ (1, a),\ (2, b)\},$$
$$\{(\varepsilon, f),\ (1, f),\ (2, s),\ (11, a),\ (12, b),\ (21, b)\}\}$$

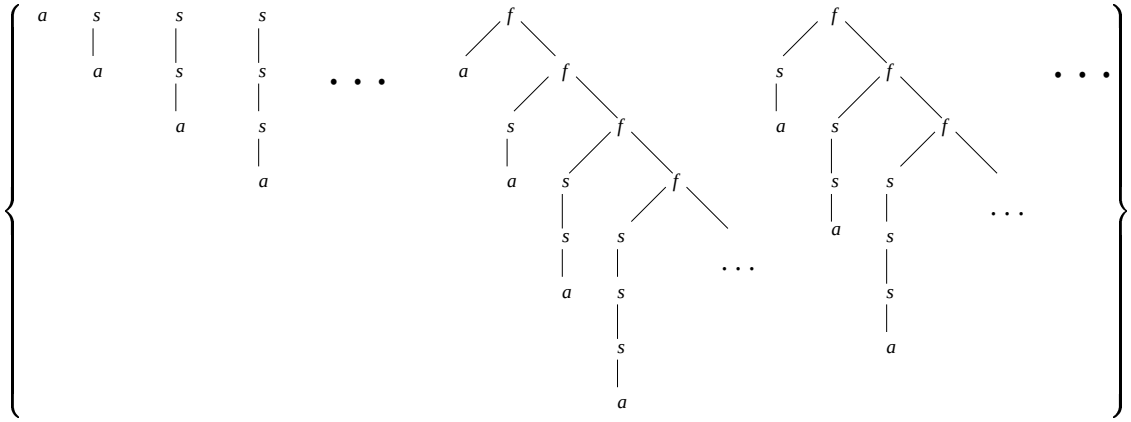
et le deuxième :



c'est-à-dire

$$\{\{(\varepsilon, a)\},$$
$$\{(\varepsilon, b)\},$$
$$\{(\varepsilon, f),\ (1, a),\ (2, f),\ (21, b),\ (22, f),\ (221, a), \ldots\},$$
$$\{(\varepsilon, f),\ (1, b),\ (2, f),\ (21, a),\ (22, f),\ (221, b), \ldots\}\}$$

et le troisième :



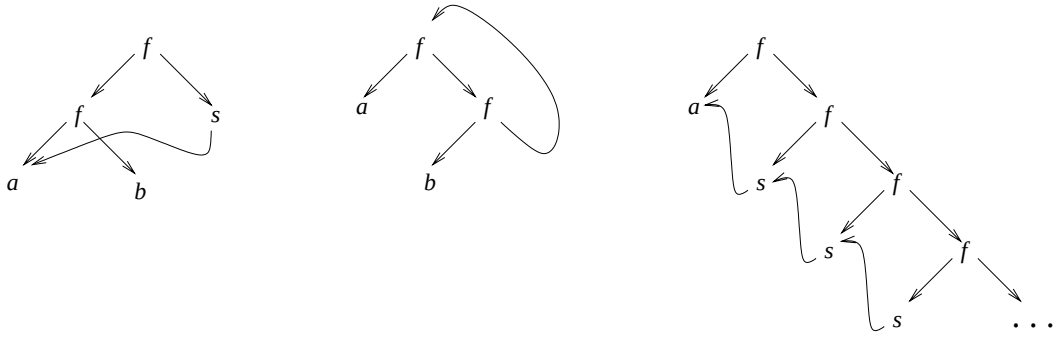
c'est-à-dire

$$\{\{(\varepsilon, a)\},$$
$$\{(\varepsilon, s),\ (1, a)\}$$
$$\{(\varepsilon, s),\ (1, a),\ (11, a)\}$$
$$\{(\varepsilon, s),\ (1, a),\ (11, a),\ (111, a)\}$$
$$\ldots$$
$$\{(\varepsilon, f),\ (1, a),\ (2, f),\ (21, s),\ (22, f), \ldots\}$$
$$\{(\varepsilon, f),\ (1, s),\ (2, f),\ (11, a),\ (21, s),\ (22, f), \ldots\}$$
$$\ldots\}$$

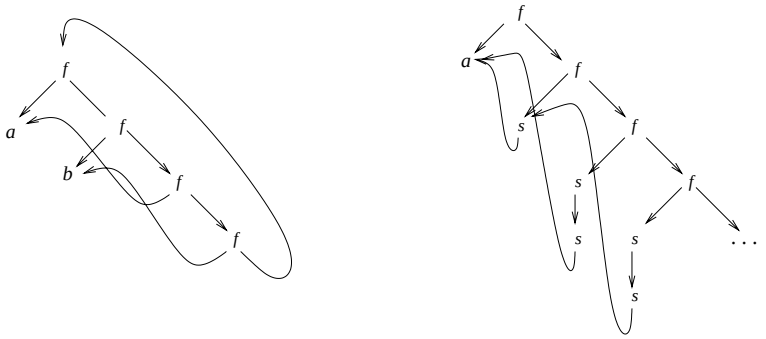
On remarque que parmi les arbres proposés, seul le premier arbre a un ensemble fini de nœuds, mais que le deuxième a quand même un ensemble fini (de formes) de sous-arbres et, que le troisième a un ensemble infini de nœuds ainsi qu'un ensemble infini de sous-arbres. Le premier est un arbre fini, le deuxième est un arbre infini rationnel et le dernier est un arbre infini non rationnel.

Définition 2.3 *Un arbre* rationnel *est un arbre dont l'ensemble des sous-arbres est fini.*

Un arbre rationnel peut toujours être représenté par un diagramme fini : il suffit de fusionner tous les nœuds d'où partent les mêmes sous-arbres. Un arbre non rationnel ne peut être représenté par un diagramme fini. Les trois exemples d'arbres peuvent être représentés respectivement par les diagrammes suivants :

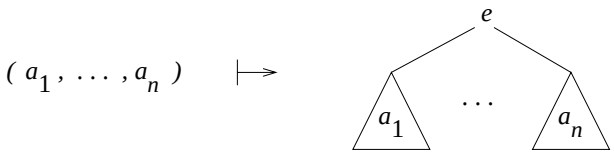


Si on ne fait pas toutes les fusions possibles, on peut obtenir des diagrammes différents. Un même arbre peut donc être représenté par différents diagrammes. Par exemple le deuxième et le troisième arbres peuvent aussi être représentés par :



2.1.2 Opérations de construction

On aimerait munir l'ensemble des arbres d'un ensemble d'opérations dite de *construction*, une pour chaque étiquette e. Elle se schématiserait comme suit, où n est l'arité de e :



Pour définir formellement ces opérations de constructions on les définit tout d'abord dans **D**, l'ensemble des ensembles de nœuds. Soit i un entier strictement positif. Si $d = (j_1 \dots j_k, e)$ est un nœud on désigne par $i{\cdot}d$ le nœud $(ij_1 \dots j_k, e)$. Si a est un ensemble de nœuds on désigne par $i{\cdot}a$ l'ensemble de nœuds $\{i{\cdot}d \,|\, d \in a\}$.

Définition 2.4 *Dans **D**, l'opération de* construction *associée à l'étiquette e d'arité n est l'application* $e^D : (a_1, \dots, a_n) \mapsto \{(\varepsilon, e)\} \cup 1{\cdot}a_1 \cup \dots \cup n{\cdot}a_n$.

Désignons par $\nu_k(a)$ l'ensemble des nœuds de a qui sont de profondeur k. Plusieurs remarques s'imposent concernant des éléments quelconques a, a_i, b de **D** :

Remarque 2.5 $a = b \;\leftrightarrow\; \bigwedge_{k=1}^{\infty} \nu_k(a) = \nu_k(b)$.

Remarque 2.6 Il existe un élément φ_0, indépendant des a_i, tel que $\nu_0(e^D(a_1, \dots, a_n)) = \varphi_0$.

Remarque 2.7 Pour tout $k \geq 0$, il existe une fonction φ_{k+1}, indépendante des $\nu_{k+1}(a_i)$, telle que $\nu_{k+1}(e^D(a_1, \dots, a_n)) = \varphi_{k+1}(\nu_k(a_1), \dots, \nu_k(a_n))$.

Remarque 2.8 Les éléments de $\nu_0(e^D(a_1, \dots, a_n))$ sont arborescents dans $e^D(a_1, \dots, a_n)$.

Remarque 2.9 Pour tout $k \geq 0$, les éléments de $\nu_{k+1}(e^D(a_1, \dots, a_n))$ sont arborescents dans $e^D(a_1, \dots, a_n)$ si et seulement si, pour chaque i, les éléments de $\nu_k(a_i)$ sont arborescents dans a_i.

2.1.3 L'algèbre des arbres et les algèbres voisines

On se donne maintenant un ensemble non vide **F** de symboles d'opérations. On considère que l'ensemble E d'étiquettes introduit au début de ce chapitre est égal à **F**. La lettre **N** désigne donc l'ensemble des nœuds étiquetés par les éléments de **F**. On désigne, par **A**, l'ensemble des éléments de **N** qui sont des arbres, par **B**, l'ensemble des éléments de **A** qui sont rationnels et, par **C**, l'ensemble des éléments de **A** qui sont finis (ont un ensemble fini de nœuds).

Si f est un élément de **F**, d'arité n, l'opération de construction f^D associée à f est une application de type $\mathbf{D}^n \to \mathbf{D}$. Soient a_i des ensembles de nœuds qui sont des arbres. Des remarques 2.8 et 2.9 on déduit que $f^D(a_1, \dots, a_n)$ est aussi un arbre. On peut donc introduire l'application

$$f^A : (a_1, \dots, a_n) \mapsto f^D(a_1, \dots, a_n), \text{ de type } \mathbf{A}^n \to \mathbf{A}.$$

D'autre part l'ensemble des sous-arbres de l'arbre $f^D(a_1, \dots, a_n)$ s'obtient en prenant l'union des ensembles de sous-arbres des différents a_i et en y ajoutant l'arbre $f^D(a_1, \dots, a_n)$, s'il n'y

figure pas déjà. De ce fait, si les a_i sont rationnels, l'arbre $f^D(a_1, \ldots, a_n)$ l'est aussi. On peut donc introduire l'application

$$f^B : (a_1, \ldots, a_n) \mapsto f^D(a_1, \ldots, a_n), \text{ de type } \mathbf{B}^n \to \mathbf{B}.$$

Enfin, si les ensembles a_i sont finis, l'ensemble $f^D(a_1, \ldots, a_n)$ l'est aussi. On peut donc introduire l'application

$$f^C : (a_1, \ldots, a_n) \mapsto f^D(a_1, \ldots, a_n), \text{ de type } \mathbf{C}^n \to \mathbf{C}$$

et conclure par:

Définition 2.10 *L'*algèbre des arbres*, l'*algèbre des arbres rationnels*, l'*algèbre des arbres finis *et l'*algèbre des ensembles de nœuds *sont respectivement les couples* $\langle \boldsymbol{A}, (f^A)_{f\in F}\rangle$, $\langle \boldsymbol{B}, (f^B)_{f\in F}\rangle$, $\langle \boldsymbol{C}, (f^C)_{f\in F}\rangle$ *et* $\langle \boldsymbol{D}, (f^D)_{f\in F}\rangle$.

Attention, pour parler de l'algèbre des arbres finis, il est nécessaire de supposer que **F** ait au moins un symbole d'arité nulle, afin que son domaine ne soit pas vide.

2.2 Théories

2.2.1 Syntaxe

Soit **V** un ensemble infini dénombrable dont les éléments sont appelés variables, soit **F** un ensemble infini de symboles de fonction et soit **R** un ensemble de symboles de relation. À chaque élément de **F** est associé un entier $n \geq 0$ et à chaque élément de **R** est associé un entier $m \geq 1$, leurs arités. À partir de **V**∪**F**∪**R** on définit l'ensemble **L** des *expressions* qui est l'ensemble $T \cup P$ avec T et P définis comme suit.

L'ensemble T des *termes* est l'ensemble des mots t qui sont de l'une des deux formes

$$v, ft_1 \ldots t_n,$$

avec v un élément de **V**, avec f un élément d'arité n de **F** et avec les t_i pris dans T, mais bien entendu de longueurs plus petites que celle de t.

L'ensemble P des *formules* est l'ensemble des mots o, construits à partir de $T \cup \mathbf{V} \cup \mathbf{R}$, de symboles logiques et des parenthèses, qui sont de l'une des treize formes :

$$\mathit{vrai}, \mathit{faux}, t_1 = t_2, rt_1 \ldots t_n, \neg(p), (p \wedge q), (p \vee q), (p \to q), (p \leftrightarrow q), (\exists x\, p), (\forall x\, p), (\exists ?x\, p), (\exists !x\, p) \tag{2.1}$$

avec les t_i pris dans T, avec $r \in \mathbf{R}$ d'arité n, avec x un mot construit sur **V** et avec p et q pris dans P, mais bien entendu de longueurs plus petites que celle de o.

Une occurrence de variable x_i dans une formule o est *liée* ou *libre* suivant qu'elle se produit ou ne se produit pas à l'intérieur d'une sous-formule de l'une des quatre dernières formes, avec x_i figurant dans x. Les *variables libres* de o sont des éléments de **V** qui ont au moins une occurrence libre dans o. Une *proposition* est une formule sans variables libres.

Remarquons que dans les quatre dernières formes, les quantificateurs portent sur des vecteurs de variables. Ceci est introduit par pure commodité syntaxique, sémantiquement une quantification $\exists x_1 \ldots x_n\, p$ est équivalente à $\exists x_1 \ldots \exists x_n\, p$ et bien entendu, l'ordre des variables n'est pas important, du fait que ces quantifications sont commutatives. La même chose est vraie pour le quantificateur $\forall$.

Nous avons introduit également les formes $\exists ?x\, p$ et $\exists !x\, p$, qui signifient informellement « il existe au plus une valeur pour chaque variable de x telle que p » et « il existe une et une seule valeur pour chaque variable de x telle que p ». Elles sont introduites également par pure commodité et donc **L** est essentiellement un langage classique du premier ordre.

Par commodité, nous adoptons les conventions suivants.

- Nous supposons que l'ensemble **V** soit totalement ordonné par une relation d'ordre $\preceq$. Nous écrivons $u \prec v$ pour signifier que $u \preceq v$ et u est différent de v.
- Nous nous permettons de remplacer des parenthèses par des crochets ou même de les supprimer lorsqu'il n'y a pas d'ambiguïté.
- Nous ne distinguons pas deux formules qui peuvent être rendues égales moyennant les transformations suivantes des sous-formules :

$$p \wedge \mathit{vrai} \implies p, \qquad p \wedge q \implies q \wedge p, \qquad (p \wedge q) \wedge r \implies p \wedge (q \wedge r).$$

- Nous écrivons $\bigwedge_{i=1}^{n} p_i$ pour $p_1 \wedge \cdots \wedge p_n \wedge \mathit{vrai}$, avec $n \geq 0$. Pour $n = 0$ on obtient donc *vrai*. De même nous écrivons $\bigvee_{i=1}^{n} p_i$ pour $p_1 \vee \cdots \vee p_n \vee \mathit{faux}$, avec $n \geq 0$. Pour $n = 0$ on obtient alors *faux*.

2.2.2 Sémantique

Comme il est habituel de le faire, on introduit la notion d'*interprétation* pour donner une définition rigoureuse des liens entre la signification d'une expression de **L** et la signification de ses différents composants. Pour ceci on introduit un ensemble non vide D, appelé *domaine* et on désigne par $\mathit{oprts}(D)$ l'ensemble des opérations que l'on peut définir dans D et par $\mathit{rels}(D)$ l'ensemble des relations que l'on peut définir sur D. On introduit aussi deux valeurs de vérité, **v** pour vrai et **f** pour faux. Chaque *opération* est une application de type $D^n \to D$, où n, l'arité de l'opération, est un entier positif ou nul. Lorsque $n = 0$ on assimile l'opération à un élément de D. Chaque *relation* est une application de type $D^m \to \{\mathbf{v}, \mathbf{f}\}$, où m, l'arité de la relation, est un entier positif.

Dans l'esprit d'un petit livre [35] de Roger Lyndon, on définit alors une *interprétation* I comme une application $e \mapsto I(e)$ de type

$$\mathbf{V} \cup \mathbf{F} \cup \mathbf{R} \cup \mathbf{L} \to D \cup \mathit{oprts}(D) \cup \mathit{rels}(D) \cup \{\mathbf{v}, \mathbf{f}\},$$

qui respecte les conditions

$$\begin{array}{lll} v \in \mathbf{V} & \text{entraîne} & I(v) \in D, \\ f \in \mathbf{F} & \text{entraîne} & I(f) \in \mathit{oprts}(D) \text{ et l'arité de } I(f) \text{ est celle de } f \\ r \in \mathbf{R} & \text{entraîne} & I(r) \in \mathit{rels}(D) \text{ et l'arité de } I(r) \text{ est celle de } r \end{array} \tag{2.2}$$

et les 15 conditions, correspondant aux 15 formes possibles d'expressions,

$$
\begin{array}{lcl}
I(v) & = & I(v), \\
I(f t_1 \ldots t_n) & = & I(f)(I(t_1), \ldots, I(t_n)), \\
I(\mathit{vrai}) & = & \mathbf{v}, \\
I(\mathit{faux}) & = & \mathbf{f}, \\
I(r t_1 \ldots t_n) & = & I(r)(I(t_1), \ldots, I(t_n)) \\
I(s = t) & = & \mathbf{v}, \text{ si } I(s) \text{ est égal à } I(t), \text{ et } \mathbf{f} \text{ sinon}, \\
I(\neg(p)) & = & \mathbf{v}, \text{ si } I(p) = \mathbf{f}, \text{ et } \mathbf{f} \text{ sinon}, \\
I(p \wedge q) & = & \mathbf{v}, \text{ si } I(p) = \mathbf{v} \text{ et } I(q) = \mathbf{v}, \text{ et } \mathbf{f} \text{ sinon}, \\
I(p \vee q) & = & \mathbf{v}, \text{ si } I(p) = \mathbf{v} \text{ ou } I(q) = \mathbf{v}, \text{ et } \mathbf{f} \text{ sinon}, \\
I(p \rightarrow q) & = & \mathbf{v}, \text{ si } I(p) = \mathbf{f} \text{ ou } I(q) = \mathbf{v}, \text{ et } \mathbf{f} \text{ sinon}, \\
I(p \leftrightarrow q) & = & \mathbf{v}, \text{ si } I(p) = I(q), \text{ et } \mathbf{f} \text{ sinon}, \\
I(\exists x\, p) & = & \mathbf{v}, \text{ si } \mathit{card}\,\{J \in \mathit{proche}(x, I) | J(p) = \mathbf{v}\} > 0, \text{ et } \mathbf{f} \text{ sinon}, \\
I(\forall x\, p) & = & \mathbf{v}, \text{ si } \mathit{card}\,\{J \in \mathit{proche}(x, I) | J(p) = \mathbf{f}\} = 0, \text{ et } \mathbf{f} \text{ sinon}, \\
I(\exists ? x\, p) & = & \mathbf{v}, \text{ si } \mathit{card}\,\{J \in \mathit{proche}(x, I) | J(p) = \mathbf{v}\} \leq 1, \text{ et } \mathbf{f} \text{ sinon}, \\
I(\exists ! x\, p) & = & \mathbf{v}, \text{ si } \mathit{card}\,\{J \in \mathit{proche}(x, I) | J(p) = \mathbf{v}\} = 1, \text{ et } \mathbf{f} \text{ sinon},
\end{array}
$$

où $\mathit{proche}(x, I)$ désigne l'ensemble des applications J qui diffèrent au plus de l'application I par le fait que pour certaines variables x_i de x on a $J(x_i) \neq I(x_i)$ et où, $\mathit{card}\, E$ désigne la cardinalité de l'ensemble E.

On remarque que, si on se donne une application I de type $\mathbf{V} \cup \mathbf{F} \cup \mathbf{R} \rightarrow D \cup \mathit{oprts}(D) \cup \mathit{rels}(D)$ qui respecte les conditions (2.2), alors il existe une seule interprétation qui la prolonge. Ceci est dû au fait que les 15 conditions astreignent la valeur $I(e)$ d'une expression e à être une fonction des valeurs $I(e_i)$ de ses sous-expressions immédiates e_i.

On remarque aussi que la valeur de vérité $I(p)$ d'une formule p ne dépend pas des valeurs $I(v)$ des éléments v de $\mathbf{V}$ dont toutes les occurrences dans p sont liées. En particulier si p est une proposition, $I(p)$ ne dépend pas des valeurs $I(v)$ des variables v. Elle dépend uniquement de la restriction J de I à $\mathbf{F} \cup \mathbf{R}$. Le couple $\mathcal{D} = \langle D, J \rangle$ est appelé *structure* ou, lorsque $\mathbf{R}$ est vide, *algèbre*.

Soit p une formule et P un ensemble (éventuellement vide ou infini) de formules. Si $I(p) = \mathbf{v}$, on dit que p *est vraie* dans I et que I *satisfait* p. On dit que I satisfait P si I satisfait chaque élément $p \in P$.

Soit $\mathcal{D} = \langle D, J \rangle$ une structure, p une formule et W un sous-ensemble de $\mathbf{V}$. Une *W-affectation*, est une application, de type $W \rightarrow D$. Une *W-solution*, de p dans $\mathcal{D}$ est une W-affectation, qui coïncide sur W avec une interprétation I qui satisfait p et qui est une extension de J.

Une *affectation* est une W-affectation, pour un sous-ensemble quelconque W de $\mathbf{V}$. Une *solution* de p dans $\mathcal{D}$ est une $\mathbf{V}$-solution de p dans $\mathcal{D}$. Une *W-base* de p est un ensemble S de W-solutions de p, tel que l'ensemble des $\mathbf{V}$-solutions de p est égal à l'ensemble des $\mathbf{V}$-affectations qui sont des extensions d'éléments de S.

2.2.3 Quelques propriétés d'une théorie

Une *théorie* est un ensemble (éventuellement vide ou infini) de propositions, appelées *axiomes* de la théorie. On dit que la formule p *est vraie dans la théorie* T et on écrit

$$T \models p, \tag{2.3}$$

si p est vraie dans toutes les interprétations qui satisfont T. On dit aussi que p *est fausse dans* T si $\neg p$ est vraie dans T. On remarque que si $x_1, \ldots, x_n$ sont les variables libres de p, la propriété (2.3) est équivalente à $T \models \forall x_1 \ldots x_n\, p$. On dit que les formules p et q sont *équivalentes dans la théorie* T ssi $T \models p \leftrightarrow q$. La théorie T est *complète* si quelle que soit la proposition p, on a soit $T \models p$ soit $T \models \neg p$. La structure $\mathcal{D} = \langle D, J\rangle$, où D est le domaine d'une interprétation I satisfaisant T et où J est la restriction de I à $\mathbf{F} \cup \mathbf{R}$, est un *modèle* de la théorie T. Pour une connaissance générale, on peut consulter [35, 4].

Dans une théorie T, nous avons des propriétés suivantes concernant des formules p, q quelconques, une formule $r[p]$ ayant une occurrence de p et la formule $r[q]$ obtenue en remplaçant cette occurrence de p par q.

Propriété 2.11 $T \models p \leftrightarrow q$ *entraîne* $T \models r[p] \leftrightarrow r[q]$.

La preuve de cette propriété se fait par induction sur la structure de la formule r.

Propriété 2.12 $T \models \exists ?x\, p$ *entraîne* $T \models (\exists x\, p \wedge \neg q) \leftrightarrow (\exists x\, p) \wedge \neg(\exists x\, p \wedge q)$.

Preuve Appelons les deux formules de l'équivalence p_1 et p_2. Soit I une interprétation satisfaisant T. Si $I(\exists x\, p) = \mathbf{f}$ alors p_1 et p_2 sont faux dans I. Supposons que $I(\exists x\, p) = \mathbf{v}$. Du fait que $T \models \exists ?x\, p$, il existe une interprétation unique $J \in$ *proche*(x, I) telle que $J(p) = \mathbf{v}$. Si $J(q) = \mathbf{f}$, $J(p \wedge \neg q) = \mathbf{v}$ d'où $I(p_1) = \mathbf{v}$. Comme J est unique, dans *proche*(x, I) il n'y a pas d'interprétation dans laquelle p et q sont vrais à la fois, donc $I(p_2) = \mathbf{v}$. Si $J(q) = \mathbf{v}$, dans *proche*(x, I) il n'y a pas d'interprétation dans laquelle p et $\neg q$ sont vrais à la fois, d'où $I(p_1) = \mathbf{f}$. Du fait que $J(p \wedge q) = \mathbf{v}$, $I(p_2) = \mathbf{f}$. Dans tous les cas, $I(p_1) = I(p_2)$. Comme I est choisi quelconque, la propriété est démontrée. □

Du fait que

$$(\exists !x\, p) \quad \text{ssi} \quad (\exists ?x\, p) \wedge (\exists x\, p)$$

on a :

Propriété 2.13 $T \models \exists !x\, p$ *entraîne* $T \models (\exists x\, p \wedge \neg q) \leftrightarrow \neg(\exists x\, p \wedge q)$.

Dans la propriété 2.12, la condition que $T \models \exists ?x\, p$ est vitale. Voici un contre exemple. Dans la théorie $\{(\exists u\; u = 1), (\exists v\; \neg(v = 1))\}$, la formule $\exists uv\; u = 1 \wedge \neg(v = 1)$ est vraie, c'est-à-dire

$$(\exists u\; u = 1), (\exists v\; \neg(v = 1)) \models \exists uv\; u = 1 \wedge \neg(v = 1)$$

Au contraire, la formule $(\exists uv\; u = 1) \wedge \neg(\exists uv\; u = 1 \wedge v = 1)$ est fausse dans la théorie. L'équivalence de la propriété 2.12 n'est donc pas valide. Ceci est dû au fait que la variable v n'est pas déterminée par la formule $u = 1$.

2.3 Théorie des arbres

La théorie des arbres **T** a, pour ensemble **F** de symboles de fonction, un ensemble infini comprenant au moins un symbole d'arité non nulle et, pour ensemble **R** de symboles de relation, l'ensemble vide. Elle consiste en l'ensemble infini de propositions de l'une des trois formes suivantes :

$$\forall x \forall y\, \neg(fx = gy) \tag{2.4}$$

$$\forall x \forall y \; fx = fy \rightarrow \bigwedge_i x_i = y_i \tag{2.5}$$

$$\forall x \exists! y \; \bigwedge_i y_i = t_i[yx] \tag{2.6}$$

où f, g sont des éléments distincts pris dans **F**, où x est un vecteur composé de variables x_i, où y est un vecteur composé de variables y_i, toutes distinctes, et où $t_i[yx]$ est un terme formé d'un élément de **F** suivi de variables prises dans x ou y.

Ces formes sont aussi appelées des schémas d'axiomes de la théorie. La propriété (2.4), dite *de conflit de symboles*, exprime que des opérations notées différemment produisent des individus distincts. La propriété (2.5), dite *d'explosion*, exprime en plus que la même opération, lorsqu'elle s'applique sur des objets distincts, produit des individus distincts. La propriété (2.6), dite *de solution unique*, exprime qu'une certaine forme de système d'équations admet une et une seule solution, lorsque certains paramètres (les x_i) sont fixés.

Michael Maher, en [36], a introduit cette théorie et a montré qu'elle était complète, c'est-à-dire que, quelle que soit la proposition p, on a soit $\mathbf{T} \models p$, soit $\mathbf{T} \models \neg p$. Il a aussi montré que pour toute formule p il existe une formule q, équivalente à p dans **T**, qui est une combinaison booléenne de conjonctions d'équations quantifiées existentiellement. Les propriétés de l'algorithme de résolution de contraintes d'arbres que nous présenterons plus loin permettrons de retrouver ces deux résultats.

Le nom « théorie des arbres » n'est pas pris par hasard, en effet :

Théorème 2.14 *L'algèbre des arbres, l'algèbre des arbres rationnels et l'algèbre des ensembles de nœuds sont des modèles de la théorie des arbres.*

Démonstration, partie I Montrons tout d'abord que l'algèbre des ensembles de nœuds est un modèle de la théorie des arbres, c'est-à-dire que, si on se place dans l'ensemble **D** des ensembles d'ensembles de nœuds et si on interprète chaque symbole d'opération f comme l'opération de construction f^D, toutes les propriétés de la théorie des arbres sont vraies. Ceci revient à montrer qu'on a à la fois:

$$(\forall a_1 \dots a_m \in \mathbf{D})(\forall b_1 \dots b_n \in \mathbf{D})\, \neg(f^D(a_1 \dots a_m) = g^D(b_1 \dots b_n)), \tag{2.7}$$

$$(\forall a_1 \dots a_n \in \mathbf{D})(\forall b_1 \dots b_n \in \mathbf{D})(f^D(a_1 \dots a_n) = f^D(b_1 \dots b_n) \; \rightarrow \; \textstyle\bigwedge_{i=1}^n a_i = b_i), \tag{2.8}$$

$$(\forall a_1 \dots a_m \in \mathbf{D})(\exists! b_1 \dots b_n \in \mathbf{D})(\textstyle\bigwedge_{i=1}^n b_i = t_i^D(b_1 \dots b_n, a_1 \dots a_m), \tag{2.9}$$

où $t_i^D(b_1 \dots b_n, a_1 \dots a_m)$ est un arbre de la forme $f^D(c_1, \dots, c_k)$, les c_i étant pris parmi les a_i et les b_i. Il est évident qu'on a (2.7) et (2.8). Il reste à montrer (2.9). Soient a_i et b_i des éléments de **D**. Du fait de la remarque 2.5 et en commutant les conjonctions multiples, la propriété

$$\bigwedge_{i=1}^{n} b_i = t_i^D(b_1 \dots a_m)$$

est équivalente à la propriété

$$\bigwedge_{k=0}^{\infty} \bigwedge_{i=1}^{n} \nu_k(b_i) = \nu_k(t_i^D(b_1 \dots a_m))$$

qui, du fait des remarques 2.6 et 2.7, est équivalente à une propriété de la forme

$$\bigwedge_{i=1}^{n} \nu_0(b_i) = \psi_0(i) \wedge (\bigwedge_{k=0}^{\infty} \bigwedge_{i=1}^{n} \nu_{k+1}(b_i) = \psi_{k+1}(i, \nu_k(b_1) \dots \nu_k(a_m)).$$

Les $\nu_0(b_i)$ ont donc des valeurs constantes et les valeurs des $\nu_{k+1}(b_i)$ ne dépendent que des valeurs des $\nu_k(b_j)$ et des a_j. Il s'ensuit que les valeurs des $\nu_k(b_i)$ ne dépendent que des valeurs des a_j. Il en va donc de même pour les valeurs des b_i et on a bien la propriété (2.9).

Démonstration, partie II Montrons que l'algèbre des arbres est un modèle de la théorie des arbres, c'est-à-dire qu'on a les propriétés (2.7'), (2.8'), (2.9') obtenues en remplaçant **N** par **A** dans (2.7), (2.8), (2.9). Du fait que **A** est un sous-enesmble de **D**, les propriétés (2.7) et (2.8) entraînent (2.7') et (2.8'). Pour montrer que la propriété (2.9) entraîne (2.9') il suffit de montrer qu'on a

$$(\forall a_1,\ldots,a_m,b_1,\ldots,b_n \in \mathbf{D})((\bigwedge_{i=1}^{n} b_i = t_i^D(b_1 \ldots a_m)) \wedge (\bigwedge_{i=1}^{m} a_i \in \mathbf{A}) \;\rightarrow\; (\bigwedge_{i=1}^{n} b_i \in \mathbf{A})). \tag{2.10}$$

Soient a, a_i, b, b_i des éléments de **D** et soit la notation

$$\text{arb}(a,b) \;\leftrightarrow\; \text{les éléments de } a \text{ sont arborescents dans } b.$$

La propriété

$$(\bigwedge_{i=1}^{n} b_i = t_i^D(b_1 \ldots a_m)) \wedge (\bigwedge_{i=1}^{m} a_i \in \mathbf{A}) \tag{2.11}$$

est équivalente à la propriété

$$(\bigwedge_{i=1}^{n} b_i = t_i^D(b_1 \ldots a_m)) \wedge (\bigwedge_{i=1}^{n} \text{arb}(\mathbf{N}, a_i)),$$

qui est équivalente à la propriété

$$(\bigwedge_{i=1}^{n} b_i = t_i^D(b_1 \ldots a_m)) \wedge (\bigwedge_{k=1}^{\infty} \bigwedge_{i=1}^{n} \text{arb}(\nu_k(\mathbf{N}), a_i))), \tag{2.12}$$

qui pour chaque $j \geq 0$ est équivalente à la propriétés

$$(\bigwedge_{i=1}^{n} b_i = t_i^D(b_1 \ldots a_m)) \wedge (\bigwedge_{k=1}^{\infty} \bigwedge_{i=1}^{n} \text{arb}(\nu_k(\mathbf{N}), a_i))) \wedge (\bigwedge_{i=1}^{m} \text{arb}(\nu_j(\mathbf{N}), b_i)). \tag{2.13}$$

En effet, du fait de la remarque 2.8, les propriétés (2.12) et (2.13) sont équivalentes pour $j = 0$ et, si on suppose que cette équivalence est vraie pour $j \geq 0$, d'après la remarque 2.9, on déduit qu'elle l'est aussi pour $j+1$. Il s'ensuit que la propriété (2.11) entraîne la propriété

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{\infty} \text{arb}(\nu_j(\mathbf{N}), b_i),$$

qui est équivalente à

$$\bigwedge_{i=1}^{n} b_i \in \mathbf{A}.$$

On a donc montré la véracité de (2.10).

Démonstration, partie III Enfin montrons que l'algèbre des arbres rationnels est un modèle de la théorie des arbres, c'est-à-dire qu'on a les propriétés (2.7"), (2.8"), (2.9") obtenues en remplaçant **A** par **B** dans les propriétés (2.7'), (2.8'), (2.9') précédemment introduites. Du fait que **B** est un sous-ensemble de **A**, les propriétés (2.7') et (2.8') entraînent (2.7") et (2.8"). D'autre part, dans la propriété (2.9'), les sous-arbres de profondeur k des b_i sont des b_j ou des sous-arbres de a_j. C'est vrai pour $k = 0$ et, si on suppose que c'est vrai pour k, on déduit immédiatement que c'est vrai pour $k+1$. Donc si les a_i sont rationnels les b_i le sont aussi et on a bien (2.9"). Ceci termine la preuve du théorème 2.14. □

Nous avons montré que l'algèbre des arbres et l'algèbre des arbres rationnels étaient des modèles de la théorie des arbres. Qu'en est-il de l'algèbre des arbres finis ? Du fait que **F** dispose d'au moins un symbole d'arité non nulle, la théorie des arbres comprend une proposition p de la forme $\exists! y \bigwedge_i y_i = fy$ avec f symbole d'arité non nulle. L'algèbre des arbres finis n'étant pas un modèle de p on conclut qu'elle n'est pas un modèle de la théorie des arbres.

Chapitre 3

Pouvoir d'expression des contraintes d'arbres

Sommaire

Ainsi que nous allons le voir dans ce chapitre, la négation et la quantification donnent aux contraintes d'arbres un pouvoir d'expression inattendu. Signalons que le contenu de ce chapitre a fait l'objet d'une publication [10].

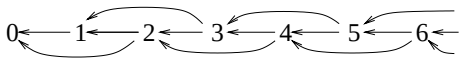
3.1 Longues imbrications de quantifications alternées

On introduit tout d'abord les notions de positions k-gagnantes et k-perdantes et on les illustre à travers trois exemples de jeux à deux adversaires. On montre comment exprimer, dans un domaine quelconque, l'ensemble des positions k-gagnantes d'un jeu par une contrainte. On termine la section par des contraintes d'arbre exprimant les positions k-gagnantes des trois exemples de jeux et faisant intervenir une imbrication de $2k$ quantifications alternées.

3.1.1 Positions k-gagnantes d'un jeu à deux adversaires

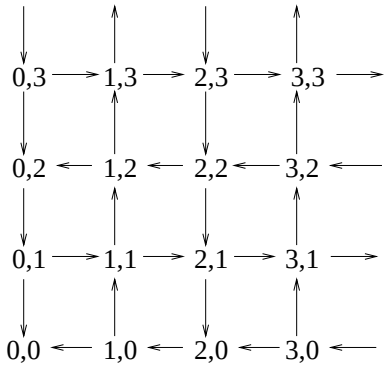
On se donne un graphe orienté (V, E) constitué d'un ensemble V de sommets et d'un ensemble $E \subseteq V \times V$ d'arêtes. On permet que V et E soient infinis et on appelle aussi *positions* les éléments de V. On considère le jeu à deux adversaires qui, étant donnée une position initiale x_0, consiste à tour de rôle à choisir une position x_1 telle que $(x_0, x_1) \in E$, puis une position x_2 telle que $(x_1, x_2) \in E$, puis une position x_3 telle que $(x_2, x_3) \in E$ et ainsi de suite. Le premier qui ne peut plus jouer a perdu et l'autre a gagné. Par exemple les trois graphes infinis suivants schématisent les jeux suivants :

Jeu 1



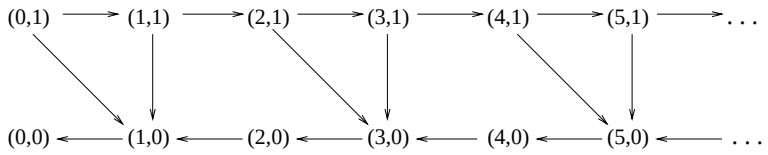
On se donne un entier positif ou nul i et à tour de rôle, on soustrait 1 ou 2 de i sans jamais le rendre strictement négatif. La première personne qui ne peut plus jouer a perdu.

Jeu 2



On se donne un couple (i, j) d'entiers positifs ou nuls et à tour de rôle, on choisit l'un des deux entiers i, j. Suivant que l'entier choisi u est impair ou pair, on augmente ou on diminue l'autre entier v de 1 sans jamais le rendre strictement négatif. La première personne qui ne peut plus jouer a perdu.

Jeu 3

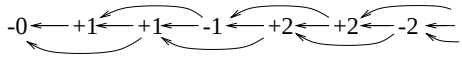


On se donne un couple (i, j) où j vaut 0 ou 1 et i est un entier positif ou nul. À tour de rôle, on modifie i et j en fonction de leur valeur. Si $j = 0$ on diminue i de 1. Si $j = 1$ et i est impair, on peut soit augmenter i de 1 et laisser j inchangé soit laisser i inchangé et diminuer j de 1. Si $j = 1$ et i est pair, on augmente i de 1 et soit on laisse j inchangé soit on diminue j de 1.

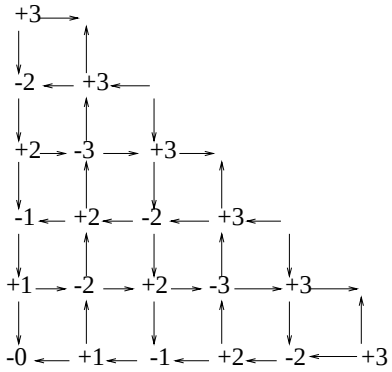
Soit $x \in V$ un sommet quelconque du graphe orienté (V, E) et supposons que ce soit au tour de la personne A de jouer. La position x est dite *k-gagnante* si, quelle que soit la façon de jouer de l'autre personne B, il est toujours possible que A gagne en jouant au plus k coups. Elle est dite *k-perdante* si, quelle que soit la façon de jouer de A, il existe toujours une façon de jouer pour B qui a pour effet que A perde et joue au plus k coups.

Considérons les trois graphes précédents et marquons $+k$ les positions qui sont k-gagnantes et $-k$ les positions qui sont k-perdantes, avec chaque fois k le plus petit possible. Le sommet 0 du premier graphe et les sommets $(0, 0)$ des deuxième et troisième graphes étant les seules positions 0-perdantes (aucune arête ne part de ces sommets), on les marque -0. En partant des

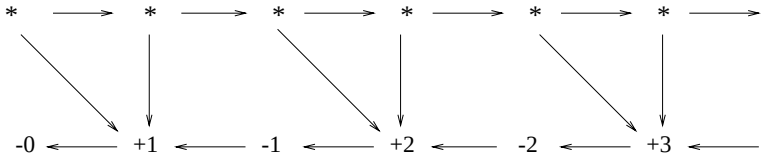
sommets marqués -0 et en remontant les flèches à l'envers on détermine, par vagues successives, les ensembles des sommets à marquer : $+1$, -1, $+2$, -2, $+3$, -3, ... On obtient pour le jeu 1



pour le jeu 2



et pour le jeu 3



Dans ce dernier dessin les sommets qui se trouvent sur la ligne haute sont ceux qui ne sont ni gagnants ni perdants. Donc en restant sur cette ligne haute, les adversaires sont assurés de ne pas perdre et de ne pas gagner.

On se convainc que l'ensemble des positions k-gagnantes du premier jeu est

$$\{i \in \mathcal{N} \mid i < 3k \text{ et } i \bmod 3 \neq 0\},$$

celui du deuxième est

$$\{(i,j) \in \mathcal{N}^2 \mid i + j < 2k \text{ et } (i+j) \bmod 2 = 1\},$$

et celui du troisième

$$\{(0,j) \in \mathcal{N}^2 \mid j < 2k \text{ et } j \bmod 2 = 1\},$$

où $\mathcal{N}$ est l'ensemble des entiers naturels.

3.1.2 Expression des positions k-gagnantes par une contrainte

Nous nous donnons maintenant un *domaine* D, c'est-à-dire un ensemble non vide, et un jeu à deux adversaires représenté par le graphe $G = (V, E)$, avec $V \subseteq D$. Nous allons exprimer les positions k-gagnantes de G par une contrainte dans D faisant intervenir une imbrication $\exists\forall\exists\ldots$ de $2k$ quantificateurs alternés.

Introduisons dans D les propriétés *coup*, *gagnant*$_k$ et *perdant*$_k$ définies par

$$\begin{array}{lcl} \mathit{coup}(x,y) & \stackrel{\text{def}}{=} & x \in D \text{ et } y \in D \text{ et } (x,y) \in E, \\ \mathit{gagnant}_k(x) & \stackrel{\text{def}}{=} & x \in D \text{ et } x \text{ est une position } k\text{-gagnante de } G, \\ \mathit{perdant}_k(x) & \stackrel{\text{def}}{=} & x \in D \text{ et } x \text{ est une position } k\text{-perdante de } G. \end{array} \tag{3.1}$$

On déduit que pour tout $k \geq 0$, on a les équivalences :

$$\begin{array}{lcl} \mathit{gagnant}_0(x) & \leftrightarrow & \mathit{faux}, \\ \mathit{gagnant}_{k+1}(x) & \leftrightarrow & \exists y\ \mathit{coup}(x,y) \wedge \mathit{perdant}_k(y), \\ \mathit{perdant}_k(x) & \leftrightarrow & \forall y\ \mathit{coup}(x,y) \rightarrow \mathit{gagnant}_k(y). \end{array} \tag{3.2}$$

Contrairement à ce que l'on pourrait croire, il s'ensuit que l'on a bien :

$$\mathit{gagnant}_k(x) \rightarrow \mathit{gagnant}_{k+1}(x)$$

et

$$\mathit{perdant}_k(x) \rightarrow \mathit{perdant}_{k+1}(x)$$

En effet, de la première et de la dernière définitions de (3.2), on conclut que ces implications sont vraies pour $k = 0$ et, si l'on suppose qu'elles sont vraies pour certain $k \geq 0$, des deux dernières définitions de (3.2) on conclut qu'elles le sont aussi pour $k+1$.

Dans la deuxième équivalence de (3.2), on peut remplacer l'occurrence de $\mathit{perdant}_k(y)$ par la formule de la troisième équivalence et on peut renommer x la variable quantifiée universellement sans interférences avec les occurrences libres de x. En se limitant à la quantification existentielle, la négation et la conjonction et en développant cette définition, on déduit une formulation explicite de $\mathit{gagnant}_k$, pour tout $k \geq 0$:

$$\mathit{gagnant}_k(x) \leftrightarrow \left[\begin{array}{l} \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \ldots \\ \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \mathit{faux} \qquad \underbrace{)\ldots)}_{2k} \end{array}\right] \tag{3.3}$$

où bien entendu toutes les quantifications portent sur des éléments de D. En descendant les négations, on obtient ainsi une imbrication de $2k$ quantificateurs alternés.

De la même façon, on déduit une formulation explicite de la relation $\mathit{perdant}_k$, pour tout $k \geq 0$:

$$\mathit{perdant}_k(x) \leftrightarrow \neg \left[\begin{array}{l} \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \ldots \\ \exists x\ \mathit{coup}(y,x) \wedge \neg(\\ \exists y\ \mathit{coup}(x,y) \wedge \neg(\\ \mathit{faux} \qquad \underbrace{)\ldots)}_{2k+1} \end{array}\right] \tag{3.4}$$

Dans l'équivalence (3.3) on peut utiliser une définition de *coup* plus générale que celle donnée en (3.1). On remarque que, pour tout entier k positif ou nul, on a la propriété suivante :

Propriété 3.1 *Soient trois graphes orientés de la forme $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ et $G = (V_1 \cup V_2, E_1 \cup E_2)$. Les graphes G_1 et G ont le même ensemble de positions k-gagnantes si à la fois :*

1. *les ensembles de sommets V_1 et V_2 sont disjoints,*
2. *pour tout $x \in V_2$, il existe $y \in V_2$ avec $(x, y) \in E_2$.*

En effet, de la première condition on déduit que E_1 et E_2 sont disjoints et donc que l'ensemble des positions k-gagnantes de G est l'union de l'ensemble des positions k-gagnantes de G_1 avec l'ensemble des positions k-gagnantes de G_2. Ce dernier ensemble est vide du fait de la deuxième condition.

Étant donné un graphe (V, E) d'un jeu, prenons V_1 l'ensemble V, V_2 le complément de V dans D, on conclut que :

Propriété 3.2 (Relation *coup* généralisée) *Les équivalences (3.3) et (3.4) sont aussi vraies pour toute relation coup respectant les trois conditions :*

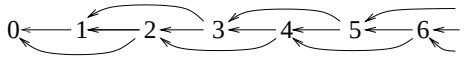
1. *pour tout $x \in V$ et $y \in V$, $coup(x, y) \leftrightarrow (x, y) \in E$,*
2. *pour tout $x \in D - V$ il existe $y \in D - V$ tel que $coup(x, y)$,*
3. *il n'existe pas $x \in V$ et $y \in D - V$ tels que $coup(x, y) \vee coup(y, x)$.*

3.1.3 Formalisation de jeux dans l'algèbre des arbres

Prenons maintenant pour domaine D l'ensemble des arbres **A** sur un ensemble de symboles d'opération **F**. Nous présentons dans ce qui suit la formalisation dans D des trois exemples de jeux introduits antérieurement.

Formalisation du jeu 1

Supposons que l'ensemble de symboles d'opérations **F** contienne les symboles $0, s$, d'arités respectives 0,1. Rappelons que le graphe du jeu est



Codons les sommets i de ce graphe par des arbres $s^i(0)$, avec bien entendu

$$s^0(0) = 0 \quad \text{et} \quad s^{i+1}(0) = s(s^i(0)).$$

Soit $G = (V, E)$ le graphe ainsi obtenu. Nous définissons $coup(x, y)$ avec $x \in \mathbf{A}$ et $y \in \mathbf{A}$ comme suit dans les différents cas.

- L'arbre x est de la forme $s(s(u))$. Bien entendu, l'arbre x représente un sommet du graphe G ssi u est un arbre $s^j(0)$ avec $j \geq 0$. Quel que soit l'arbre u, on prend $y = u$.
- L'arbre x est de la forme $s(u)$. Ici encore, l'arbre x représente un sommet du graphe G ssi u est de la forme $s^j(0)$, avec $j \geq 0$. Quel que soit l'arbre u, on prend $y = u$.
- La racine de l'arbre x n'est pas étiquetée par s et x n'est pas l'arbre 0. Dans ce cas on prend $y = x$.

Il est bien entendu que la relation *coup* définie suivant ces cas satisfait bien les conditions données dans la propriété 3.2. Cette relation *coup* généralisée se formule comme suit :

$$coup(x, y) \stackrel{\text{def}}{=} x = s(y) \vee x = s(s(y)) \vee (\neg(x = 0) \wedge \neg(\exists u\ x = s(u)) \wedge x = y)$$

Par conséquence, d'après la propriété 3.2, l'ensemble des positions k-gagnantes du jeu 1 est l'ensemble des solutions en x de la contrainte $\mathit{gagnant}_k(x)$ définie en (3.3). Par exemple, avec $k = 1$, la contrainte $\mathit{gagnant}_k(x)$ doit être équivalente à

$$x = s(0) \ \vee \ x = s(s(0))$$

et avec $k = 2$ à

$$x = s(0) \ \vee \ x = s(s(0)) \ \vee \ x = s(s(s(0)))) \ \vee \ x = s(s(s(s(0))))).$$

De la même façon, l'ensemble des positions k-perdantes du jeu 1 est l'ensemble des solutions en x de la contraintes $\mathit{perdant}_k(x)$ définie en (3.4). Par exemple, avec $k = 1$, la contrainte $\mathit{perdant}_k(x)$ doit être équivalente à

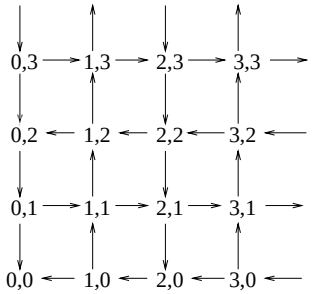
$$x = 0 \ \vee \ x = s(s(s(0)))$$

et avec $k = 2$, à

$$x = 0 \ \vee \ x = s(s(s(0))) \ \vee \ x = s(s(s(s(s(0)))))).$$

Formulation du jeu 2

Supposons que l'ensemble **F** de symboles d'opération ait notamment les symboles $0, f, g, c$, d'arités respectives 0,1,1,2. Rappelons que le graphe du jeu est



Codons les sommets (i, j) du graphe du jeu par les arbres $c(\overline{i}, \overline{j})$, avec $\overline{i} = (fg)^{\frac{i}{2}}(0)$ si i est pair, et $\overline{i} = g(\overline{i-1})$ si i est impair. Bien entendu,

$$(fg)^0(x) = x \quad \text{et} \quad (fg)^{i+1}(x) = f(g((fg)^i(x))).$$

Soit $G = (V, E)$ le graphe ainsi obtenu. En considérant toutes les formes possibles d'arbre $x \in \mathbf{A}$ et en suivant la même façon de raisonnement que dans le cas du jeu 1, on construit la relation *coup* généralisée dans l'algèbre des arbres infinis :

$$\mathit{coup}(x, y) \stackrel{\text{def}}{=} \mathit{transition}(x, y) \vee (\neg(\exists u \, \exists v \, x = c(u, v)) \wedge x = y)$$

avec

$$\textit{transition}(x,y) \stackrel{\text{def}}{=} \left[\begin{array}{l}\exists u\,\exists v\,\exists w \\ \left[\begin{array}{l}(x=c(u,v)\wedge y=c(u,w))\vee \\ (x=c(v,u)\wedge y=c(w,u))\end{array}\right] \\ \wedge \\ \left[\begin{array}{l}(\exists i\, u=g(i)\wedge \textit{suc}(v,w))\vee \\ (\neg(\exists i\, u=g(i))\wedge \textit{pred}(v,w))\end{array}\right]\end{array}\right]$$

$$\textit{suc}(v,w) \stackrel{\text{def}}{=} \left[\begin{array}{l}((\exists j\, v=g(j))\wedge w=f(v))\vee \\ (\neg(\exists j\, v=g(j))\wedge w=g(v))\end{array}\right]$$

$$\textit{pred}(v,w) \stackrel{\text{def}}{=} \left[\begin{array}{l}(\exists j\, v=f(j)\wedge \left[\begin{array}{l}(\exists k\, j=g(k)\wedge w=j)\vee \\ (\neg(\exists k\, j=g(k))\wedge w=v)\end{array}\right])\vee \\ (\exists j\, v=g(j)\wedge \left[\begin{array}{l}(\exists k\, j=g(k)\wedge w=v)\vee \\ (\neg(\exists k\, j=g(k))\wedge w=j)\end{array}\right])\vee \\ (\neg(\exists j\, v=f(j))\wedge\neg(\exists j\, v=g(j))\wedge \\ \neg(v=0)\wedge w=v)\end{array}\right]$$

D'après la propriété 3.2, l'ensemble des positions k-gagnantes du jeu 2 est l'ensemble des solutions en x de la contrainte $\textit{gagnant}_k(x)$ définie en (3.3). Par exemple, avec $k=1$, la contrainte $\textit{gagnant}_k(x)$ doit être équivalente à

$$x=c(g(0),0)\vee x=c(0,g(0)))$$

et avec $k=2$ à

$$\left[\begin{array}{l}x=c(0,g(0))\vee x=c(g(0),0)\vee \\ x=c(0,g(f(g(0))))\vee x=c(g(0),f(g(0)))\vee x=c(f(g(0)),g(0))\vee x=c(g(f(g(0))),0)\end{array}\right]$$

De même, l'ensemble des positions k-perdantes du jeu 2 est l'ensemble des solutions en x de la contrainte $\textit{perdant}_k(x)$ définie en (3.4). Par exemple, avec $k=1$, la contrainte $\textit{perdant}_k(x)$ doit être équivalente à

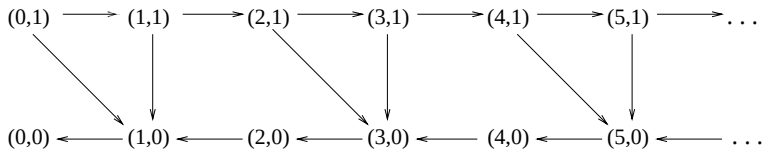
$$x=c(0,0)\vee x=c(0,f(g(0)))\vee x=c(f(g(0)),0)$$

et avec $k=2$ à

$$\left[\begin{array}{l}x=c(0,0)\vee x=c(0,f(g(0)))\vee x=c(f(g(0)),0)\vee \\ x=c(g(0),g(0))\vee x=c(0,f(g(f(g(0)))))\vee x=c(f(g(f(g(0)))),0)\vee x=c(f(g(0)),f(g(0)))\end{array}\right]$$

Formulation du jeu 3

Supposons que l'ensemble **F** de symbole d'opération contienne Reprenons le jeu 3 introduit dans la les symboles $0,1,f,g,c$, d'arités respectives 0,0,1,1,2. Le graphe du jeu est



Les sommets (i,j) du graphe du jeu sont codés par $c(\overline{i},0)$ et $c(\overline{i},1)$, avec $\overline{i} = (fg)^{\frac{i}{2}}(0)$ si i est pair, et $\overline{i} = g(\overline{i-1})$ si i est impair. Bien entendu

$$(fg)^0(x) = x \quad \text{et} \quad (fg)^{i+1}(x) = f(g((fg)^i(x))).$$

Soit $G = (V,E)$ le graphe ainsi obtenu. En considérant toutes les formes possibles d'un arbre $x \in \mathbf{A}$, on définit la relation *coup* généralisée dans l'algèbre des arbres comme suit :

$$\mathit{coup}(x,y) \stackrel{\text{def}}{=} \mathit{transition}(x,y) \vee (\neg(\exists u \exists v\ x = c(u,v)) \wedge x = y)$$

avec

$$\mathit{transition}(x,y) \stackrel{\text{def}}{=} \left[\begin{array}{l} \exists u_1 \exists v_1 \exists u_2 \exists v_2 \\ x = c(u_1,v_1) \wedge y = c(u_2,v_2) \wedge \\ \left[\begin{array}{l} (v_1 = 0 \wedge v_2 = v_1 \wedge \mathit{pred}(u_1,u_2) \\ \vee \\ (v_1 = 1 \wedge \left[\begin{array}{l} (\exists w\ u_1 = g(w) \wedge \left[\begin{array}{l} (u_2 = f(u_1) \wedge v_2 = v_1) \vee \\ (u_2 = u_1 \wedge v_2 = 0) \end{array}\right]) \vee \\ (\neg(\exists w\ u_1 = g(w)) \wedge u_2 = g(u_1) \wedge (v_2 = v_1 \vee v_2 = 0)) \end{array}\right]) \\ \vee \\ (\neg(v_1 = 0) \wedge \neg(v_1 = 1) \wedge u_2 = u_1 \wedge v_2 = v_1) \end{array}\right] \end{array}\right]$$

D'après la propriété 3.2, l'ensemble des positions k-gagnantes du jeu 3 est l'ensemble des solutions en x de la contrainte $\mathit{gagnant}_k(x)$ définie en (3.3). Par exemple, avec $k = 1$, la contrainte $\mathit{gagnant}_k(x)$ doit être équivalente à

$$x = c(g(0),0)$$

et avec $k = 2$ à

$$x = c(g(0),0) \vee x = c(g(f(g(0))),0)$$

L'ensemble des positions k-perdantes du jeu 3 est l'ensemble des solutions en x de la contrainte $\mathit{perdant}_k(x)$ définie en (3.4). Par exemple, avec $k = 1$, la contrainte $\mathit{perdant}_k(x)$ doit être équivalente à

$$x = c(0,0) \vee x = c(f(g(0)),0)$$

et avec $k = 2$ à

$$x = c(0,0) \vee x = c(f(g(0)),0) \vee x = c(f(g(f(g(0)))),0)$$

Contrairement aux jeux précédents, il existe des positions qui, quel que soit k, ne sont ni k-perdantes ni k-gagnantes.

3.2 Quasi universalité des contraintes d'arbres

Après toutes ces quantifications, nous abordons des contraintes tellement expressives que leur résolution devient quasi indécidable.

3.2.1 Une contrainte définissant un arbre fini énorme

On pose $\delta(k) = 2^{2^{\cdot^{\cdot^{2}}}}$, avec k occurrences de 2 et une association des exponentiations de haut en bas. Plus précisément on prend

$$\delta(0) = 1, \qquad \delta(k+1) = 2^{\delta(k)},$$

avec $k \geq 0$. La fonction δ croit d'une façon stupéfiante, puisque $\delta(0) = 1$, $\delta(1) = 2$, $\delta(2) = 4$, $\delta(3) = 16$, $\delta(4) = 65536$ et $\delta(5) = 2^{65536}$. L'entier $\delta(5)$ est supérieur à 10^{20000}, un nombre probablement bien supérieur au nombre d'atomes constituant l'univers et au nombre de nanosecondes qui se sont écoulées depuis sa création !

Pour construire une famille de contraintes définissant un arbre de taille $\delta(k)$, nous adoptons une idée de P. Mielniczuk [40] concernant les « feature trees » [3]. On construit une suite de triangles isocèles (h_i, b_i) dont les dimensions h_i, b_i de hauteur et de base sont telles que $b_i = 2^{h_i}$. En démarrant avec $h_0 = 0$ et en prenant chaque fois $h_{i+1} = b_i$, à la k-ième étape on obtient un triangle (h_k, b_k) dont la base b_k a pour taille la tour $\delta(k)$ de puissances de 2 de hauteur k.

Plus précisément, on suppose que l'ensemble d'arbres **A** est construit sur un ensemble **F** de symboles d'opération comprenant notamment les symboles $0, 1, 2, 3, s, f$ d'arités respectives $0, 0, 0, 0, 1, 4$. Pour $k \geq 0$ introduisons la contrainte :

$$\mathit{énorme}_k(x) \stackrel{\text{def}}{=} \exists z\ \mathit{triangle}_k(3, x, z, 0)$$

avec toujours pour $k \geq 0$,

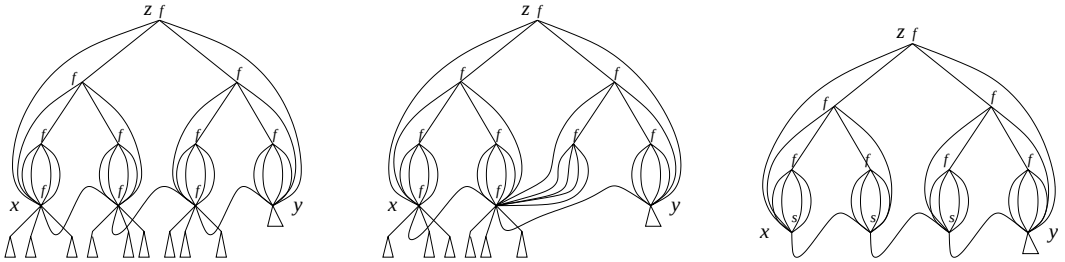
$$\begin{aligned}
\mathit{triangle}_0(t, x, z, y) &\stackrel{\text{def}}{=} z = x \wedge z = y \\
\mathit{triangle}_{k+1}(t, x, z, y) &\stackrel{\text{def}}{=} \left[\begin{array}{l}
[\exists u_1\ \exists u_2\ z = f(x, u_1, u_2, y)] \\
\wedge \\
\left[\begin{array}{l}
\forall t'\ \forall y'\ \forall z' \\
\left[\begin{array}{l}(t' = 1 \vee t' = 2) \wedge \\ \mathit{triangle}_k(t', z, z', y')\end{array}\right] \rightarrow \\
\left[\begin{array}{l}(t' = 1 \wedge \mathit{forme1}(y')) \vee \\ (t' = 2 \wedge \left[\begin{array}{l}\exists u\ \exists v\ \mathit{forme2}(u, y', v) \wedge \\ (t = 1 \rightarrow \mathit{trans1}(u, v)) \wedge \\ (t = 2 \rightarrow \mathit{trans2}(u, v)) \wedge \\ (t = 3 \rightarrow \mathit{trans3}(u, v))\end{array}\right])\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{aligned} \tag{3.5}$$

et

$$\begin{aligned}
\mathit{forme1}(x) &\stackrel{\text{def}}{=} \exists u_1 \ldots \exists u_4\ x = f(u_1, f(u_2, u_2, u_2, u_2), f(u_3, u_3, u_3, u_3), u_4) \\
\mathit{forme2}(x, z, y) &\stackrel{\text{def}}{=} \exists u_1 \ldots \exists u_6\ z = f(u_1, f(u_1, u_2, u_3, x), f(y, u_4, u_5, u_6), u_6) \\
\mathit{trans1}(x, y) &\stackrel{\text{def}}{=} \exists u_1 \ldots \exists u_4\ x = f(u_1, u_2, u_3, u_4) \wedge (y = u_2 \vee y = u_3) \\
\mathit{trans2}(x, y) &\stackrel{\text{def}}{=} \mathit{trans1}(x, y) \vee x = y \\
\mathit{trans3}(x, y) &\stackrel{\text{def}}{=} x = s(y)
\end{aligned} \tag{3.6}$$

Dans la contrainte $\mathit{triangle}_k$, chaque paramètre a une signification que nous allons étudier plus en détail dans la démonstration de la propriété 3.3 ci-dessous. Pour donner une première vision de ce que la contrainte $\mathit{triangle}_k(t, x, z, y)$ signifie, voici trois arbres x, z, y tels que

triangle$_2(t, x, z, y)$, avec, de gauche à droite, $t = 1$, $t = 2$ et $t = 3$:



Convenons que la taille $|p|$ d'une contrainte p est le nombre d'occurrences de tous les symboles à l'exception des parenthèses et des virgules. (Les contraintes pourraient être écrites en notation infixée.) On a la double propriété :

Propriété 3.3 (petite contrainte, gros arbre)

$$\boxed{|\textit{énorme}_k(x)| = 9 + 158k \qquad \text{et} \qquad \textit{énorme}_k(x) \leftrightarrow x = s^{\delta(k)-1}(0).}$$

Preuve Pour montrer l'égalité, il suffit de compter :

$$\begin{array}{lcl} |\textit{énorme}_k(x)| & = & |\textit{triangle}_k(t, x, z, y)| + 2, \\ |\textit{triangle}_0(t, x, z, y)| & = & 7, \\ |\textit{triangle}_{k+1}(t, x, z, y)| & = & |\textit{triangle}_k(t, x, z, y)| + (54 + 27 + 23 + 27 + 23 + 4) \end{array}$$

et de conclure.

Montrons maintenant l'équivalence dans l'algèbre des arbres. Écrivons $x\{f, k_1, ..., k_m\}y$ pour signifier que x est un arbre dont la racine est étiquetée f et qu'il existe $i \in \{k_1, \ldots, k_m\}$ tel que l'arbre y soit le i-ème fils de x. Convenons aussi que :

$$\begin{array}{lcl} x\{f, k_1, ..., k_m\}^0 y & \leftrightarrow & x = y, \\ x\{f, k_1, ..., k_m\}^{n+1} y & \leftrightarrow & \exists u\, x\{f, k_1, ..., k_m\}u \wedge u\{f, k_1, ..., k_m\}^n y \end{array}$$

avec $n \geq 0$.

Compte tenu de la définition de *énorme*$_k(x)$, pour montrer la deuxième partie de la propriété 3.3 il suffit de montrer que dans l'algèbre des arbres on a la dernière des trois équivalences :

$$\begin{array}{lcl} (\exists z\ \textit{triangle}_k(1, x, z, y)) & \leftrightarrow & x\{f, 2, 3\}^{\delta(k)-1} y \\ (\exists z\ \textit{triangle}_k(2, x, z, y)) & \leftrightarrow & \bigvee_{i=0}^{\delta(k)-1} x\{f, 2, 3\}^i y \\ (\exists z\ \textit{triangle}_k(3, x, z, y)) & \leftrightarrow & x\{s, 1\}^{\delta(k)-1} y \end{array} \tag{3.7}$$

Montrons par induction sur k que les trois équivalences sont vraies. Elles sont vraies pour $k = 0$. Supposons qu'elles soient vraies pour un certain $k \geq 0$ et montrons qu'elles sont vraies

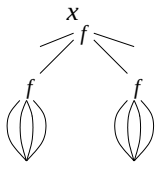
pour $k+1$. De (3.5) on déduit tout d'abord que

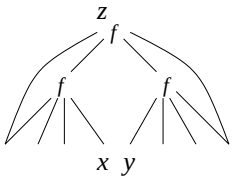
$$\textit{triangle}_{k+1}(t,x,z,y) \leftrightarrow \left[\begin{array}{l} [\exists u_1\, \exists u_2\, z = f(x,u_1,u_2,y)] \\ \wedge \\ \left[\begin{array}{l} \forall y' \\ (\exists z'\ \textit{triangle}_k(1,z,z',y')) \rightarrow \\ \textit{forme1}(y') \end{array}\right] \\ \wedge \\ \left[\begin{array}{l} \forall y' \\ (\exists z'\ \textit{triangle}_k(2,z,z',y')) \rightarrow \\ \left[\begin{array}{l} \exists u\, \exists v\, \textit{forme2}(u,y',v) \wedge \\ (t=1 \rightarrow \textit{trans1}(u,v)) \wedge \\ (t=2 \rightarrow \textit{trans2}(u,v)) \wedge \\ (t=3 \rightarrow \textit{trans3}(u,v)) \end{array}\right] \end{array}\right] \end{array}\right]$$

Du fait que l'on a supposé les équivalences (3.7) vraies pour k et en se servant des nouvelles notations, on a

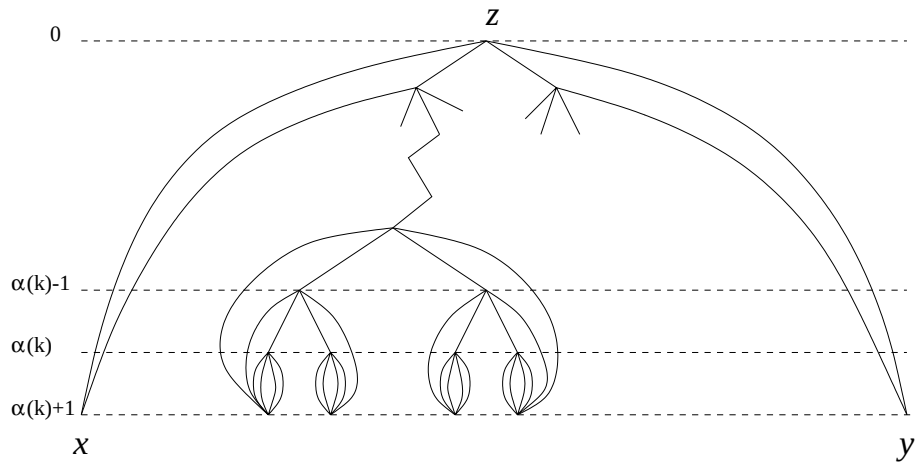
$$\textit{triangle}_{k+1}(t,x,z,y) \leftrightarrow \left[\begin{array}{l} [z\{f,1\}x \wedge z\{f,4\}y] \\ \wedge \\ \left[\begin{array}{l} \forall y' \\ z\{f,2,3\}^{\delta(k)-1}y' \rightarrow \\ \textit{forme1}(y') \end{array}\right] \\ \wedge \\ \left[\begin{array}{l} \forall y' \\ [\bigvee_{i=0}^{\delta(k)-1} z\{f,2,3\}^i y'] \rightarrow \\ \left[\begin{array}{l} \exists u\, \exists v\, \textit{forme2}(u,y',v) \wedge \\ (t=1 \rightarrow u\{f,2,3\}v) \wedge \\ (t=2 \rightarrow u\{f,2,3\}v \vee u=v) \wedge \\ (t=3 \rightarrow u\{s,1\}v) \end{array}\right] \end{array}\right] \end{array}\right]$$

du fait que le haut de l'arbre x satisfaisant $\textit{forme1}(x)$ et le haut de l'arbre z satisfaisant $\textit{forme2}(x,z,y)$ sont respectivement de la forme





le haut de l'arbre z satisfaisant $\mathit{triangle}\,(t, x, z, y)$ est de la forme



Il s'ensuit que

$$\exists z\ \mathit{triangle}_{k+1}(t,x,z,y) \;\leftrightarrow\; \exists z \left[\begin{array}{l} [z\{f,2\}^{\delta(k)+1}x \wedge z\{f,3\}^{\delta(k)+1}y] \\ \wedge \\ \left[\bigwedge_{i=0}^{\delta(k)} \left[\begin{array}{l}\forall y'\, z\{f,2,3\}^i y' \rightarrow \\ \left[\begin{array}{l}\exists u\, \exists v \\ y'\{f,2\}u \wedge y'\{f,3\}v\end{array}\right]\end{array}\right]\right] \\ \wedge \\ \left[\begin{array}{l}\forall y'\, \forall u\, \forall v \\ \left[\begin{array}{l} z\{f,2,3\}^{\delta(k)}y' \wedge \\ y'\{f,2\}u \wedge y'\{f,3\}v\end{array}\right] \rightarrow u=v\end{array}\right] \\ \wedge \\ \left[\bigwedge_{i=0}^{\delta(k)-1} \left[\begin{array}{l}\forall y'\, \forall u\, \forall v\, \forall u'\, \forall v' \\ \left[\begin{array}{l} z\{f,2,3\}^i y' \wedge \\ y'\{f,2\}u' \wedge u'\{f,3\}^{\delta(k)-i}u \wedge \\ y'\{f,3\}v' \wedge v'\{f,2\}^{\delta(k)-i}v \wedge\end{array}\right] \rightarrow \\ \left[\begin{array}{l}(t=1 \rightarrow u\{f,2,3\}v) \wedge \\ (t=2 \rightarrow u\{f,2,3\}v \vee u=v) \wedge \\ (t=3 \rightarrow u\{s,1\}v)\end{array}\right]\end{array}\right]\right] \end{array}\right]$$

Du fait que, dans un arbre binaire, le nombre de nœuds de profondeur n est égal à 2^n,

$$\exists z\ \mathit{triangle}_{k+1}(t,x,y,z) \;\leftrightarrow \exists u_1 \ldots \exists u_{\delta(k)} \left[\begin{array}{l} x=u_1 \wedge u_{\delta(k+1)}=y \wedge \\ \left[\bigwedge_{i=1}^{\delta(k+1)-1} \left[\begin{array}{l}(t=1 \rightarrow u_i\{f,2,3\}u_{i+1}) \wedge \\ (t=2 \rightarrow u_i\{f,2,3\}u_{i+1} \vee u_i=u_{i+1}) \wedge \\ (t=3 \rightarrow u_i\{s,1\}u_{i+1}\end{array}\right]\right]\end{array}\right]$$

On conclut qu'on a bien les équivalences (3.7) pour $k+1$, ce qui termine la démonstration. □

3.2.2 Expression d'un programme logique effectuant une multiplication

Soit $P(x,y)$ une formule faisant intervenir deux variables libres x et y. Si on modifie la formule $\mathit{triangle}_k(t,x,z,y)$ en posant

$$\mathit{trans3}(x,y) \stackrel{\text{def}}{=} x=y \vee P(x,y)$$

et si on introduit la formule

$$P_k^\star(x,y) \stackrel{\text{def}}{=} \exists z\, \mathit{triangle}_k(3,x,z,y)$$

on a alors

$$P_k^\star(x,y) \leftrightarrow \bigvee_{n=0}^{\delta(k)-1} \begin{bmatrix} \exists u_0 \ldots \exists u_n \\ x=u_0 \wedge \\ P(u_0,u_1) \wedge \\ P(u_1,u_2) \wedge \\ \ldots \\ P(u_{n-1},u_n) \wedge \\ u_n = y \end{bmatrix} \tag{3.8}$$

avec

$$|P_k^\star(x,y)| = 9 + k(158 + |P(x,y)|).$$

La relation binaire définie par $P_k^\star$ est en quelque sorte une fermeture transitive *limitée* de la relation binaire définie par P.

En partant du programme logique

$$\begin{array}{lcl} \mathit{fois}(i,j,k) & \leftarrow & \mathit{vrai}, \\ \mathit{fois}(s(i),j,k') & \leftarrow & \mathit{fois}(i,j,k) \wedge \mathit{plus}(j,k,k'), \\ \mathit{plus}(0,j,j) & \leftarrow & \mathit{vrai}, \\ \mathit{plus}(s(i),j,s(k)) & \leftarrow & \mathit{plus}(i,j,k), \end{array}$$

et en simulant son exécution par une machine Prolog, on arrive au résultat qui suit.

En prenant

$$P(x,y) \stackrel{\text{def}}{=} \begin{bmatrix} \exists i \exists j \exists k \exists k' \exists l \\ \begin{bmatrix} x = \mathit{sq}(\mathit{times}(0,j,0),l) \wedge \\ y = l \end{bmatrix} \vee \\ \begin{bmatrix} x = \mathit{sq}(\mathit{times}(s(i),j,k'),l) \wedge \\ y = \mathit{sq}(\mathit{times}(i,j,k), \mathit{sq}(\mathit{plus}(j,k,k'),l)) \end{bmatrix} \vee \\ \begin{bmatrix} x = \mathit{sq}(\mathit{plus}(0,j,j),l) \wedge \\ y = l \end{bmatrix} \vee \\ \begin{bmatrix} x = \mathit{sq}(\mathit{plus}(s(i),j,s(k)),l) \wedge \\ y = \mathit{sq}(\mathit{plus}(i,j,k),l) \end{bmatrix} \end{bmatrix}$$

et en introduisant la contrainte

$$\mathit{product}_{pq}(x) \stackrel{\text{def}}{=} P_5^\star(\mathit{sq}(\mathit{times}(s^p(0), s^q(0), x), \mathit{nil}), \mathit{nil})$$

de taille $1201 + p + q$, avec $p \geq 0$ et $q \geq 0$, on a

$$\mathit{product}_{pq}(x) \;\leftrightarrow\; x = s^{p \times q}(0)$$

sous condition que $p(q+2) + 1 \leq \delta(5)$.

Étant donnés deux entiers p et q, on est donc capable de représenter leur produit par une contrainte de taille linéaire en $p + q$. La seule restriction est que p et q soient plus petits que le nombre d'atomes de l'univers !

On peut procéder de même pour toute exécution de programme logique qui "s'arrête raisonnablement" et la remplacer par une contrainte dans les arbres.

3.2.3 Universalité versus complexité

À la place d'une machine Prolog on peut prendre une machine de Turing M, et exprimer par $P(x, y)$ le fait que M puisse passer de la configuration x à la configuration y en exécutant une instruction. On en conclut que :

Propriété 3.4 *Le résultat de l'exécution d'une machine de Turing, exécutant au plus $\delta(k)$ instructions, peut s'exprimer par une contrainte d'arbres, de dimension inférieure ou égale à un nombre proportionnel à k.*

Ici encore en prenant $k = 5$ on pourra exprimer tout ce que l'ordinateur le plus puissant pourrait calculer. Les contraintes d'arbres ont donc un pouvoir d'expression quasi universel ce qui a pour conséquence que la complexité des algorithmes pour les résoudre ne peut qu'être très élevée. Examinons ce point plus en détails dans le cas de contraintes sans variables libres.

Assimilons un algorithme à une machine de Turing M dont l'exécution se termine quel que soit le mot $x \in V^\star$ qui lui est fourni en entrée. La complexité de M est l'application de type $\mathcal{N} \to \mathcal{N}$:

$$n \mapsto \max \left\{ i \in \mathcal{N} \;\middle|\; \begin{array}{l} \text{il existe } x \in V^\star, \text{ avec } |x| = n, \text{ tel que } M \\ \text{exécute } i \text{ instructions, avec } x \text{ comme entrée.} \end{array} \right\}$$

Soit Φ_δ un ensemble de fonctions croissantes de type $\mathcal{N} \to \mathcal{N}$ telles que

1. les fonctions de la forme $n \mapsto an + f(bn)$, avec $a \in \mathcal{N}$, $b \in \mathcal{N}$ et $f \in \Phi_\delta$, appartiennent aussi à Φ_δ,
2. il existe un langage L, reconnaissable par une machine de Turing de complexité majorée par δ, mais par aucune machine de Turing de complexité majorée par un élément de Φ_δ, où δ est toujours la fonction définie au début de la section 3.2.1.

Lemme 3.5 *Soit T une machine de Turing permettant de décider si une contrainte d'arbres sans variables libres est vraie. La complexité de T ne peut être majorée par un élément de Φ_δ.*

Preuve Supposons qu'il existe une telle machine T de complexité majorée par un élément f de Φ_δ et montrons que nous aboutissons à une contradiction. Du fait que Φ_δ n'est pas vide le langage $L \subseteq V^\star$, intervenant au point 2 de la définition de Φ_δ, existe. D'après la propriété 3.4, à tout mot $x \in V^\star$, on peut alors associer une contrainte d'arbres p_x sans variables libres telle que

1. $x \in L$ si et seulement si p_x est vraie,
2. $|p_x| \leq b|x|$, pour une certaine constante $b \in \mathcal{N}$,
3. la transformation $x \mapsto p_x$ puisse être exécutée par une machine de Turing S de complexité majorée par $n \mapsto an$, où a est une constante. (Ce point pourrait être plus détaillé.)

En enchaînant les exécutions des machines S et T, on construit alors une machine M' qui reconnaît L et dont la complexité est majorée par $n \mapsto an + f(bn)$, une fonction qui par définition appartient à Φ_δ. Il y a donc contradiction sur les propriétés de L. □

Sergei Vorobyov [48] a montré, en utilisant la méthode de Compton et Henson [16], qu'aucun algorithme pour décider si une formule du premier ordre, sans variables libres, est vraie dans l'algèbre des arbres n'a une complexité majorée par une composition finie de fonctions élémentaires. À condition de montrer que pour ensemble Φ_δ on puisse prendre l'ensemble des fonctions, de type $\mathcal{N} \to \mathcal{N}$, obtenues par composition finie des fonctions élémentaires $n \mapsto$ cst, $+$, $\times$, $n \mapsto 2^n$, on retrouve le résultat de Sergei Vorobyov [48], mais à la façon de Pawel Mielniczuk [40] :

Propriété 3.6 *La complexité d'un algorithme, qui décide si une contrainte d'arbres sans variables libres est vraie, ne peut être majorée par une fonction obtenue par composition finie de fonctions élémentaires mentionnées ci-dessus.*

Chapitre 4

Résolution de contraintes d'arbres

Sommaire

Nous allons voir dans ce chapitre que nous avons un algorithme pour résoudre effectivement nos contraintes d'arbres. Des résultats de ce chapitre ont aussi été présentés dans notre papier [21].

4.1 Formes de base

À partir d'ici, on impose les conditions importantes suivantes sur une formule quelconque p :

- Les variables quantifiées de p sont aussi distinctes que possible.
- Pour toutes variables u, v et toute sous-formule q de p, si u et v ont des occurrences respectivement libres et liées dans q alors $u \prec v$.

Il est facile de remplir ces conditions en renommant les variables quantifiées. De plus, on utilise les notations suivantes :

- les lettres f, g, s désignent des symboles de fonction,
- les lettres u, v, w désignent des variables, les lettres x, y, z des vecteurs de variables, les lettres X, Y, Z l'ensemble des variables figurant respectivement dans x, y, z,

– la lettre t désigne un terme,
– les lettres a, b, c désignent des conjonctions d'équations, les lettre p, q, r des formules.

Toutes ces lettres peuvent être altérées par des primes ou des indices.

4.1.1 Conjonction d'équations

Définition 4.1 *Une* équation élémentaire *est une formule de la forme $u = t$, où u est une variable et t un terme à plat, c'est-à-dire de la forme v ou $f v_1 \dots v_n$, avec v et v_i des variables et f un symbole de fonction d'arité n.*

Une conjonction d'équations élémentaires *est une formule de la forme $v_1 = t_1 \wedge \cdots \wedge v_n = t_n \wedge$ vrai, avec $n \geq 0$, où les $v_i = t_i$ sont des équations élémentaires.*

Une conjonction d'équations élémentaires est résolue *si tous ses membres gauches sont distincts et si aucune de ses équations n'est de la forme $u = v$, où u et v sont des variables telles que $u \preceq v$.*

Exemple 4.1 Les équations $u = f v_1 v_2$ et $u = v$ sont des équations élémentaires, et les équations $f u = f v$ et $u = f g v$ ne le sont pas.

Dans ce qui suit $v_i = t_i$ sont des équations élémentaires et a, b des conjonctions d'équations élémentaires. Le fait qu'une conjonction résolue d'équations élémentaires ne contienne pas d'équations de la forme $u = v$ avec $u \preceq v$ assure que l'ensemble des équations de la conjonction ne contient pas de sous-ensembles de la forme

$$\{u_1 = u_2, \dots, u_{n-1} = u_n, u_n = u_1\}$$

avec $n \geq 1$. Un ensemble d'équations de cette forme est appelé circulaire, également interdit dans la procédure de décision de M. Maher [36]. Une conjonction résolue peut donc s'écrire

$$\bigwedge_{i=1}^{n} u_i = v_i \wedge \bigwedge_{j=1}^{m} w_j = t_j,$$

avec $n \geq 0$, $m \geq 0$, les u_i, v_i, w_j des variables et les t_j des terme faisant intervenir exactement un symbole fonctionnel, les ensembles $\{u_1, \dots, u_n\}$ et $\{v_1, \dots, v_n\}$ différents et les ensembles $\{u_1, \dots, u_n\}$ et $\{w_1, \dots, w_m\}$ disjoints. Par l'axiome de solution unique de **T**, les conjonctions résolues ont donc la propriété suivante :

Propriété 4.2 *Si a est une conjonction résolue et x la suite de ses membres gauches, alors $\boldsymbol{T} \models \exists! x\, a$.*

D'autre part, dans la théorie **T**, les conjonctions résolues ont la propriété importante suivante :

Propriété 4.3 *Soient a et b des conjonctions résolues. Si a et b ont les mêmes membres gauches et $\boldsymbol{T} \models a \to b$ alors $\boldsymbol{T} \models a \leftrightarrow b$.*

Preuve Soit x la suite des membres gauches et y la suite des autres variables figurant dans a et b dans un ordre quelconque. On a dans **T** la suite d'équivalences :

$$\begin{array}{lll}
 & \forall x y\ a \to b & \\
\leftrightarrow & \forall y x\ \neg a \vee b & \\
\leftrightarrow & \forall y (\neg(\exists x\ a \wedge \neg b)) & \\
\leftrightarrow & \forall y (\neg(\neg(\exists x\ a \wedge b))) & \text{d'après les propriétés 4.2 et 2.13} \\
\leftrightarrow & \forall y (\neg(\neg(\exists x\ b \wedge a))) & \\
\leftrightarrow & \forall y (\neg(\exists x\ b \wedge \neg a)) & \text{d'après les propriétés 4.2 et 2.13} \\
\leftrightarrow & \forall y x\ \neg b \vee a & \\
\leftrightarrow & \forall x y\ b \to a &
\end{array}$$

c'est-à-dire

$$\mathbf{T} \models (\forall xy\ a \to b) \leftrightarrow (\forall xy\ b \to a).$$

D'autre part, on a $\mathbf{T} \models a \to b$ ssi $\mathbf{T} \models \forall xy\ a \to b$. Par conséquent, le fait que $\mathbf{T} \models a \to b$ entraîne le fait que $\mathbf{T} \models b \to a$. □

Cette propriété a été établie par Alain Colmerauer [8], mais la démonstration présentée ci-dessus, utilisant la propriété 2.13 d'une théorie quelconque, est différente. Introduisons maintenant la notion d'équations et de variables accessibles à partir d'une variable dans une formule.

Définition 4.4 *Les équations et les variables* accessibles à partir de la variable u *de la formule* $\exists x \bigwedge_{i=1}^{n} v_i = t_i$ *sont celles qui figurent dans au moins une de ses sous-formules de la forme* $\bigwedge_{j=1}^{m} v_{k_j} = t_{k_j}$, *avec* v_{k_1} *étant la variable* u *et* $v_{k_{j+1}}$ *figurant dans* t_{k_j} *pour tout* $j \in 1..(m-1)$. *Les équations et les variables* accessibles *de cette formule sont celles qui sont accessibles à partir d'une variable qui ne figure pas dans* x.

Exemple 4.2 Dans la conjonction

$$\exists uvw\ z = fuv \wedge v = gvu \wedge w = fuv,$$

les équations $z = fuv$ et $v = gvu$ et les variables u et v sont accessibles et l'équation $w = fuv$ et la variable w ne le sont pas.

Du fait que dans la théorie $\mathbf{T}$, par la propriété d'explosion, une équation de la forme $u = t$ avec v figurant dans t entraîne que v est uniquement déterminé par rapport à u, les conjonctions d'équations élémentaires ont la propriété suivante :

Propriété 4.5 *Si toutes les variables figurant dans* x *sont accessibles dans* $\exists x\ a$ *alors* $\boldsymbol{T} \models \exists ?x\ a$.

Nous établissons ensuite la propriété très importante suivante.

Propriété 4.6 *Soient* x, x_i *des vecteurs de variables et* a_i *des conjonctions résolues d'équations élémentaires tels que toutes les variables de* x_i *soient accessibles dans* $\exists x_i\ a_i$. *Si chaque conjonction* a_i *contient une occurrence d'une variable de* x, *alors*

$$\boldsymbol{T} \models (\exists x \bigwedge_{i=1}^{n} \neg(\exists x_i\ a_i)) \tag{4.1}$$

Preuve Soient X et X_i les ensembles des variables figurant respectivement dans les vecteurs x et x_i. Du fait que les conjonctions a_i sont résolues, elles ne contiennent pas d'équations de la forme $u = u$ ou $u = v$, avec $u \in X$ et $v \in X_i$, car $u \prec v$. D'après les hypothèses, chaque conjonction a_i contient, pour une variable $u \in X$, soit une équation de la forme $u = fv_1 \ldots v_n$, soit une équation de la forme $u = v$, avec $v \notin X_i$ et v différent de u, soit une équation de la forme $v = t$ avec u figurant dans le terme t. Dans ce dernier cas, du fait que toutes les variables de x_i sont accessibles dans $\exists x_i\ a_i$, cette équation fait partie d'une sous-formule de la forme $\bigwedge_{j=1}^{k} v_j = t[y_j]$ avec v_1 ne figurant pas dans le vecteur xx_i et v_{j+1} figurant dans le vecteur y_j. Soit I une interprétation satisfaisant $\mathbf{T}$. Soit J une interprétation dans $\mathit{proche}(x, I)$ telle que pour une variable $u \in X$, les conditions suivantes soient satisfaites.

1. Si une conjonction a_i contient une formule de la forme $u = fv_1 \ldots v_n$ alors $J(u) \neq I(f)(I(v_1), \ldots, I(v_n))$, pour ceci il suffit d'associer à $J(u)$ l'interprétation d'un terme dont le symbole fonctionnel est différent de f,

2. Si une conjonction a_i contient une équation de la forme $u = v$, alors $J(u) \neq J(v)$.
3. Si une conjonction a_i contient une sous-formule de la forme $\bigwedge_{j=1}^{k} v_j = t[y_j]$ avec v_1 ne figurant pas dans le vecteur $x_i x$, avec v_{j+1} figurant dans le vecteur y_j et avec u figurant dans le vecteur y_k, alors $J(u)$ est différent de l'élément u déterminé par $I(v_1)$. (Ce point pourrait être plus détaillé.)

Par l'infinité de **F**, une telle interprétation J existe toujours. Par la propriété de conflit de symboles de la théorie **T**, on a $J(\exists x_j\, a_j) = \mathbf{f}$ pour $j \in 1..n$. Par conséquent,

$$I(\exists x \bigwedge_{i=1}^{n} \neg(\exists x_i\, a_i)) = \mathbf{v}.$$

Du fait que I satisfaisant **T** est choisie quelconque, la formule (4.1) est donc prouvée. □

Nous allons voir ultérieurement que cette propriété est cruciale, du fait qu'elle permet d'éliminer des quantificateurs dans la résolution. Nous illustrons cette propriété par l'exemple suivant.

Exemple 4.3 Considérons la formule

$$\exists v\ \neg(\exists w_1\ v = f w_1) \wedge \neg(\exists w_2\ u = f w_2 \wedge w_2 = f v).$$

Cette formule satisfait les conditions posées dans cette propriété. On voit bien que la formule $\exists w_1\ v = f w_1$ astreint v à être un terme dont le symbole fonctionnel est f et que, la formule $\exists w_2\ u = f w_2 \wedge w_2 = f v$ astreint v à être tel que $u = f f v$. Par conséquence, pour toute interprétation I satisfaisant **T**, on peut choisir une interprétation $J \in \mathit{proche}(v, I)$ telle que

$$J(v) = I(g)(\underbrace{I(a), \ldots, I(a)}_{n}),$$

avec $a \in \mathbf{F}$ d'arité nulle, avec $g \in \mathbf{F}$ différent de f et d'arité n et avec $I(f)(I(f)(J(v))) \neq I(u)$. Si **F** ne contient pas de symbole d'arité nulle, on peut choisir

$$J(v) = \underbrace{I(g)(I(g)(\underbrace{\ldots}_{n}), \ldots, I(g)(\underbrace{\ldots}_{n}))}_{n}$$

de telle façon que $I(f)(I(f)(J(v))) \neq I(u)$. On a donc

$$J(\exists w_1\ v = f w_1) = \mathbf{f} \quad \text{et} \quad J(\exists w_2\ u = f w_1 \wedge w_2 = v) = \mathbf{f},$$

d'où

$$I(\exists v\ \neg(\exists w_1\ v = f w_1) \wedge \neg(\exists w_2\ u = f w_1 \wedge w_2 = v)) = \mathbf{v}.$$

Ceci est vrai pour toute interprétation I satisfaisant **T**. On a donc bien

$$\mathbf{T} \models (\exists v \neg(\exists w_1\ v = f w_1) \wedge \neg(\exists w_2\ u = f w_2 \wedge w_2 = f v)).$$

4.1.2 Formule normalisée

Les formules considérées sont en général des formules logiques du premier ordre avec l'égalité comme seule relation. Ces formules sont mises sous une forme dite *normalisée*, qui

ne s'exprime qu'à l'aide des équations élémentaires, la constantes logique *vrai*, les opérateurs $\wedge$ et $\neg$ et le quantificateur existentiel $\exists$.

Définition 4.7 *Une* formule normalisée *de profondeur $d \geq 1$ est une formule de la forme*

$$p = \neg(\exists x\ a \wedge \bigwedge_{i=1}^{n} p_i), \qquad \textit{avec } n \geq 0, \tag{4.2}$$

où a est une conjonction d'équations élémentaires et les p_i des formules normalisées de profondeur d_i, avec $d = 1 + \max\{0, d_1, \ldots, d_n\}$.

Par simplicité d'écriture, nous omettons $\exists x$ si x est le vecteur vide. Il est facile de transformer une formule quelconque en une formule normalisée équivalente dans la théorie vide. La conjonction vide est *vrai*. Pour transformer les autres formules, il suffit par exemple de suivre les étapes suivants :

1. Introduire un supplément d'équations et de variables quantifiées existentiellement, pour éliminer les membres gauches et les sous-termes strictes des membres droits d'équations qui ne sont pas des variables.
2. Exprimer tous les quantificateurs, constantes et connecteurs logiques uniquement à l'aide de *vrai*, $\neg$, $\wedge$, $\exists$.
3. Enlever les doubles négations ($\neg\neg p$ devient p) à tous les niveaux.
4. Si la formule p ainsi obtenue ne commence pas par $\neg$, alors la remplacer par $\neg\neg p$.
5. Nommer les variables quantifiées par des noms différents de ceux de variables libres et aussi différents que possible de la façon qu'elles respectent la discipline de variables quantifiées.
6. Faire passer les quantifications au dessus des conjonctions, une formule $p \wedge (\exists x\ q)$ devient $\exists x\ p \wedge q$, car les variables libres de p sont distinctes de celles dans x.

Exemple 4.4 À titre d'exemple, prenons la formule :

$$fuv = fwu \vee \exists u \forall w\ u = fgw.$$

La transformation produit la suite suivante de formules, où la dernière est normalisée :

$$\begin{aligned}
&\Longrightarrow \quad \exists u_1\ u_1 = fuv \wedge u_1 = fwu \vee \exists u \forall w \exists u_2\ u = fu_2 \wedge u_2 = gw \\
&\Longrightarrow \quad \neg(\neg(\exists u_1\ u_1 = fuv \wedge u_1 = fwu) \wedge \neg(\exists u \neg(\exists w \neg(\exists u_2\ u = fu_2 \wedge u_2 = gw)))) \\
&\Longrightarrow \quad \neg(\neg(\exists u_1\ u_1 = fuv \wedge u_1 = fwu) \wedge \neg(\exists u_3 \neg(\exists u_4 \neg(\exists u_5\ u_3 = fu_5 \wedge u_5 = gu_4)))).
\end{aligned}$$

Si la formule de départ ne contient ni le connecteur $\leftrightarrow$ ni les quantificateurs ensemblistes $\exists ?$, $\exists !$, cette transformation se fait linéairement, c'est-à-dire qu'il existe une constante k telle que $n_2 \leq k n_1$, où n_1 est la taille de la formule à transformer et n_2 celle de la conjonction de formules normalisées obtenue.

4.1.3 Formule résolue

Définition 4.8 *Une* formule résolue *est de la forme*

$$\bigvee_{i=1}^{n} (\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\ b_{ij})), \qquad \textit{avec } n \geq 0,\ n_i \geq 0, \tag{4.3}$$

où a_i et les éventuels b_{ij} sont des conjonctions d'équations élémentaires, qui respectent les conditions suivantes :

1. *Les conjonctions a_i et $a_i \wedge b_{ij}$ sont résolues.*
2. *Toutes les variables et les équations de $\exists x_i\ a_i$ sont accessibles et toutes les variables et les équations de $\exists y_{ij}\ b_{ij}$ sont accessibles.*
3. *Aucun b_{ij} n'est égal à vrai.*

Lorsque $n = 0$, bien entendu la formule (4.3) est la constante logique *faux*. Soit $var(p)$ l'ensemble des variables qui ont une occurrence libre dans la formule p.

Définition 4.9 *L'ensemble S d'affectations est* visible *dans la formule résolue $p = p_1 \vee \cdots \vee p_n \vee$ faux, si*
(1) chaque p_i est de la forme $\exists x a$ et admet une et une seule $var(p_i)$-solution σ_i,
(2) S est l'ensemble des σ_i.

Définition 4.10 *Soit p une formule résolue. L'ensemble S d'affectations est* visible *dans la formule $\neg p$, si p est de la forme $(\neg p_1 \wedge \cdots \wedge \neg p_n \wedge vrai) \vee$ faux et si S est visible dans la formule résolue $p_1 \vee \cdots \vee p_n \vee$ faux.*

Exemple 4.5 Les formules suivantes sont résolues :

$$(u{=}0) \vee (\exists v\ u{=}sv \wedge v{=}0),$$
$$(u{=}0) \vee (\exists v\ u{=}sv),$$
$$(u{=}0) \vee (\exists v\ u = fvw \wedge \neg(\exists v_1\ v = gvv_1 \wedge v_1 = fv) \wedge \neg(\exists v_2\ w = gwv_2 \wedge v = fv_2)).$$

Parmi ces formules, seule la première a un ensemble visible de $\{u\}$-affectations.

Soit $p = p_1 \vee \cdots \vee p_n \vee$ *faux* une formule résolue, où chaque p_i est de la forme suivante, qui satisfait les conditions de la définition 4.8 :

$$\exists x\ a \wedge \bigwedge_{j=1}^{m} \neg(\exists y_j\ b_j). \tag{4.4}$$

Soit p_i une formule de cette forme. Soit W un sous-ensemble de **V** et soit $W^\star$ l'ensemble des variables qui appartiennent à W ou qui sont accessibles dans a à partir d'une variable de W. Soit $mbg(a)$ l'ensemble des membres gauches d'équations de la conjonction a. Soit $\mathcal{M}$ un modèle de la théorie **T**. Rappelons que X désigne l'ensemble des variables figurant dans le vecteur x. Montrons le lemme suivant :

Lemme 4.11 *Si $W^\star \subseteq mbg(a)$ alors p_i a une et une seule W-solution dans $\mathcal{M}$, sinon p_i a un ensemble infini de W-solutions. Si p_i a une et une seule $var(p_i)$-solution, alors p_i est de la forme $\exists x a$, où $mbg(a) = X \cup var(p_i)$.*

Preuve Du fait que toute variable de y_j est accessible dans $\exists y_j\ b_j$ et que b_j est différent de *vrai*, chaque conjonction b_j contient une équation de la forme $u_j = t_j$, avec $u_j \notin mbg(a)$ et $u_j \notin Y_j$. Soit V l'ensemble de ces variables u_j et des variables v_j si le terme t_j est la variable v_j. Du fait que les conjonctions b_j sont résolues, on a $v_j \prec u_j$, donc $v_j \notin Y_j$. Par suite des hypothèses, si $n > 0$ alors $V - mbg(a) \neq \emptyset$. Comme a est résolue, les conjonctions de la forme suivante sont résolues :

$$a \wedge (\textstyle\bigwedge_{u \in (V \cup W^\star) - mbg(a)} u{=}t), \tag{4.5}$$

où chaque terme t contient exactement un symbole fonctionnel, distinct de tous les autres symboles fonctionnels figurant dans la conjonction (4.5) et dans p_i. L'infinité de **F** assure l'existence de telles conjonctions. Pour chaque conjonction c de cette forme, il existe donc une interprétation I satisfaisant **T** telle que $I(c) = \mathbf{v}$.

D'après la construction de (4.5), pour tout $j \in 1..m$, on a

$$I(\textstyle\bigwedge_{u \in V - mbg(a)} u = t) = \mathbf{v} \text{ entraîne } I(\exists y_j\, b_j) = \mathbf{f}.$$

Toute interprétation qui satisfait (4.5) satisfait donc p_i. De plus, dans un modèle $\mathcal{M}$ de **T**, une W-solution de (4.5) est aussi une W-solution de p_i. La formule p_i a donc au moins une W-solution.

Si $W^\star \subseteq mbg(a)$, alors d'après la propriété 4.2, on a $\mathbf{T} \models \exists! z\, a'$, avec z un vecteur composé des variables dans $W^\star$ et, avec a' la conjonction des équations dans a dont le membre gauche figure dans $W^\star$. La formule p_i a donc une seule W-solution.

Si $W^\star \not\subseteq mbg(a)$, alors la conjonction $\bigwedge_{u \in (V \cup W^\star) - mbg(a)} u = t$ dans la forme (4.5) est différente de *vrai*. Du fait que **F** est infini, il existe une infinité de conjonctions de forme (4.5) telles que les W-solutions de deux conjonctions différentes soient différentes. Comme leur W-solution est aussi une W-solution de p_i, la formule p_i a donc une infinité de W-solutions.

Par conséquent, si p_i a une seule W-solution, avec $W = var(p_i)$, alors dans la forme (4.4), aucun b_j ne contient des équations dont le membre gauche est dans $var(p_i)$. Du fait que toutes les variables de X sont accessibles dans $\exists x\, a$ et appartiennent à $mbg(a)$, forcément $m = 0$. La formule p_i est donc de la forme $\exists x\, a$, où $mbg(a) = X \cup var(p_i)$. □

On déduit immédiatement les résultats suivants :

Propriété 4.12 *Soit p une formule résolue. Toute $var(p)$-base finie de p est visible dans p.*

Par exemple, la première formule de l'exemple 4.5 précédent a une $\{u\}$-base finie et visible, avec $\{u\}$ l'ensemble de ces variables libres.

Corollaire 4.13 *La seule formule résolue p telle que $\mathbf{T} \models \neg p$ est la formule faux.*

Cette propriété prouve un résultat similaire à celui dans les arbres finis de J.-L. Lassez et K. Marriott [33] et à celui dans les arbres rationnels de M. J. Maher et P. J. Stuckey [37]. Elle montre que l'ajout de négations ne permet pas de restreindre une formule ayant un ensemble infini de W-solutions en une formule ayant un ensemble fini de W-solutions. Il montre aussi un résultat qui ressemble à celui dans les arbres finis de [32] : les formules doivent avoir le même ensemble de variables restreintes (ici c'est l'ensemble W) et même nombre d'équations. Signalons que Maher et Stuckey ont montré dans [37] que dans le cas où l'ensemble **F** est fini et contient au moins un symbole d'arité plus grande ou égale à 2, le problème de décider si une expression de termes rationnels (une formule composée d'équations, de $\wedge$, $\vee$ et de $\neg$) est équivalente à une disjonction de conjonction d'équations est un problème co-NP-complet, et que le problème de décider si une expression représente un ensemble non vide d'arbres rationnels est NP-complet. Dans le cas où **F** est infini, pour des conjonctions d'équations et de diséquations, ces problèmes sont décidés en temps polynomial. Remarquons en fin que la forme résolue, en s'écrivant sous la forme normalisée, est

$$\neg \bigwedge_{i=1}^{n} \neg(\exists x_i\, a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\, b_{ij})).$$

4.2 Système de règles de réécriture

Nous présentons dans cette section le système de règles de réécriture de sous-formules, qui est le cœur de notre algorithme de résolution de contraintes dans **T**. Ces règles de réécriture fonctionnent sur des formules de travail, qui sont des formules normalisées, où les occurrences de négation $\neg$ portent un numéro qui indique les propriétés de la formule. Après avoir introduit les règles de réécriture, nous montrons qu'elles sont correctes, c'est-à-dire toute application des règles termine et conserve l'équivalence dans **T** ainsi que la forme de travail des formules.

4.2.1 Formule de travail

Définition 4.14 *Une* formule de travail *est une formule normalisée dont les occurrences de* $\neg$ *ont été remplacées par des* $\neg^k$ *avec* k *pris dans l'intervalle 0..4 et telle que toute occurrence de sous-formule de la forme*

$$p = \neg^k(\exists x\ a \wedge q), \qquad \text{avec } k > 0, \tag{4.6}$$

satisfasse aux k *premières conditions de la liste de conditions ci-dessous. Ici* a *est une conjonction d'équations,* q *une conjonction de formules de travail* $\bigwedge_{i=1}^{n} \neg^{k_i}(\exists y_i\ b_i \wedge q_i)$*, avec* $n \geq 0$ *et, dans les conditions,* a' *désigne la partie d'équations de la sur-formule de travail immédiate de* p*, si elle existe.*

1. *Si* a' *existe, alors* $\boldsymbol{T} \models a \rightarrow a'$ *et l'ensemble des membres gauches de* a' *est inclus dans celui de* a*.*
2. *La conjonction* a *est résolue.*
3. *Si* a' *existe, l'ensemble des équations de* a' *est inclus dans celui de* a*.*
4. *Toutes les équations et les variables de* $\exists x\, a$ *sont accessibles et si* $n > 0$*, l'ensemble des équations de* a *est strictement inclus dans celui de chaque* b_i*.*

Voici un exemple de formule de travail :

$$\neg^3 \left[\begin{array}{l} \exists v_1\ u = f\,v_1 \wedge \\ \quad \neg^2(\exists w_1\ u = f\,w_1 \wedge w_1 = v_1) \wedge \\ \quad \neg^3(\exists w_2\ u = f\,v_1 \wedge w_2 = f\,v_1) \wedge \\ \quad \neg^4(\exists w_3\ u = f\,v_1 \wedge v_1 = f\,w_3) \end{array}\right]$$

Qualifions d'*initiale* une formule de travail de la forme $\neg^0\neg^3 p$, où dans p, on a $k = 0$ pour toutes les occurrences $\neg^k$. Qualifions de *finale* une formule de travail de la forme $\neg^0 p$, où p est une conjonction de formules de profondeur inférieure ou égale à 2 et où $k = 4$ pour toutes les occurrences de $\neg^k$. En ce qui concerne la relation entre la forme de travail finale et la forme résolue, nous avons la propriété suivante :

Propriété 4.15 *Soit* p *une formule de travail finale de la forme*

$$\neg^0(\bigwedge_{i=1}^{n} \neg^4(\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg^4(\exists y_{ij}\ a_i \wedge b_{ij}))).$$

La formule suivante est une formule résolue équivalente à p *dans la théorie vide*

$$\bigvee_{i=1}^{n} (\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\ b_{ij})).$$

De cette propriété et du corollaire 4.13, on déduit que :

Corollaire 4.16 *Si p est de la forme $\bigwedge_{i=1}^{n} \neg^4(\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg^4(\exists y_{ij}\ a_i \wedge b_{ij}))$ et si $\mathbf{T} \models p$ alors p est la formule vrai, c'est-à-dire $n = 0$.*

4.2.2 Les règles de réécriture

La transformation des formules de travail est représentée sous la forme d'un système de 11 règles de réécriture de sous-formules. Ces règles transforment une conjonction de formules de travail de profondeur quelconque en une conjonction de formules finales équivalente dans **T**. L'application d'une règle $p_1 \Longrightarrow p_2$ à la formule p consiste à remplacer dans p une occurrence de la sous-formule p_1 par la formule p_2. Dans ce qui suit, les lettres u, v, w désignent des variables, les lettres x, y, z désignent des vecteurs de variables, les lettres a, b, c des conjonctions d'équations, la lettre p une formule de travail, la lettre q une conjonction de formules de travail, la lettre r une conjonction d'équations et de formules de travail. Ces lettres peuvent être altérées par des primes ou des indices.

$$
\begin{array}{lll}
1 & \neg^1(\exists x\ u{=}u \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ r) \\
2 & \neg^1(\exists x\ v{=}u \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u{=}v \wedge r) \\
3 & \neg^1(\exists x\ u{=}v \wedge u{=}t \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u{=}v \wedge v{=}t \wedge r) \\
4 & \neg^1(\exists x\ u{=}fv_1..v_n \wedge u{=}gw_1..w_m \wedge r) & \Longrightarrow \quad \textit{vrai} \\
5 & \neg^1(\exists x\ u{=}fv_1..v_n \wedge u{=}fw_1..w_n \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u{=}fv_1..v_n \wedge \bigwedge_{i=1}^{n} v_i{=}w_i \wedge r) \\
6 & \neg^1(\exists x\ r) & \Longrightarrow \quad \neg^2(\exists x\ r) \\
7 & \neg^3(\exists x\ a \wedge q \wedge \neg^0(\exists y\ r)) & \Longrightarrow \quad \neg^3(\exists x\ a \wedge q \wedge \neg^1(\exists y\ a \wedge r)) \\
8 & \neg^3(\exists x\ a \wedge q \wedge \neg^2(\exists y\ a' \wedge r)) & \Longrightarrow \quad \neg^3(\exists x\ a \wedge q \wedge \neg^3(\exists y\ a \wedge r)) \\
9 & \neg^3(\exists x\ a \wedge \neg^4(\exists y\ a) \wedge q) & \Longrightarrow \quad \textit{vrai} \\
10 & \neg^3(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg^4(\exists y_i\ b_i)) & \Longrightarrow \quad \neg^4(\exists x'\ a' \wedge \bigwedge_{i \in K} \neg^4(\exists y'_i\ b'_i[y'_i])) \\
11 & \neg^3\left[\begin{array}{l}\exists x\ a \wedge q[x] \wedge \\ \neg^4\left[\begin{array}{l}\exists y\ b \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists z_i\ c_i[xy])\end{array}\right]\end{array}\right] & \Longrightarrow \quad \left[\begin{array}{l}\neg^3(\exists x\ a \wedge q \wedge \neg^4(\exists y\ b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^3(\exists x_i y_i z_i\ c_i[x_i y_i] \wedge q[x_i])\end{array}\right]
\end{array}
$$

Dans ces règles, les variables u et v sont telles que $v \prec u$, les f, g sont des symboles fonctionnels distincts. Dans la règle 6, la conjonction des équations contenues dans r est résolue. Dans la règle 8, les conjonctions a et a' ont les mêmes membres gauches d'équations. Dans la règle 11, dans la conjonction $q[x_i]$, les occurrences de $\neg^k$ sont toutes remplacées par $\neg^0$. Dans la règle 10,

- aucune conjonction b_i n'est égale à a,
- (x', a') est le couple des variables et des équations accessibles de $\exists x\ a$,
- (y'_i, b'_i) est le couple des variables et des équations accessibles de $\exists y_i x''\ b_i$, avec x'' le vecteur des variables non accessibles étant le membre gauche d'une équation dans a,
- $K \subseteq \{1, .., n\}$ l'ensemble des indices tel que $i \in K$ ssi aucune variable de $x - x'x''$ ne figure dans b'_i.

Les règles de simplification d'équations 1–5 exploitent des axiomes de types (2.4) et (2.5) de la théorie des arbres. La règle 6 est triviale. La règle 7 de propagation, exprime que les

équations d'une formule, une fois mises sous forme résolues, sont propagées dans les sous-formules immédiates. La règle 8 de restauration, restaure les équations ayant été recopiées. L'élimination de quantificateurs est effectuée par la règle 10, où l'on considère les formules de profondeur 1 ou 2. La règle 11 distribue en fait des "ou" sur des "et".

Les indices sur les négations restreignent l'application des règles de réécriture. Bien entendu, à l'initialisation, toutes les occurrences de $\neg$ reçoivent l'indice 0, sauf la deuxième qui reçoit 3. Si la négation d'une des sous-formules immédiates porte l'indice 0, la règle 7 est applicable. Elle propage les équations dans cette sous-formule et change l'indice de la négation en 1. Sur les sous-formules commençant par $\neg^1$, les règles 1 et 5 sont applicables. Lorsque ces règles transforment la conjonction d'équations en une conjonction résolue, la règle 6 change l'indice de la sous-formule en 2. Remarquons que si une sous-formule porte l'indice 2, alors la sur-formule directe qui la contient porte l'indice 3. Sur celle dernière, la règle 8 est applicable. Elle remplace dans la sous-formule d'incide 2 les équations qui ont les mêmes membres gauches que les équations de la sur-formule par ces équations-là, et change la négation $\neg^2$ en $\neg^3$. Sur les sous-formules de la forme $\neg^3(\exists x\ a)$, la règle 10 est applicable, en considérant que dans cette règle, n vaut 0. Elle élimine les variables quantifiées et les équations inaccessibles, puis change la négation en $\neg^4$. Cette règle est encore applicable sur les sous-formules de la forme $\neg^3(\exists x\ a \wedge \bigwedge \neg^4(\exists y_i\ b_i))$, elle les transforme en des formules où toutes les négations sont $\neg^4$. Remarquons que la règle 10 ne simplifie que les sous-formules de profondeur 1 ou 2. Les sous-formules de profondeur 3 sont mises en conjonctions de formules de profondeur 1 ou 2 à l'aide de la règle 11.

Exemple 4.6 Pour mieux comprendre les règles et leur application, examinons quelques exemples.

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^0(\exists w_1\ v_1 = gw_1) \wedge \neg^0(\exists w_2\ u_2 = gw_2 \wedge w_2 = gu_3))$$

La règle 7 s'applique sur la formule

$$\neg^3 \left[\begin{array}{l} \exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \\ \quad \neg^1(\exists w_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge v_1 = gw_1) \wedge \\ \quad \neg^0(\exists w_2\ u_2 = gw_2 \wedge w_2 = gu_3) \end{array}\right]$$

L'application de la règle sur les deux équations ayant comme membres gauches v_1 simplifie la sous-formule avec $\neg^1$ en la constante *vrai*. La formule devient

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^0(\exists w_2\ u_2 = gw_2 \wedge w_2 = gu_3)).$$

Encore une fois, la règle 7 s'applique

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^1(\exists w_2\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge u_2 = gw_2 \wedge w_2 = gu_3)).$$

L'application de la règle 5 donne

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^1(\exists w_2\ v_1 = fu_1u_2 \wedge u_2 = gw_2 \wedge w_2 = u_1 \wedge w_2 = gu_3)).$$

L'application de la règle 3 donne

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^1(\exists w_2\ v_1 = fu_1u_2 \wedge u_2 = gw_2 \wedge w_2 = u_1 \wedge u_1 = gu_3)).$$

La conjonction d'équations de la sous-formule avec $\neg^1$ étant résolue, la règle 6 s'applique

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^2(\exists w_2\ v_1 = fu_1u_2 \wedge u_2 = gw_2 \wedge w_2 = u_1 \wedge u_1 = gu_3))$$

puis la règle 8 s'applique

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^3(\exists w_2\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge w_2 = u_1 \wedge u_1 = gu_3)).$$

La règle 10, s'appliquant à la sous-formule interne $\neg^3$, donne

$$\neg^3(\exists v_1\ v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge \neg^4(v_1 = fu_1u_2 \wedge u_2 = gu_1 \wedge u_1 = gu_3)).$$

Encore une fois, elle s'applique à la formule $\neg^3$ et on obtient

$$\neg^4(u_2 = gu_1 \wedge \neg^4(u_2 = gu_1 \wedge u_1 = gu_3)).$$

Exemple 4.7

$$\neg^3\begin{bmatrix}\exists v_1v_2v_3\ u{=}sv_1 \wedge v_2{=}fuv_3 \wedge \\ \quad \neg^3(\exists w_1\ u{=}sv_1 \wedge v_2{=}fuv_3 \wedge w_1{=}sv_3 \wedge v_3{=}fv_2v_3) \wedge \\ \quad \neg^3(\exists w_2\ u{=}sv_1 \wedge v_2{=}fuv_3 \wedge v_1{=}sw_2)\end{bmatrix}$$

La règle 10, en s'appliquant sur les sous-formule $\neg^3$ internes, donne

$$\neg^3\begin{bmatrix}\exists v_1v_2v_3\ u{=}sv_1 \wedge v_2{=}fuv_3 \wedge \\ \quad \neg^4(u{=}sv_1 \wedge v_2{=}fuv_3 \wedge v_3{=}fv_2v_3) \wedge \\ \quad \neg^4(\exists w_2\ u{=}sv_1 \wedge v_2{=}fuv_3 \wedge v_1{=}sw_2)\end{bmatrix}$$

puis en s'appliquant sur la formule $\neg^3$, donne

$$\neg^4(\exists v_1\ u{=}sv_1 \wedge \neg^4(\exists w_2\ u{=}sv_1 \wedge v_1{=}sw_2))$$

Exemple 4.8

$$\neg^3\begin{bmatrix}\neg^4(u = sv) \wedge \\ \neg^4(\exists w_1\ u = sw_1 \wedge w_1 = sv) \wedge \\ \neg^4(v = u \wedge \neg^4(v = u \wedge u = 0) \wedge \neg^4(\exists w_2\ v = u \wedge u = sw_2))\end{bmatrix}$$

L'application de la règle 11 sur cette formule donne

$$\begin{bmatrix}\neg^3(\neg^4(u = sv) \wedge \neg^4(\exists w_1\ u = sw_1 \wedge w_1 = sv) \wedge \neg^4(v = u)) \wedge \\ \neg^3(v = u \wedge u = 0 \wedge \neg^0(u = sv) \wedge \neg^0(\exists w_{11}\ u = sw_{11} \wedge w_{11} = sv)) \wedge \\ \neg^3(\exists w_2\ v = u \wedge u = sw_2 \wedge \neg^0(u = sv) \wedge \neg^0(\exists w_{12}\ u = sw_{12} \wedge w_{12} = sv))\end{bmatrix}$$

4.2.3 Correction des règles de réécriture

Propriété 4.17 *Toute application répétée de ces règles de réécriture sur une formule de travail initiale p se termine et produit une formule de travail finale, équivalente à p dans la théorie **T**.*

Preuve Du fait qu'une formule de travail initiale commence par $\neg^0\neg^3$ et les autres occurrences de négation sont $\neg^0$, tout au long de l'application des règles de réécriture, en descendant le long des négations, on trouve $\neg^0$ puis une suite de $\neg^3$ suivie d'une suite de $\neg^k$, où les négations $\neg^4$ apparaissent au plus deux fois et à la fin de l'imbrication. Tant que les sous-formules sont encore de profondeur plus grande que 2, ou que les indices de négation ne valent pas tous 4, les règles sont encore applicables. Par conséquent, lorsque les règles ne peuvent plus s'appliquer, on obtient $\neg^0 q$, où q est une conjonction de formules de profondeur 2 et où les autres négations sont toutes d'indice 4. C'est donc une formule de travail finale.

Il reste à prouver que l'application des règles de réécriture termine toujours et que ces règles conservent l'équivalence dans **T** et la forme de travail des formules.

Première partie : L'application des règles termine toujours. Du fait que les variables respectent l'ordre $\prec$ strict et total, on peut numéroter les variables par des entiers positifs, de telle façon que le numéro de la variable u soit plus grand que celui de la variable v ssi $v \prec u$. Soit p une formule de travail. On définit les nombres n_i entiers non négatifs, avec $i = 1..8$, comme suit.

- n_1, la valeur $\alpha(p)$ où la fonction α est définie comme suit :

$$\alpha(\mathit{vrai}) = 0,$$
$$\alpha(\neg(\exists x\, a \wedge p_1 \wedge \cdots \wedge p_n)) = 2^{\alpha(p_1)+\cdots+\alpha(p_n)}$$

  Cette fonction est monotone pour la substitution, c'est-à-dire que $\alpha(p_2) < \alpha(p_1)$ entraîne $\alpha(p[p_2]) < \alpha(p)$ où $p[p_2]$ est la formule obtenue de p lorsque l'on remplace l'occurrence de la formule p_1 dans p par p_2.
- n_2, le nombre d'occurrences de $\neg^0$,
- n_3, le nombre d'occurrences de $\neg^1$,
- n_4, le nombre d'occurrences de symboles fonctionnels dans les sous-formules $\neg^1(\ldots)$,
- n_5, la somme des numéros de variables dans les sous-formules $\neg^1(\ldots)$,
- n_6, le nombre d'équations de la forme $v = u$ avec $v \prec u$ dans les sous-formules $\neg^1(\ldots)$,
- n_7, le nombre d'occurrences de $\neg^2$,
- n_8, le nombre d'occurrences de $\neg^3$.

Pour chaque règle, il existe un indice i telle que l'application de la règle laisse inchangés les n_j, avec $1 \leq j < i$, et diminue n_i. Ces i valent 1 pour les règle 4, 9 et 11, vaut 2 pour la règle 7, vaut 3 pour la règle 6, vaut 4 pour la règle 5, valent 5 pour les règles 1 et 3, vaut 6 pour la règle 2, vaut 7 pour la règle 8 et vaut 8 pour la règle 10. Du fait que les n_i sont tous entiers non négatifs, ils ne peuvent pas être diminués infiniment. Par conséquent, l'application des règles se termine.

Deuxième partie : Les règles conservent l'équivalence dans T et la forme de travail des formules. Les règles 1, 2, 3, 6, 7 et 9 sont correctes dans la théorie vide et donc dans toute théorie. On peut vérifier aisément qu'elles conservent aussi la forme de travail de formules. La règle 4 est correcte par l'axiome de conflit de symboles de la théorie **T**. La règle 5 est aussi correcte, par l'axiome d'explosion de **T**. Considérons la règle 8 :

$$\neg^3(\exists x\, a \wedge q \wedge \neg^2(\exists y\, a' \wedge r)) \implies \neg^3(\exists x\, a \wedge q \wedge \neg^3(\exists y\, a \wedge r))$$

Ici a et a' ont les mêmes membres gauches d'équations. Remarquons que lorsque les équations de la conjonction a sont propagées dans une sous-formule, et que cette sous-formule ne devient pas *vrai*, alors les membres gauches des équations de a restent toujours comme membres gauches d'équations dans la sous-formule. Par conséquence, la conjonction a' existe. Par les propriétés de $\neg^2$, la formule gauche de la règle est de la forme

$$\neg(\exists x\, a \wedge q \wedge \neg(\exists y\, a' \wedge b \wedge q'))$$

où $\mathbf{T} \models a' \wedge b \rightarrow a$. On a donc

$$\mathbf{T} \models a' \wedge b \rightarrow a \wedge b.$$

Du fait que les conjonctions $a \wedge b$ et $a' \wedge b$ sont résolues et ont les mêmes membres gauches d'équations, d'après la propriété 4.3,

$$\mathbf{T} \models a' \wedge b \leftrightarrow a \wedge b.$$

Les équations de la conjonction a' peuvent donc être remplacées par celle de la conjonction a tout en gardant l'équivalence de formules. La règle 8 est donc correcte.

Considérons la règle 10

$$\neg^3(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg^4(\exists y_i\ b_i)) \implies \neg^4(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg^4(\exists y'_i\ b'_i[y'_i]))$$

Rappelons que le vecteur x est décomposé en trois sous-vecteurs distincts x', x'' et x''', la conjonction a est décomposée en deux sous-conjonctions distinctes a' et a'' et que

- aucune conjonction b_i n'est égale à a,
- (x', a') est le couple des variables et des équations accessibles dans $\exists x\ a$,
- x'' est composé des membres gauches de a'',
- (y'_i, b'_i) est le couple des variables et des équations accessibles dans $\exists y_i x''\ b_i$, et
- $K \subseteq \{1, \ldots, n\}$ tel que $i \in K$ ssi aucune variable de x''' ne figure dans b'_i.

Du fait que toutes les équations de a' sont accessibles, si une variable y figure, elle est aussi accessible. On déduit qu'aucune variable du vecteur $x''x'''$ ne figure dans a'. D'après la propriété 4.2, $\mathbf{T} \models \exists! x''\ a''$. Dans le cas où $n = 0$, dans la théorie $\mathbf{T}$, la formule $\neg(\exists x'\ a')$ est résolue et est équivalente à $\neg(\exists x\ a)$. La règle est donc correcte si $n = 0$.

Si $n > 0$, d'après la propriété 2.13, la formule gauche de cette règle est équivalente dans $\mathbf{T}$ à la formule suivante

$$\neg(\exists x'x'''\ a' \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i x''\ a'' \wedge b_i)).$$

Du fait que les équations dans a'' figurent aussi dans b_i, on peut enlever a'' des sous-formules. Appliquons cette même règle dans le cas où $n = 0$ sur les sous-formules $\neg(\exists y_i x''\ b_i)$, nous obtenons des formules résolues équivalentes $\neg(\exists y'_i\ b'_i)$. La formule est donc équivalente à

$$\neg(\exists x'x'''\ a' \wedge \bigwedge_{i=1}^{n} \neg(\exists y'_i\ b'_i)).$$

L'ensemble des indice $1..n$ est décomposé en deux sous-ensemble distincts K et $\{1..n\} - K$ tels que, $i \in K$ ssi aucune variable de x''' ne figure dans b'_i. Du fait que les variables de x''' ne figurent pas dans a', cette formule est équivalente à

$$\neg(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg(\exists y'_i\ b'_i) \wedge (\exists x''' \bigwedge_{i\in\{1..n\}-K} \neg(\exists y'_i\ b'_i))).$$

On remarque que dans les formules $\neg(\exists y'_i\ b'_i)$, toutes les équations et toutes les variables sont accessibles. La conjonction de ces formules avec $i \in \{1..n\} - K$ satisfait donc les conditions de la propriété 4.6 et

$$\mathbf{T} \models \exists x''' \bigwedge_{i\in\{1..n\}-K} \neg(\exists y'_i\ b'_i).$$

La formule gauche est donc équivalente à

$$\neg(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg(\exists y'_i\ b'_i)).$$

Il ne reste qu'à montrer que dans cette formule, tous les b'_i sont différents de a. Du fait que toutes les variables de y_i et les équations de b_i sont accessibles dans $\exists y_i\ b_i$, du fait que b_i est

résolue et différent de a, il existe une variable $u \in \mathit{mbg}(b_i) - \mathit{mbg}(a)$, qui ne figure pas dans y_i et qui n'est pas accessible depuis les variables de x''. Par conséquence, elle est accessible dans $\exists y_i x''\ b_i$. Elle figure donc dans $\mathit{mbg}(b'_i) - \mathit{mbg}(a)$. La conjonction b'_i est donc différent de a. En mettant les indices appropriés, la formule devient donc la formule droite de la règle 10. Cette règle est alors correcte.

Considérons la règle 11 :

$$\neg^3 \left[\begin{array}{l} \exists x\ a \wedge q[x] \wedge \\ \neg^4 \left[\begin{array}{l} \exists y\ b \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists z_i\ c_i[xy]) \end{array}\right] \end{array}\right] \implies \left[\begin{array}{l} \neg^3(\exists x\ a \wedge q \wedge \neg^4(\exists y\ b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^3(\exists x_i y_i z_i\ c_i[x_i y_i] \wedge q[x_i]) \end{array}\right]$$

La formule gauche de la règle peut s'écrire

$$\neg(\exists x\ a \wedge q[x] \wedge \neg(\exists y\ b \wedge \neg \bigvee_{i=1}^{n} (\exists z_i\ c_i[xy]))) \tag{4.7}$$

D'après la condition de $\neg^4$ et d'après la propriété 4.5, on a $\mathbf{T} \models \exists ? y\ b$. D'après la propriété 2.12,

$$\mathbf{T} \models \neg(\exists y\ b \wedge \neg \bigvee_{i=1}^{n} (\exists z_i\ c_i)) \leftrightarrow \neg(\exists y\ b) \vee \bigvee_{i=1}^{n} (\exists y z_i\ b \wedge c_i)$$

On peut donc remplacer dans (4.7) l'occurrence de la formule gauche de cette équivalence par la formule droite. En distribuant les $\wedge$ sur les $\vee$, on obtient :

$$\neg((\exists x\ a \wedge q \wedge \neg(\exists y\ b)) \vee \bigvee_{i=1}^{n} (\exists x y z_i\ a \wedge b \wedge c_i \wedge q)).$$

En descendant la négation à l'intérieur du $\vee$, cette formule devient :

$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ b)) \wedge \bigwedge_{i=1}^{n} \neg(\exists x y z_i\ a \wedge b \wedge c_i \wedge q).$$

Du fait que les équations de $a \wedge b$ figurent aussi dans c_i, on peut enlever $a \wedge b$ des sous-formules. En renommant des noms de variables quantifiées pour respecter les conditions mentionnées au début de la section 4.1 et en mettant les indices appropriés, on obtient effectivement la formule droite de la règle 11. La règle est donc correcte. Ceci termine la preuve de la correction des règles. □

Ajoutons un mot sur la terminaison des règles de réécriture. Bien entendu il existe d'autres façons pour montrer que l'application de ces règles termine, par exemple on peut utiliser une combinaison de l'ordre lexicographique, l'ordre sur les multi-ensembles [25] et l'ordre de chemin récursif [22]. On trouve de bonnes introductions et synthèses sur des ordres et sur la terminaison des systèmes de réécriture dans [23, 24]. Cependant cette combinaison d'ordres ne démontre la terminaison que d'une façon qualitative. La façon que nous avons utilisée est plutôt quantitative. Nous constatons que les entiers n_i que nous avons utilisés, surtout la fonction α, nous permettent d'avoir une valuation sommaire de la complexité dans le pire des cas de la résolution. Nous considérerons cet aspect plus en détail dans la section 4.6 de ce chapitre.

Nous voyons bien que dans la règle 10, l'élimination de quantificateurs comprend deux tâches principales :

– l'élimination des variables quantifiées non accessibles apparaissant comme membre gauche d'équation et ces équations, la partie $\exists x''\ a''$, et

– l'élimination des variables quantifiées non accessibles qui n'ont pas d'occurrence dans a, la partie x'''.

Cette élimination de quantificateurs fonctionne sur des formules de profondeur 1 et 2. Du fait que les formules générales font intervenir tous les connecteurs logiques, la considération des formules de profondeur 2 est nécessaire.

4.3 Algorithme de résolution de contraintes d'arbres

4.3.1 Algorithme

La résolution d'une formule du premier ordre p consiste à :

1. Transformer la formule p en une formule de travail initiale, équivalente à p dans la théorie vide.
2. Appliquer les règles de réécriture autant de fois que possible. D'après la propriété 4.17, ce processus se termine et produit une formule finale, équivalente à p dans **T**, de la forme :

$$\neg^0(\bigwedge_{i=1}^{n} \neg^4(\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg^4(\exists y_{ij}\ a_i \wedge b_{ij}))), \qquad \text{avec } n \geq 0,\ n_i \geq 0.$$

3. Extraire la formule résolue, équivalente à la formule de travail finale obtenue, dans la théorie vide, par la propriété 4.15 :

$$\bigvee_{i=1}^{n} (\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\ b_{ij})) \tag{4.8}$$

Les règles de réécriture sont correctes quelle que soit la stratégie d'application. On peut donc en fait dériver toute une famille d'algorithmes en fixant la stratégie d'application.

Théorème 4.18 *L'algorithme de résolution transforme toute formule p du premier ordre en une formule résolue q équivalente dans **T**. Cette formule q est la constante logique vrai si $\mathbf{T} \models p$, et la constante logique faux si $\mathbf{T} \models \neg p$. De plus, dans tout modèle $\mathcal{M}$ de **T** :*

1. *Toute var(p)-base finie de p est visible dans q.*
2. *Si la formule de travail initiale est de la forme $\neg^0\neg^3\neg^0\neg^0 p'$, alors toute var$(p)$-base finie de $\neg p$ est visible dans $\neg q$.*

Preuve Par suite des propriétés 4.17 et 4.15, la formule q est résolue et équivalente à la formule p. De plus :

– La formule finale ne contient pas d'autres variables libres que celles de p.

– Si la formule p est fausse dans **T** ($\mathbf{T} \models \neg p$), la formule q est la constante logique *faux*, par le corollaire 4.13.

– Si p est vraie dans **T** ($\mathbf{T} \models p$), alors la formule q est la constante logique *vrai*. En effet, du fait que $\mathbf{T} \models p$, et que p est mis en une formule de travail initiale de la forme $\neg^0\neg^3 p'$, on a $\mathbf{T} \models p'$. Du fait que l'attribution de $\neg^4$ se fait en remontant le long des imbrications de négations d'une formule, p' est mis en une conjonction p'' de formules de profondeur inférieure ou égale à 2, où les négations sont toutes d'indice 4, avant que la sous-formule $\neg^3(\mathit{vrai} \wedge p')$ ne le soit. Comme $\mathbf{T} \models p'$, par le corollaire 4.16, la formule p'' est la constante *vrai*. Par conséquent, on obtient $\neg^0(\neg^4(\mathit{vrai}))$, donc la formule q est *vrai*.

- Si p admet une $var(p)$-base finie, alors q l'admet aussi. Cette base est donc visible dans q, d'après la propriété 4.12.
- Si la formule de travail initiale est de la forme $\neg\neg\neg\neg p'$, il faut d'abord transformer $\neg\neg\neg p'$, qui est équivalente à $\neg p$. Par conséquent, si $\neg p$ admet une $var(p)$-base finie, alors elle est visible dans $\neg q$.

□

Si la formule finale n'est ni *vrai* ni *faux*, elle contient forcément une variable libres, du fait qu'elle doit contenir une variable accessible. Par conséquent, si la formule initiale est une proposition (formule sans variables libres), la formule finale est soit la constante *vrai* soit la constante *faux*. On a donc une autre preuve de la complétude de la théorie des arbres finis ou infinis **T**.

D'autre part, d'après la propriété 2.12, la formule résolue (4.8) obtenue est équivalente à

$$\bigvee_{i=1}^{n} [(\exists x_i \; a_i) \wedge \bigwedge_{j=1}^{n_i} \neg(\exists x_i y_{ij} \; a_i \wedge b_{ij})]$$

On conclut donc qu'avec nos règles de réécriture, on peut transformer toute formule en une disjonction de conjonctions d'équations quantifiées existentiellement.

On retrouve donc les deux résultats de Michael Maher [36] mentionnés dans la section 2.3 du chapitre 2.

Remarquons que dans la démonstration de la correction des algorithmes, on n'a pas utilisé explicitement l'indépendance de diséquations [8, 32, 12, 34]. En échange, tout le long de la démonstration de la correction des algorithmes, nous avons utilisé une hypothèse importante, qui entraîne l'indépendance de diséquations, c'est l'infinité de l'ensemble de symboles fonctionnels **F**. Elle assure la correction de la propriété 4.6, qui entraîne la correction de l'élimination de quantificateurs dans la règle 10, et assure la correction de la propriété 4.12, qui prouve les propriétés principales de la forme résolue. Ces propriétés sont les points cruciaux de la correction des algorithmes.

4.3.2 Exemple de résolution d'une contrainte

Reprenons le premier jeu introduit dans la section 3.1 du chapitre 3. Dans ce jeu, on se donnait un entier positif ou nul i et à tour de rôle, on soustrayait 1 ou 2 de i sans jamais le rendre strictement négatif. La première personne qui ne pouvait plus jouer avait perdu. La formulation de ce jeu dans les arbres avec la relation *coup* généralisée est représentée dans la sous-section 3.1.3. Dans cette définition de *coup*, la seule position où les adversaires ne peuvent plus jouer est 0. On se donne maintenant à résoudre la contrainte

$$\neg\exists v \; coup(u, v) \tag{4.9}$$

qui est donc équivalente à $u = 0$. En remplaçant $coup(u, v)$ par sa définition, la formule (4.9) devient :

$$\neg\exists v \; (u{=}sv \vee (\exists w_1 \; u{=}sw_1 \wedge w_1{=}sv) \vee (u{=}v \wedge \neg(u{=}0) \wedge \neg(\exists w_2 \; u{=}sw_2)))$$

c'est-à-dire sous forme normalisée :

$$\neg\exists v \; \neg(\neg(u{=}sv) \wedge \neg(\exists w_1 \; u{=}sw_1 \wedge w_1{=}sv) \wedge \neg(u{=}v \wedge \neg(u{=}0) \wedge \neg(\exists w_2 \; u{=}sw_2)))$$

on obtient ainsi la formule de travail initiale

$$\neg^0\neg^3\neg^0\exists v\ \neg^0\begin{bmatrix}\neg^0(u=sv)\wedge\\ \neg^0(\exists w_1\ u=sw_1\wedge w_1=sv)\wedge\\ \neg^0(u=v\wedge\neg^0(u=0)\wedge\neg^0(\exists w_2\ u=sw_2))\end{bmatrix}$$

On notera l'utilisation d'indices supplémentaires pour nommer les copies de variables quantifiées. En descendant le long de l'imbrication des négations et en simplifiant les trois sous-formules entre crochets, l'application des règles donne

$$\neg^0\neg^3\neg^3\exists v\ \neg^3\begin{bmatrix}\neg^4(u=sv)\wedge\\ \neg^4(\exists w_1\ u=sw_1\wedge w_1=sv)\wedge\\ \neg^4(v=u\wedge\neg^4(v=u\wedge u=0)\wedge\neg^4(\exists w_2\ v=u\wedge u=sw_2))\end{bmatrix}$$

L'application de la règle 11 sur la dernière sous-formule de la forme $\neg^3(\ldots)$ donne

$$\neg^0\neg^3\neg^3\exists v\begin{bmatrix}\neg^3(\neg^4(u=sv)\wedge\neg^4(\exists w_1\ u=sw_1\wedge w_1=sv)\wedge\neg^4(v=u))\wedge\\ \neg^3(v=u\wedge u=0\wedge\neg^0(u=sv)\wedge\neg^0(\exists w_{11}\ u=sw_{11}\wedge w_{11}=sv))\wedge\\ \neg^3(\exists w_2\ v=u\wedge u=sw_2\wedge\neg^0(u=sv)\wedge\neg^0(\exists w_{12}\ u=sw_{12}\wedge w_{12}=sv))\end{bmatrix}$$

Les règles 1-10 en simplifiant les sous-formules entre crochets, donne

$$\neg^0\neg^3\neg^3\exists v\begin{bmatrix}\neg^4(\neg^4(u=sv)\wedge\neg^4(\exists w_1\ u=sw_1\wedge w_1=sv)\wedge\neg^4(v=u))\wedge\\ \neg^4(v=u\wedge u=0)\wedge\\ \neg^4\begin{bmatrix}\exists w_2\ v=u\wedge u=sw_2\wedge\\ \neg^4(v=u\wedge u=sw_2\wedge w_2=v)\wedge\neg^4(v=u\wedge u=sw_2\wedge w_2=sv)\end{bmatrix}\end{bmatrix}$$

L'application de la règle 11 sur la sous-formule $\neg^3(\exists v\ldots)$, en particularisant la première sous-formule, donne

$$\neg^0\neg^3\begin{bmatrix}\neg^3\begin{bmatrix}\exists v\ \neg(\mathit{vrai})\wedge\neg^0(v=u\wedge u=0)\wedge\\ \neg^0\begin{bmatrix}\exists w_2\ v=u\wedge u=sw_2\wedge\\ \neg^0(v=u\wedge u=sw_2\wedge w_2=v)\wedge\neg^0(v=u\wedge u=sw_2\wedge w_2=sv)\end{bmatrix}\end{bmatrix}\wedge\\ \neg^3\begin{bmatrix}\exists v_1\ u=sv_1\wedge\neg^0(v_1=u\wedge u=0)\wedge\\ \neg^0\begin{bmatrix}\exists w_{21}\ v_1=u\wedge u=sw_{21}\wedge\\ \neg^0(v_1=u\wedge u=sw_{21}\wedge w_{21}=v_1)\wedge\neg^0(v_1=u\wedge u=sw_{21}\wedge w_{21}=sv_1)\end{bmatrix}\end{bmatrix}\wedge\\ \neg^3\begin{bmatrix}\exists v_2 w_1\ u=sw_1\wedge w_1=sv_2\wedge\neg^0(v_2=u\wedge u=0)\wedge\\ \neg^0\begin{bmatrix}\exists w_{22}\ v_2=u\wedge u=sw_{22}\wedge\\ \neg^0(v_2=u\wedge u=sw_{22}\wedge w_{22}=v_2)\wedge\neg^0(v_2=u\wedge u=sw_{22}\wedge w_{22}=sv_2)\end{bmatrix}\end{bmatrix}\wedge\\ \neg^3\begin{bmatrix}\exists v_3\ v_3=u\wedge\neg^0(v_3=u\wedge u=0)\wedge\\ \neg^0\begin{bmatrix}\exists w_{23}\ v_3=u\wedge u=sw_{23}\wedge\\ \neg^0(v_3=u\wedge u=sw_{23}\wedge w_{23}=v_3)\wedge\neg^0(v_3=u\wedge u=sw_{23}\wedge w_{23}=sv_3)\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Les règles 1-10, en simplifiant les sous-formules dans la grande colonne, donne

$$\neg^0\neg^3\begin{bmatrix}\neg^4(\exists v_1\ u=sv_1)\wedge\\ \neg^4(\exists v_2 w_1\ u=sw_1\wedge w_1=sv_2)\wedge\\ \neg^3(\neg^4(u=0)\wedge\neg^4\begin{bmatrix}\exists w_{23}\ u=sw_{23}\wedge\neg^4(u=sw_{23}\wedge w_{23}=u)\wedge\\ \neg^4(\exists v_3\ u=sw_{23}\wedge w_{23}=sv_3\wedge v_3=u)\end{bmatrix})\end{bmatrix}$$

Encore une fois, l'application de la règle 11 sur la troisième sous-formule dans la grande colonne

$$\neg^0\neg^3\begin{bmatrix}\neg^4(\exists v_1\ u=sv_1)\wedge\\ \neg^4(\exists v_2 w_1\ u=sw_1\wedge w_1=sv_2)\wedge\\ \neg^3(\neg^4(u=0)\wedge\neg^4(\exists w_{23}\ u=sw_{23}))\wedge\\ \neg^3(\exists w_{231}\ u=sw_{231}\wedge w_{231}=u\wedge\neg^0(u=0))\wedge\\ \neg^3(\exists w_{232}v_3\ u=sw_{232}\wedge w_{232}=sv_3\wedge v_3=u\wedge\neg^0(u=0))\end{bmatrix}$$

L'application des règles 1-10 sur les sous-formules entre crochets donne

$$\neg^0\neg^3\begin{bmatrix}\neg^4(\exists v_1\ u=sv_1)\wedge\\ \neg^4(\exists v_2 w_1\ u=sw_1\wedge w_1=sv_2)\wedge\\ \neg^4(\neg^4(u=0)\wedge\neg^4(\exists w_{23}\ u=sw_{23}))\wedge\\ \neg^4(\exists w_{231}\ u=sw_{231}\wedge w_{231}=u)\wedge\\ \neg^4(\exists w_{232}v_3\ u=sw_{232}\wedge w_{232}=sv_3\wedge v_3=u)\end{bmatrix}$$

La règle 11, en s'appliquant sur la sous-formule $\neg^3(\ldots)$, la transforme en une conjonction de formules. La formule ci-dessus devient

$$\neg^0\begin{bmatrix}\neg^3\begin{bmatrix}u=0\wedge\\ \neg^0(\exists v_1\ u=sv_1)\wedge\\ \neg^0(\exists v_2 w_1\ u=sw_1\wedge w_1=sv_2)\wedge\\ \neg^0(\exists w_{231}\ u=sw_{231}\wedge w_{231}=u)\wedge\\ \neg^0(\exists w_{232}v_3\ u=sw_{232}\wedge w_{232}=sv_3\wedge v_3=u)\end{bmatrix}\wedge\\ \neg^3\begin{bmatrix}\exists w_{23}\ u=sw_{23}\wedge\\ \neg^0(\exists v_2\ u=sv_1)\wedge\\ \neg^0(\exists v_2 w_1\ u=sw_1\wedge w_1=sv_2)\wedge\\ \neg^0(\exists w_{231}\ u=sw_{231}\wedge w_{231}=u)\wedge\\ \neg^0(\exists w_{232}v_3\ u=sw_{232}\wedge w_{232}=sv_3\wedge v_3=u)\end{bmatrix}\end{bmatrix}$$

Les règles 1-10 simplifient les sous-formules dans la grande colonne. La première sous-formule étant simplifiée à $\neg^4(u=0)$, la deuxième étant simplifiée à la constante *vrai*, la formule devient

$$\neg^0\neg^4(u=0)$$

La formule résolue obtenue est $u=0$. Nous obtenons donc bien une $\{u\}$–solution explicite.

4.4 Décision de la validité d'une formule

Nos algorithmes de résolution transforment toute formule p, soit en la constante logique *vrai*, soit en la constante *faux*, soit en une formule résolue qui contient au moins une occurrence libre de variable. Dans ce dernier cas, la formule p n'est ni toujours vraie ni toujours fausse dans **T**. Si nous nous intéressons seulement à décider si la formule p est toujours vraie ou à décider si p est toujours fausse dans **T**, il se révèle que l'on pourrait obtenir la réponse sans attendre la fin de l'application des règles. Nous expliquons ceci dans ce qui suit.

Propriété 4.19 *Soit p une formule de travail de la forme*

$$\neg^3(\exists x\ a\wedge\textstyle\bigwedge_{i=1}^{n}\neg^3(\exists y_i\ b_i\wedge q_i)) \tag{4.10}$$

Si chaque conjonction b_i contient une équation $u_i=t_i$ ne figurant pas dans a avec u_i ne figurant pas dans y_i, alors $\boldsymbol{T}\not\models p$.

Preuve Il suffit de montrer qu'il existe une interprétation I satisfaisant **T** telle que

$$I(a \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i\ b_i)) = \mathbf{v},$$

puisque cette équation assure que $I(p) = \mathbf{f}$. Sachant que chaque b_i contient une équation particulière $u_i = t_i$, désignons par V l'ensemble de ces variables u_i et des variables v_i si t_i est la variable v_i. Du fait que l'ensemble de symboles fonctionnels **F** est infini, on peut définir une conjonction

$$a \wedge \bigwedge_{u \in V - mbg(a)} u = t,$$

où chaque terme t contient exactement un symbole $f \in \mathbf{F}$, et ces symboles sont tous distincts et ne figurent ni dans a ni dans aucun b_i. Dans ces termes, des nouvelles variables sont introduites si besoin est. La conjonction a étant résolue, et aucune variable de $V - mbg(a)$ n'étant le membre gauche d'une équation de a, cette conjonction est donc résolue. Il existe donc une interprétation I satisfaisant **T** telle que $I(a) = \mathbf{v}$ et $I(u = t) = \mathbf{v}$, pour tout $u \in V$. Du fait que les b_i sont résolues, $I(b_i) = \mathbf{f}$, pour tout $i \in 1..n$. Du fait que u_i ne figure pas dans y_i, $I(\exists y_i\ b_i) = \mathbf{f}$. □

Comme nous l'avons discuté dans la sous-section 4.3.1, il existe forcément une étape à laquelle la formule de travail p, sur laquelle on applique les règles, prend la forme

$$\neg^0\neg^3(p_1 \wedge \cdots \wedge p_n)$$

où les p_i sont de la forme (4.10). S'il y a une formule p_i qui satisfait aux conditions de la propriété 4.19, alors $\mathbf{T} \not\models p$. Sinon chaque fois que la règle 11 transforme une formule p_i en une conjonction de formules, on vérifie si une des formules ajoutées n'est pas vraie dans **T**. Si c'est le cas, alors $\mathbf{T} \not\models p$.

Ensuite, lorsque la formule p devient

$$\neg^0(p'_1 \wedge \cdots \wedge p'_m)$$

où les p_i sont de la forme (4.10), on vérifie de la même façon si $\mathbf{T} \not\models \neg p$.

Remarquons que les conditions de cette propriété 4.19 ne sont que des conditions suffisantes. Bien entendu si aucune sous-formule p_i satisfait ces conditions, l'algorithme continue et donne en s'arrêtant une formule de l'un des trois cas envisagés.

4.5 Élimination de redondances

Nous présentons dans cette section l'utilisation de notre résolution de contraintes pour éliminer certaines sous-formules redondantes. Dans une formule normalisée de la forme

$$\neg(\exists x\ a \wedge p_1 \wedge \cdots \wedge p_n) \tag{4.11}$$

on dit que la sous-formule p_i est redondante ssi

$$\mathbf{T} \models \neg(\exists x\ a \wedge p_1 \wedge \cdots \wedge p_n) \leftrightarrow \neg(\exists x\ a \wedge p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n) \tag{4.12}$$

Propriété 4.20 *La sous-formule $p_i = \neg(\exists x_i\ r_i)$ est redondante si*

$$\boldsymbol{T} \models \neg(\exists x y_i\ a \wedge r_i \wedge p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n). \tag{4.13}$$

Si toutes les variables de x sont accessibles dans $\exists x\, a$, alors p_i est redondante si et seulement si (4.13).

Preuve Avec $q = p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n$, (4.13) devient

$$\mathbf{T} \models \neg(\exists x y_i \; a \wedge r_i \wedge q) \leftrightarrow \mathit{vrai} \tag{4.14}$$

(4.14) est vraie ssi

$$\mathbf{T} \models (a \wedge r_i \wedge q) \leftrightarrow \mathit{faux},$$

c'est-à-dire

$$\mathbf{T} \models (a \wedge q \wedge \neg r_i) \leftrightarrow (a \wedge q).$$

Par conséquent, nous avons

$$\mathbf{T} \models \neg(\exists x \; a \wedge \neg(\exists y_i \; r_i) \wedge q) \leftrightarrow \neg(\exists x \forall y \; a \wedge q).$$

Du fait que les variables dans y ne figurent pas dans a et dans q, nous obtenons

$$\mathbf{T} \models \neg(\exists x \; a \wedge \neg(\exists y_i \; r_i) \wedge q) \leftrightarrow \neg(\exists x \; a \wedge q)$$

ce qui est effectivement l'équivalence (4.12).

Si toutes les variables de x sont accessibles dans $\exists x \; a$, alors d'après la propriété 4.5, on a $\mathbf{T} \models \exists ? x \; a$. D'après la propriété 2.12

$$\mathbf{T} \models \neg(\exists x \; a \wedge \neg(\exists y_i \; r_i) \wedge q) \leftrightarrow \neg(\exists x \; a \wedge q) \vee (\exists x_i y_i \; a[x_i] \wedge r_i[x_i])$$

Pour que la formule (4.12) soit vraie, il faut et il suffit que

$$\mathbf{T} \models (\exists x_i y_i \; a[x_i] \wedge r_i[x_i]) \rightarrow \neg(\exists x \; a \wedge q)$$

c'est-à-dire que

$$\mathbf{T} \models \neg(\exists x x_i y_i \; a \wedge a[x_i] \wedge r_i[x_i] \wedge q)$$

Du fait que $\mathbf{T} \models \exists ? x \; a$, il faut et il suffit de montrer que

$$\mathbf{T} \models \neg(\exists x x_i y_i \; a \wedge x = x_i \wedge r_i[x_i] \wedge q).$$

Ici $x = x_i$ représente la conjonction des équations $u = u_i$ avec u figurant dans x et u_i correspondant à u et figurant dans x_i. En remplaçant dans $r_i[x_i]$ les occurrences des variables de x_i par des variables correspondantes dans x et en éliminant la quantification $\exists x_i$, il faut et il suffit donc de montrer que :

$$\mathbf{T} \models \neg(\exists x y_i \; a \wedge r_i[x] \wedge q)$$

ce qui est effectivement la formule (4.14). □

Nous pouvons donc appliquer la résolution dans **T** sur la formule dans (4.13). Si la formule équivalente obtenue est *vrai*, alors la sous-formule p_i est redondante. Lorsque la formule (4.11) est une formule de travail et sa négation est d'indice 4, le test (4.13) nous permet donc d'éliminer toutes les sous-formules redondantes de type (4.12).

Exemple 4.9 Dans la formule

$$\neg(\exists v \; u = f v \wedge \neg(\exists w_1 \; v = s w_1) \wedge \neg(\exists w_2 w_3 \; v = s w_2 \wedge w_2 = s w_3)), \tag{4.15}$$

la sous-formule $\neg(\exists w_2 w_3\ v = sw_2 \wedge w_2 = sw_3)$ est redondante. En effet, cette sous-formule est vraie si v n'est pas de forme ssw pour toute w, comme l'indique la sous-formule $\neg(\exists w_1\ v = sw_1)$. La formule (4.15) est donc équivalente à

$$\neg(\exists v\ u = fv \wedge \neg(\exists w_1\ v = sw_1))$$

En appliquant le test pour la redondance de la deuxième sous-formule de (4.15), il nous amène à résoudre la formule

$$\neg(\exists v w_2 w_3\ u = fv \wedge v = sw_2 \wedge w_2 = sw_3 \wedge \neg(\exists w_1\ v = sw_1))$$

La première conjonction d'équations étant résolue, simplifions la conjonction d'équations dans la sous-formule, nous obtenons

$$\neg(\exists v w_2 w_3\ u = fv \wedge v = sw_2 \wedge w_2 = sw_3 \wedge \neg(\exists w_1\ u = fv \wedge v = sw_2 \wedge w_2 = sw_3 \wedge w_1 = w_2))$$

C'est-à-dire, en poursuivant la simplification,

$$\neg(\exists v w_2 w_3\ u = fv \wedge v = sw_2 \wedge w_2 = sw_3 \wedge \neg(u = fv \wedge v = sw_2 \wedge w_2 = sw_3))$$

Cette formule est effectivement équivalente à *vrai* dans **T**. La deuxième sous-formule de (4.15) est donc redondante.

4.6 Borne supérieure de la taille de la formule finale

Convenons que la taille $|p|$ d'une formule p est le nombre d'occurrences de variable et de symboles de fonction utilisés pour écrire la formule. Du fait que l'application des règles de réécriture termine, il existe une borne supérieure de la taille des formules finales. Nous nous intéressons maintenant à estimer cette borne.

On introduit les fonctions $\exp_k(n)$ et $\beta_k(n)$ suivantes pour $k \geq 0$:

$$\begin{array}{ll} \exp_0(n) = n, & \exp_{k+1}(n) = 2^{\exp_k(n)}, \\ \beta_0(n) = n, & \beta_{k+1}(n) = n2^{\beta_k(n)}. \end{array}$$

Ces fonctions $\exp_k(n)$ et $\beta_k(n)$ sont donc des tours d'exponentielles dont la hauteur est $k+1$. Nous montrons les lemmes suivants.

Lemme 4.21 *Avec $k \geq 1$ et $n \geq 1$, on a $(2k+1)\beta_{k-1}(n) < \exp_k(2n)$.*

Preuve Montrons d'abord que $k^2 \leq 2^k$. Ceci est évident pour $k = 1$ et $k = 2$. Supposons que ce soit vrai pour $k \geq 2$ et montrons que c'est vrai pour $k+1$. De $k^2 \leq 2^k$ on déduit $2k^2 \leq 2^{k+1}$ et donc du fait que $(k+1)^2 \leq 2k^2$, avec $k \geq 2$, on a $(k+1)^2 \leq 2^{k+1}$.

Constatons ensuite que si deux entiers a et b sont tels que $a \geq 2$ et $b \geq 2$ alors $ab \geq a + b$. En effet, puisque $a \geq 2$, $a/(a-1) \leq 2$, donc $a/(a-1) \leq b$, d'où $a + b \leq ab$.

Montrons maintenant que $(2k+1)n(\beta_{k-1}(n)+1) \leq \exp_k(n)$ par induction sur k. Si $k = 1$, on a $(2k+1)n(\beta_{k-1}(n)+1) = 3n^2 + n$ et $3n^2 + n \leq (2n)^2 \leq 2^{2n} = \exp_1(2n)$. Supposons que ce soit vrai pour $k \geq 1$, montrons que c'est aussi vrai pour $k+1$. On a

$$(2(k+1)+1)n(\beta_k(n)+1) \leq (2k+3)n^2(2^{\beta_{k-1}(n)}+1)$$

Du fait que $(2k+3) < (2k+1)^2$, que $(2k+1)^2 n^2 \leq 2^{(2k+1)n}$ et du fait que $2^{\beta_{k-1}(n)}+1 < 2^{\beta_{k-1}(n)+1}$ avec $k \geq 1$, on a

$$(2k+3)n^2(2^{\beta_{k-1}(n)}+1) < 2^{(2k+1)n+\beta_{k-1}(n)+1}$$

Du fait que $(2k+1)n > 2$ et $(\beta_{k-1}(n)+1) \geq 2$ avec $k \geq 1$, on a

$$2^{(2k+1)n+\beta_{k-1}(n)+1} < 2^{(2k+1)n(\beta_{k-1}(n)+1)}$$

La deuxième valeur étant plus petite que $2^{\exp_k(2n)}$ par l'hypothèse d'induction, on conclut donc que l'inéquation à montrer est aussi vraie pour $k+1$. □

Lemme 4.22 *Soit p une formule de profondeur k dont toutes les sous-formules de la forme $\neg(\exists x\ a \wedge q)$ sont telles que $|a| \leq t$. Après les applications des règles de 1 à 10, toutes ces sous-formules sont telles que $|a| < t3^k$.*

Preuve Pour estimer une borne supérieure, nous ne tenons compte que les règles qui agrandissent la formule. Les règles de 1 à 6 doublent au plus la taille d'une conjonction d'équations. De plus, du fait que la règle 7 propage les équations d'une formule dans ces sous-formules et que la règle 8 restaure les équations propagées, la taille maximale de la conjonction d'équations d'une sous-formule à la profondeur i est bornée par t_i, avec t_i le i-ème élément de la suite suivante :

$$\begin{array}{lcl} t_1 & = & 2t, \\ t_{i+1} & = & 2(t+t_i)+t_i \end{array}$$

Par conséquent, $t_k = t(3^k-1)$ et la taille de toute conjonction d'équations est plus petite que $t3^k$. □

Appelons la largeur d'une formule p de la forme $\neg(\exists x\ a \wedge q)$ le nombre d'éléments composant la conjonction q. On la désigne par $\|q\|$.

Propriété 4.23 *Soit t et n des entiers positifs. Soit p une formule de profondeur k dont toutes les sous-formules de la forme $\neg(\exists x\ a \wedge q)$ sont telles que $|a| \leq t$ et $\|q\| \leq n$. L'algorithme transforme p en une formule finale p' telle que $|p'| < t\exp_{k+1}(2n)$.*

Preuve La seule règle qui augmente la largeur d'une formule est la règle 11. Cependant d'après la démonstration de la terminaison des règles, cette règle 11 diminue $\alpha(p)$. Rappelons que la fonction α est définie comme suit :

$$\begin{array}{l} \alpha(\mathit{vrai}) = 0 \\ \alpha(\neg(\exists x\ a \wedge p_1 \wedge \cdots \wedge p_n)) = 2^{\alpha(p_1)+\cdots+\alpha(p_n)} \end{array}$$

Il est simple de vérifier que le nombre de sous-formules à chaque instant ne peut dépasser $\alpha(p)$. Chaque fois que la règle 11 s'applique, elle crée des sous-formules sur lesquelles les règles 1-10 deviennent applicables. Ces formules sont de profondeurs au plus égales à k et le nombre fois que les règles 1-10 deviennent applicables est plus petit que le nombre de fois que la règle 11 peut s'appliquer. Par le lemme 4.22, la taille de la conjonction d'équations d'une sous-formule est plus petite que

$$t3^{k\alpha(p)}.$$

La taille $|p'|$ de la formule finale est donc bornée par $t3^{k\alpha(p)}\alpha(p)$ qui est plus petit que

$$t2^{(2k+1)\alpha(p)}.$$

Du fait que p est de profondeur k et n est la largeur maximale des sous-formules de p, on a $\alpha(p) \leq \beta_{k-1}(n)$. D'après le lemme 4.21, on a

$$t2^{(2k+1)\beta_{k-1}(n)} < t2^{\exp_k(2n)}$$

On a donc $|p| < t\exp_{k+1}(2n)$, ce qui termine la preuve. □

La taille des formules finales est donc bornée par une fonction qui est proche d'une tour de puissances de 2 (avec association de haut en bas) dont la hauteur est la profondeur maximale d'imbrication des négations de la formule normalisée initiale.

Les fonctions $n \mapsto$ cst, $+$, $\times$, $n \mapsto 2^n$, de type $\mathcal{N} \to \mathcal{N}$, sont élémentaires [26]. La tour $t\exp_{k+1}(2n)$ dépasse donc toute composition finie de fonctions élémentaires. Comme nous l'avons mentionné dans la section 3.2 du chapitre 3, le problème de décision de la validité d'une proposition dans la théorie **T** est un problème non-élémentaire, c'est-à-dire que la complexité en temps ou en espace de tout algorithme résolvant le problème ne peut être bornée supérieurement par une fonction obtenue par composition finie de fonctions élémentaires. Il est donc normal que la complexité de notre algorithme de résolution ainsi que la taille des formules finales soient de cet ordre.

Chapitre 5

Extensions possibles

Sommaire

Nous savons maintenant comment résoudre des contraintes du premier ordre dans la théorie des arbres finis ou infinis, avec une infinité de symboles fonctionnels et avec une seule relation, l'égalité. Nous savons donc résoudre des contraintes l'algèbre des arbres éventuellement infinis, où les variables sont interprétées par des arbres finis ou infinis. Dans ce chapitre nous proposons des algorithmes de résolution de contraintes dans les cas où certaines variables ou toutes les variables ne peuvent être que des arbres finis.

5.1 Extension avec la contrainte *fini*

Comme dans le chapitre précédent, nous présentons la théorie, les formes de formules et la résolution par des règles de réécriture.

5.1.1 La théorie T∪FINI et ses modèles

La théorie **T**∪FINI a, pour ensemble **F** de symboles de fonction, un ensemble comprenant au moins un symbole d'arité nulle et une infinité de symboles d'arités non nulles et, pour ensemble **R** de symboles de relation, l'ensemble formé du seul symbole *fini* d'arité 1. Elle consiste en l'ensemble infini de propositions de l'une des cinq formes dont trois composent la théorie **T** :

$$\forall x \forall y \, \neg(fx = gy) \tag{5.1}$$

$$\forall x \forall y \, fx = fy \rightarrow \bigwedge_i x_i = y_i \tag{5.2}$$

$$\forall x \exists! y \ \bigwedge_i y_i = t_i[yx] \tag{5.3}$$

et les deux autres formes de FINI sont :

$$\forall x \forall u \ \neg(u = t[ux] \wedge \mathit{fini}(u)) \tag{5.4}$$

$$\forall x \forall u \ u = fx \wedge \mathit{fini}(u) \leftrightarrow u = fx \wedge \bigwedge_i \mathit{fini}(x_i) \tag{5.5}$$

où f, g sont des éléments distincts pris dans **F**, où x est un vecteur composé de variables x_i, y est un vecteur composé de variables y_i, toutes distinctes, où $t_i[yx]$ est un terme formé d'un élément de **F** suivi de variables prises dans x ou y et où $t[ux]$ est un terme faisant intervenir au moins un élément de **F**, la variable u et des variables prises dans x.

Dans la forme (5.5), si f est une constante, c'est-à-dire d'arité 0, et si x est le vecteur vide de variables, nous avons :

$$\forall u \ u = f \wedge \mathit{fini}(u) \leftrightarrow u = f$$

ceci signifie que la contrainte $\mathit{fini}(f)$ est vérifiée pour toute constante f.

À partir de l'algèbre des arbres, de l'algèbre des arbres rationnels et de l'algèbre des ensembles de nœuds définis en **F**, on construit alors les trois structures

$$\mathcal{A} = \langle \mathbf{A}, (f^A)_{f \in F}, \mathit{fini}^A \rangle,$$
$$\mathcal{B} = \langle \mathbf{B}, (f^B)_{f \in F}, \mathit{fini}^B \rangle,$$
$$\mathcal{D} = \langle \mathbf{D}, (f^D)_{f \in F}, \mathit{fini}^D \rangle,$$

où les relations unaires $\mathit{fini}^D(u)$, $\mathit{fini}^A(u)$ et $\mathit{fini}^B(u)$ sont vraies ssi u est un ensemble fini de nœuds.

Cette théorie **T**∪FINI est satisfaisable, du fait qu'elle a la structure $\mathcal{A}$ comme modèle. En effet, si on se place dans l'ensemble des arbres, si on interprète chaque symbole $f \in \mathbf{F}$ comme l'opération de construction f^A et l'on interprète $\mathit{fini} \in \mathbf{R}$ comme fini^A, alors toutes les propriétés de la théorie **T**∪FINI sont vraies. Les propriétés (5.1), (5.2) et (5.3) sont vraies comme a démontré le théorème 2.14 dans le chapitre 2. La forme (5.4) se traduit par le fait qu'un arbre fini ne contient pas lui-même comme un sous-arbre strict. La forme (5.5) correspond au fait qu'un arbre est fini si et seulement si ses sous-arbres immédiats sont finis. De la même façon on montre que les structures $\mathcal{B}$ et $\mathcal{D}$ sont modèles de la théorie **T**∪FINI.

La théorie **T**∪FINI n'est pas redondante. La structure $\mathcal{A}$, dans laquelle on aurait interprété $\mathit{fini}^A(u)$ comme toujours faux, satisfait tous les axiomes sauf (5.5) si f est une constante et celle, dans laquelle on aurait interprété $\mathit{fini}^A(u)$ comme toujours vrai, satisfait tous les axiomes sauf (5.4).

Remarque 5.1 Du fait que **F** contient au moins un élément d'arité nulle et un élément d'arité non nulle, de (5.5) on conclut que dans le domaine d'un modèle de **T**∪FINI, il existe une infinité de u tel que $\mathit{fini}(u)$.

5.1.2 Formes de formules

Comme dans la résolution dans la théorie **T**, on impose les conditions sur une formule quelconque p :

- Les variables quantifiées de p sont aussi distinctes que possible.
- Pour toutes variables u, v et toute sous-formule q de p, si u et v ont des occurrences respectivement libres et liées dans q alors $u \prec v$.

Nous utilisons aussi ces notations :

- les lettres f, g, s désignent des symboles fonctionnels,
- les lettres u, v, w désignent des variables, les lettres x, y, z des vecteurs de variables, les lettres X, Y, Z les ensembles des variables figurant respectivement dans x, y, z,
- la lettre t désigne un terme,
- les lettres a, b, c désignent des conjonctions d'équations, les lettre p, q, r des formules.

Toutes ces lettres peuvent être altérées par des primes ou des indices.

Conjonction étendue d'équations

En considérant des formules faisant intervenir des occurrences de *fini*, une question se pose, comment considérer ces occurrences, comme des équations ou comme des sous-formules négativées ? Constatons que si une formule p contient la restriction $\mathit{fini}(u)$ pour une variable u, alors cette restriction peut être appliquée partout dans toutes les sous-formules de p. Ceci nous amène alors à propager la contrainte $\mathit{fini}(u)$ dans les sous-formules de p ; c'est la façon dont on traite les équations. Nous groupons donc les occurrences de *fini* avec les équations de p.

Définition 5.2 *Une* conjonction étendue d'équations *est de la forme*

$$u_1 = t_1 \wedge \cdots \wedge u_n = t_n \wedge \mathit{fini}(v_1) \wedge \cdots \wedge \mathit{fini}(v_m) \wedge \mathit{vrai}$$

avec $n, m \geq 0$, où les équations $u_i = t_i$ sont élémentaires et où les v_i sont des variables. Une conjonction étendue est résolue *si les variables u_i et v_i sont toutes distinctes et toute équation de la forme $u = v$ est telle que $v \prec u$.*

La contrainte *fini* étant une-aire, la définition de variables et d'équations accessibles n'est pas changée. Une contrainte $\mathit{fini}(u)$ est accessible ssi la variable u l'est.

Exemple 5.1 Dans la formule

$$\exists uvw\ z = fuv \wedge v = fuv \wedge w = fuv \wedge \mathit{fini}(u) \wedge \mathit{fini}(w),$$

les variables accessibles sont z, u, v, les équations accessibles sont $z = fuv$ et $v = fuv$ et la seule contrainte accessible est $\mathit{fini}(u)$. La variable u est accessible depuis v, la variable v est accessible depuis elle-même.

Dans une conjonction étendue résolue, si une variable u figure dans une occurrence de $\mathit{fini}(u)$, alors elle ne figure pas comme membre gauche d'équation. Les propriétés 4.5, 4.2 et 4.3 restent donc toujours valides dans la théorie $\mathbf{T} \cup$ FINI, ce qui correspondent aux propriétés suivantes.

Propriété 5.3 *Soit b une conjonction étendue résolue d'équations et soit y un vecteur de variables. Si toutes les variables de y sont accessibles dans $\exists y\ b$, alors $\mathbf{T} \cup \mathit{FINI} \models \exists ?y\ b$.*

Propriété 5.4 *Si a est une conjonction résolue d'équations élémentaires et x la suite de ses membres gauches, alors $\mathbf{T} \cup \mathit{FINI} \models \exists ! x\ a$.*

Propriété 5.5 *Soient a et b des conjonctions résolues d'équations élémentaires. Si a et b ont les mêmes membres gauches et $\mathbf{T} \cup \mathit{FINI} \models a \rightarrow b$ alors $\mathbf{T} \cup \mathit{FINI} \models a \leftrightarrow b$.*

De plus, montrons la propriété suivante sur les conjonctions étendues d'équations.

Propriété 5.6 *Si a est une conjonction d'équations élémentaires et b et c sont des conjonctions de fini telles que $a \wedge b$ et $a \wedge c$ soient résolues, alors $\mathbf{T} \cup FINI \models a \wedge b \leftrightarrow a \wedge c$ si et seulement si b et c contiennent les mêmes contraintes.*

Preuve Si $b = c$ il est évident que $\mathbf{T} \cup \text{FINI} \models a \wedge b \leftrightarrow a \wedge c$. Montrons maintenant l'autre direction d'implication. Du fait que b et c sont interchangeables, il suffit de montrer que si une contrainte $\mathit{fini}(u)$ figure dans b alors elle figure aussi dans c.

Si $\mathit{fini}(u)$ figure dans b, nous avons $\mathbf{T} \cup \text{FINI} \models a \wedge b \rightarrow \mathit{fini}(u)$, d'où $\mathbf{T} \cup \text{FINI} \models a \wedge c \rightarrow \mathit{fini}(u)$. Du fait que $a \wedge b$ est résolue, u n'est pas un membre gauche d'équation de a. La conjonction $a \wedge c$ ne contient donc pas de sous-formule de la forme $u = t[x] \wedge \bigwedge \mathit{fini}(x_i)$. Puisque $a \wedge c$ est résolue, c ne contient pas de contraintes de la forme $\mathit{fini}(v)$ où v est un membre gauche de a. La conjonction $a \wedge c$ ne contient non plus de sous-formule de la forme $v = t[xu] \wedge \mathit{fini}(v)$. Par conséquent, $\mathit{fini}(u)$ doit figurer dans c. □

La propriété 4.6 dans la théorie **T** peut aussi être généralisée dans cette théorie **T**∪FINI.

Propriété 5.7 *Soient x et x' des vecteurs disjoints de variables. Soient a_i des conjonctions étendues résolues d'équations et x_i des vecteurs de variables tels que toutes les variables de x_i soient accessibles dans $\exists x_i a_i$. Si chaque conjonction a_i contient, soit une occurrence de $\mathit{fini}(u)$, avec $u \in X$, soit une équation contenant u, avec $u \in X \cup X'$, alors*

$$\mathbf{T} \cup FINI \models \exists x x' (\bigwedge_{u \in X'} \mathit{fini}(u)) \wedge (\bigwedge_{i=1}^{n} \neg(\exists x_i \, a_i)) \tag{5.6}$$

Preuve D'après les hypothèses, chaque conjonction a_i contient, soit la contrainte $\mathit{fini}(u)$ pour une variable $u \in X$, soit une équation faisant intervenir u, avec $u \in X \cup X'$. Du fait que a_i est résolue, cette équation est, soit de la forme $u = f v_1 \ldots v_n$, soit de la forme $u = v$, avec $v \notin X_i$ et v différent de u, soit de la forme $v = t$ avec u figurant dans t. Dans ce dernier cas, du fait que toutes les variables de x_i sont accessibles dans $\exists x_i \, a_i$, cette équation fait partie d'une sous-formule de la forme $\bigwedge_{j=1}^{k} v_j = t[y_j]$ avec $v_1 \notin (X \cup X' \cup X_i)$ et $v_{j+1} \in Y_j$. Soit I une interprétation satisfaisant **T**∪FINI. On construit une interprétation $J \in \mathit{proche}(xx', I)$ telle que, pour une variable $u \in X \cup X'$, les conditions suivantes soient satisfaites.

1. $J(\mathit{fini}(u)) = \mathbf{v}$ pour toute variable $u \in X'$.
2. Si une conjonction a_i contient la contrainte $\mathit{fini}(u)$, avec $u \in X$, alors $J(u) = J(f)(J(u), \ldots, J(u))$, avec f d'arité $n > 0$ ne figurant dans aucun a_i.
3. Si une conjonction a_i contient une formule de la forme $u = f v_1 \ldots v_n$, alors $J(u) \neq J(f)(J(v_1), \ldots, J(v_n))$.
4. Si une conjonction a_i contient une équation de la forme $u = v$, alors $J(u) \neq J(v)$.
5. Si une conjonction a_i contient une sous-formule de la forme $\bigwedge_{j=1}^{k} v_j = t[y_j]$, avec $v_1 \notin (X \cup X' \cup X_i)$, $v_{j+1} \in Y_j$ et $u \in Y_k$, alors $J(u)$ est différent de l'élément u déterminé par $I(v_1)$. (Ce point pourrait être plus détaillé.)

Par l'infinité de symboles d'arités non nulles dans **F** et d'après la remarque 5.1, une telle interprétation existe toujours. Par les axiomes (5.1) et (5.4) de **T**∪FINI et par la construction de J, on a $J(\exists x_i \, a_i) = \mathbf{f}$ pour $i \in 1..n$. Par conséquent,

$$I(\exists x x' (\textstyle\bigwedge_{u \in x'} \mathit{fini}(u)) \wedge (\bigwedge_{i=1}^{n} \neg(\exists x_i \, a_i))) = \mathbf{v}$$

Du fait que I satisfaisant **T**∪FINI est choisie quelconque, la formule (5.6) est donc prouvée. □

Formule normalisée

Définition 5.8 *Une* formule normalisée *de profondeur $d \geq 1$, est une formule de la forme*

$$p = \neg(\exists x\ a \wedge \bigwedge_{i=1}^{n} p_i), \qquad \textit{avec } n \geq 0 \tag{5.7}$$

où a est une conjonction étendue d'équations et, les p_i des formules normalisées de profondeur d_i, avec $d = 1 + \max\{0, d_1, \ldots, d_n\}$.

La seule différence par rapport à la forme normalisée dans **T** est que, dans la formule (5.7), la conjonction a peut contenir des occurrences de *fini*. Avec les transformations comme celles présentées dans la sous-section 4.1.2, on peut transformer toute formule en une conjonction de formules normalisées équivalente dans la théorie vide.

Formule résolue

Définition 5.9 *Une* formule résolue *est de la forme*

$$\bigvee_{i=1}^{n} (\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\ b_{ij})), \qquad \textit{avec } n \geq 0,\ n_i \geq 0, \tag{5.8}$$

où a_i et les éventuels b_{ij} sont des conjonctions étendues d'équations, qui respectent les conditions suivantes :

1. *Les conjonctions a_i et b_{ij} sont résolues.*
2. *Avec a_i' la conjonction des équations dans a_i, les conjonctions $a_i' \wedge b_{ij}$ sont résolues.*
3. *Si $\mathit{fini}(u)$ est dans a_i alors, soit b_{ij} contient $\mathit{fini}(u)$, soit b_{ij} contient $\mathit{fini}(v)$ où v est accessible dans $a_i \wedge b_{ij}$ à partir de u et où v n'est pas le membre gauche d'une équation de $a_i \wedge b_{ij}$.*
4. *Toutes les variables et équations de $\exists x_i\ a_i$ sont accessibles et, pour chaque i, toutes les variables et équations de $\exists y_{ij} b_{ij}$ sont accessibles.*
5. *Chaque conjonction b_i ne contient pas seulement des contraintes fini de a.*

Exemple 5.2 La formule suivante est une formule résolue

$$\exists v\ u_1 = f\,v \wedge \mathit{fini}(u_2) \wedge \mathit{fini}(u_3) \wedge \neg(\exists w\ u_2 = f\,w \wedge \mathit{fini}(w) \wedge \mathit{fini}(u_3))$$

La notion qu'un ensemble d'affectations soit visible dans une formule résolue p ou dans $\neg p$ est la même que celle dans la théorie **T**. Comme la forme résolue dans la théorie **T**, cette forme résolue possède une propriété exactement comme la propriété 4.12 dans la théorie **T**, il faut seulement étudier en détail chaque cas. Soit $p = p_1 \vee \cdots \vee p_n \vee \mathit{faux}$ une formule résolue, où chaque p_i est de la forme suivante, qui satisfait les conditions de la définition 5.9 :

$$\exists x\ a \wedge \bigwedge_{j=1}^{m} \neg(\exists y_j\ b_j).$$

Soit p_i une formule de cette forme. Soit W un sous-ensemble de **V** et soit $W^\star$ l'ensemble des variables qui appartiennent à W ou qui sont accessibles dans a à partir d'une variable de W. Soit $\mathcal{M}$ un modèle de **T**∪FINI.

Lemme 5.10 *Dans $\mathcal{M}$, la formule p_i a un ensemble infini de W-solutions si $W^\star \subseteq \mathit{mbg}(a)$, sinon une et une seule W-solution. Par conséquent, si p_i a une seule $\mathit{var}(p_i)$-solution, alors p_i est de la forme $\exists x\, a$, où $\mathit{mbg}(a) = X \cup \mathit{var}(p_i)$.*

Preuve Du fait que les conditions dans la définition 5.9 sont satisfaites, chaque conjonction b_j contient, soit une équation de la forme $u_j = t_j$, avec $u_j \notin Y_j \cup \mathit{mbg}(a)$, soit une contrainte $\mathit{fini}(u_j)$, avec $u_j \notin Y_j \cup \mathit{mbg}(a)$ et $\mathit{fini}(u_j)$ ne figurant pas dans a. Soit V l'ensemble de ces variables u_j et des variables v_j si le terme t_j est la variable v_j. Du fait que b_j est résolue, $v_j \prec u_j$, donc $v_j \notin Y_j$. Par suite des hypothèses, si $n > 0$ alors $V - \mathit{mbg}(a) \neq \emptyset$. Considérons les conjonctions de la forme suivante :

$$a \wedge (\bigwedge_{u \in (V \cup W^\star) - mbg(a)} u = t) \wedge v_0 = f_0, \tag{5.9}$$

avec v_0 une nouvelle variable et f_0 une constante dans **F**. Ici les termes t sont choisis comme suit, sous condition que les symboles de fonction soient tous distincts et différents de ceux figurant dans a et dans p_i :

- Si $\mathit{fini}(u)$ figure dans a, alors t est un terme $f v_0 ... v_0$, avec $f \in \mathbf{F}$ d'arité $n > 0$.
- Si $\mathit{fini}(u)$ figure dans un b_j et ne figure pas dans a, alors t est un terme $f u ... u$, avec $f \in \mathbf{F}$ d'arité $n > 0$.

L'infinité de symboles d'arités non nulles de **F** assure l'existence d'une infinité de conjonctions de cette forme.

D'après la construction de (5.9), les conjonctions de cette forme sont résolues. De plus, pour toute interprétation I satisfaisant **T**∪FINI, pour tout $j \in 1..n$,

$$I((\bigwedge_{u \in V - mbg(a)} u = t) \wedge v_0 = f_0) = \mathbf{v} \text{ entraîne } I(\exists y_j\, b_j) = \mathbf{f}.$$

Toute interprétation qui satisfait (5.9) satisfait donc p_i. De plus, dans un modèle $\mathcal{M}$ de **T**∪FINI, une W-solution de (5.9) est aussi une W-solution de p_i. La formule p_i a donc au moins une W-solution.

Si $W^\star \subseteq \mathit{mbg}(a)$, alors d'après la propriété 5.4, on a **T**∪FINI $\models \exists! z a'$, avec z un vecteur composé des variables dans $W^\star$ et, avec a' la conjonction des équations dans a dont le membre gauche figure dans $W^\star$. La formule p_i a donc une seule W-solution.

Si $W^\star \not\subseteq \mathit{mbg}(a)$, alors la conjonction $\bigwedge_{u \in (V \cup W^\star) - mbg(a)} u = t$ dans (5.9) est différente de *vrai*. Du fait que **F** a une infinité de symboles d'arité non nulles, il existe une infinité de conjonctions de la forme (5.9) telles que les W-solutions de deux conjonctions différentes soient différentes. Du fait que leur W-solution est aussi une W-solution de p_i, la formule p_i a donc une infinité de W-solutions.

Par conséquent, si p_i a une seule W-solution, avec $W = \mathit{var}(p_i)$, alors aucun b_j ne contient des équations dont le membre gauche figure dans $\mathit{var}(p_i)$. Du fait que toutes les variables de X sont accessibles dans $\exists x a$ et appartiennent à $\mathit{mbg}(a)$, forcément $m = 0$ et p_i est donc de la forme $\exists x\, a$, où $\mathit{mbg}(a) = X \cup \mathit{var}(p_i)$. □

Nous déduisons immédiatement les résultats suivants :

Propriété 5.11 *Soit p une formule résolue. Toute $\mathit{var}(p)$-base finie de p est visible dans p.*

Corollaire 5.12 *La seule formule résolue p telle que **T**∪FINI $\models \neg p$ est la formule faux.*

5.1.3 Système de règles de réécriture

Formule de travail

Définition 5.13 *Une* formule de travail *est une formule normalisée dont les occurrences de $\neg$ ont été remplacées par des $\neg^k$, avec k pris dans l'intervalle $0..5$ et telle que toute occurrence de sous-formule de la forme*

$$p = \neg^k(\exists x\, a \wedge q), \qquad \textit{avec } k > 0 \tag{5.10}$$

satisfasse aux k premières conditions de la liste de conditions ci-dessous. Ici a est une conjonction étendue d'équations, q une conjonction de formules de travail $\bigwedge_{i=1}^{n} \neg^{k_i}(\exists y_i\ b_i \wedge q_i)$, avec $n \geq 0$, et, dans les conditions, a' désigne la conjonction étendue d'équation de la sur-formule de travail immédiate de p, si elle existe.

1. *Si a' existe, alors $\mathbf{T} \cup FINI \models a \rightarrow a'$ et l'ensemble des membres gauches et de variables figurant dans des occurrence de fini de a' est inclus dans celui de a.*
2. *La conjonction des équations de a est résolue.*
3. *La conjonction a (consiste d'équations et de contraintes fini) est résolue.*
4. *Si a' existe, l'ensemble des équations de a' est inclus dans celui de a.*
5. *Les variables de x et les équations et les contraintes fini de a sont toutes accessibles dans $\exists x\ a$ et si $n > 0$, la conjonction a est différente de chaque b_i.*

Remarquons que l'ajout des contraintes *fini* fait que les indices varient de 0 à 5, en distinguant le fait qu'une conjonction d'équations est résolue et qu'une conjonction étendue d'équations est résolue. La dernière condition indique que chaque conjonction b_i contient soit une équation, soit une contrainte *fini* qui ne figure pas dans a. Ici on ne pose pas de condition que toute contrainte *fini* figurant dans a figure aussi dans b_i. Ceci en effet n'est pas nécessaire : si deux conjonctions résolues a est b_i sont équivalentes, et qu'elles ont les mêmes équations, alors elles ont les mêmes contraintes *fini*, d'après la propriété 5.6.

Une formule de travail initiale est de la forme $\neg^0\neg^4 p$ où dans p, $k = 0$ pour les occurrences de $\neg^k$. Une formule de travail finale est de la forme $\neg^0 p$, où p est une conjonction de formules de profondeur inférieure ou égale à 2, avec $k = 5$ pour toutes les occurrences de $\neg^k$. En ce qui concerne la relation entre formule de travail finale et formule résolue, on a la propriété suivante :

Propriété 5.14 *Soit p une formule de travail finale de la forme*

$$\neg^0(\bigwedge_{i=1}^{n} \neg^5(\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg^5(\exists y_{ij}\ a_{ij} \wedge b_{ij}))),$$

avec $n \geq 0$, avec $n_i \geq 0$, avec a_i et a_{ij} ayant les mêmes équations et les contraintes fini(u) qui figurent dans a_{ij} figurant aussi dans a_i. La formule

$$\bigvee_{i=1}^{n} (\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\ b_{ij}))$$

est une formule résolue, équivalente à p dans la théorie vide.

De cette propriété et du corollaires 5.12, on déduit que :

Corollaire 5.15 *Si p est de la forme $\bigwedge_{i=1}^{n} \neg^5(\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg^5(\exists y_{ij}\ a_{ij} \wedge b_{ij}))$ et si $\mathbf{T} \cup FINI \models p$ alors p est la formule vrai, c'est-à-dire $n = 0$.*

Règles de réécriture

Nous présentons maintenant 16 règles de réécriture pour transformer une conjonction de formules de travail initiales en une conjonction de formules finales équivalente dans **T**∪FINI. Le seule ajout est celui des 5 règles de simplification concernant des contraintes *fini*. Dans

ce qui suit, les lettres u, v, w désignent des variables, les lettres x, y, z désignent des vecteurs de variables, les lettres a, b, c des conjonctions étendues d'équations, la lettre p une formule de travail, la lettre q une conjonction de formules de travail, la lettre r une conjonction d'équations, de *fini* et de formules de travail. Ces lettres peuvent être altérées par des primes ou des indices.

$$
\begin{array}{lll}
1 & \neg^1(\exists x\ u{=}u \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ r) \\
2 & \neg^1(\exists x\ v{=}u \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u{=}v \wedge r) \\
3 & \neg^1(\exists x\ u{=}v \wedge u{=}t \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u{=}v \wedge v{=}t \wedge r) \\
4 & \neg^1(\exists x\ u{=}f v_1..v_n \wedge u{=}g w_1..w_m \wedge r) & \Longrightarrow \quad \mathit{vrai} \\
5 & \neg^1(\exists x\ u{=}f v_1..v_n \wedge u{=}f w_1..w_n \wedge r) & \Longrightarrow \quad \neg^1(\exists x\ u{=}f v_1..v_n \wedge \bigwedge_{i=1}^{n} v_i{=}w_i \wedge r) \\
6 & \neg^1(\exists x\ r) & \Longrightarrow \quad \neg^2(\exists x\ r) \\
 & & \\
7 & \neg^2(\exists x\ \mathit{fini}(u) \wedge \mathit{fini}(u) \wedge r) & \Longrightarrow \quad \neg^2(\exists x\ \mathit{fini}(u) \wedge r) \\
8 & \neg^2(\exists x\ u{=}v \wedge \mathit{fini}(u) \wedge r) & \Longrightarrow \quad \neg^2(\exists x\ u{=}v \wedge \mathit{fini}(v) \wedge r) \\
9 & \neg^2(\exists x\ \mathit{fini}(u) \wedge a \wedge q) & \Longrightarrow \quad \mathit{vrai} \\
10 & \neg^2(\exists x\ u{=}f v_1..v_n \wedge \mathit{fini}(u) \wedge r) & \Longrightarrow \quad \neg^2(\exists x\ u{=}f v_1..v_n \wedge \bigwedge_{i=1}^{n} \mathit{fini}(v_i) \wedge r) \\
11 & \neg^2(\exists x\ r) & \Longrightarrow \quad \neg^3(\exists x\ r) \\
 & & \\
12 & \neg^4(\exists x\ a \wedge q \wedge \neg^0(\exists y\ r)) & \Longrightarrow \quad \neg^4(\exists x\ a \wedge q \wedge \neg^1(\exists y\ a \wedge r)) \\
13 & \neg^4(\exists x\ a \wedge a' \wedge q \wedge \neg^3(\exists y\ a'' \wedge r)) & \Longrightarrow \quad \neg^4(\exists x\ a \wedge q \wedge \neg^4(\exists y\ a \wedge r)) \\
14 & \neg^4(\exists x\ a \wedge \neg^5(\exists y\ a) \wedge q) & \Longrightarrow \quad \mathit{vrai} \\
15 & \neg^4(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg^5(\exists y_i\ b_i)) & \Longrightarrow \quad \neg^5(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg^5(\exists y'_i\ b'_i[y'_i])) \\
 & & \\
16 & \neg^4 \left[\begin{array}{l} \exists x\ a \wedge q \wedge \\ \neg^5 \left[\begin{array}{l} \exists y\ b \wedge \\ \bigwedge_{i=1}^{n} \neg^5(\exists z_i\ c_i) \end{array} \right] \end{array} \right] & \Longrightarrow \quad \left[\begin{array}{l} \neg^4(\exists x\ a \wedge q \wedge \neg^5(\exists y\ b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists x_i y_i z_i\ c_i[x_i y_i] \wedge q[x_i]) \end{array} \right]
\end{array}
$$

Dans ces règles, les variables u et v sont telles que $v \prec u$, les symboles f et g sont distincts.

- Dans la règle 6, la conjonction des équations de r est résolue.
- Dans la règle 9, la variable u est accessible depuis elle-même dans la conjonction a.
- Dans la règle 10, la variable u n'est pas accessible depuis elle-même dans la conjonction a.
- Dans la règle 11, la conjonction des équations et des *fini* dans r est résolue.
- Dans la règle 13, a et a'' sont des conjonctions d'équations qui ont les mêmes membres gauches d'équations, a' est une conjonction de contraintes *fini*.
- Dans la règle 15, aucune conjonction b_i n'est égale à a, et les conjonctions et les vecteurs de variables sont calculés comme suit. Soit $x = x'x''x'''$, $a = a' \wedge a'' \wedge a'''$, où :
  - (x', a') est le couple des variables et des équations et des *fini* accessibles de $\exists x\ a$,
  - a'' est la conjonction des équations inaccessibles dans a, x'' est le vecteur des membres gauches de a'',
  - a''' est la conjonction des *fini* inaccessibles dans a,
  - $b_i^\star$ est la conjonction obtenue de b_i en enlevant les *fini* figurant dans a''',
  - (y'_i, b'_i) est le couple des variables et des équations et des *fini* accessibles dans $\exists y_i x''\ b_i^\star$,
  - l'ensemble K est un sous-ensemble de $1..n$ tel que $i \in K$ ssi aucune variable de x''' ne figure dans b'_i.

– Dans la règle 16, dans les $q[x_i]$, les occurrences de $\neg^k$ sont toutes remplacées par $\neg^0$.

Correction des règles de réécriture

Propriété 5.16 *Toute application répétée de ces règles de réécriture sur une formule de travail initiale p se termine et produit une formule de travail finale, équivalente à p dans* ***T****∪FINI.*

Preuve Par le même raisonnement que celui du début de la démonstration de la propriété 4.17, on montre que si l'application des règles termine, on obtient une formule de travail finale. Il reste à montrer que l'application des règles termine toujours et que ces règles conservent l'équivalence dans **T**∪FINI et la forme de travail des formules.

Terminaison des applications des règles On utilise la fonction α introduite dans la démonstration de la propriété 4.17. Rappelons que

$$\begin{array}{l}\alpha(\mathit{vrai}) = 0,\\ \alpha(\neg(\exists x\ a \wedge p_1 \wedge \cdots \wedge p_n)) = 2^{\alpha(p_1)+\cdots+\alpha(p_n)}\end{array}$$

Rappelons également que $\alpha(p_2) < \alpha(p_1)$ entraîne $\alpha(p[p_2]) < \alpha(p[p_1])$, où $p[p_2]$ est la formule obtenue de p lorsque l'on remplace l'occurrence de la formule p_1 dans p par p_2.

On introduit ensuite la fonction $\gamma : (u, a) \mapsto n$, avec u une variable, a une conjonction d'équations et de *fini* et n en entier, définie comme suit :

$$\gamma(u,a) = \begin{cases} 0, & \text{si la conjonction des équations de } a \text{ n'est pas résolue, sinon}\\ 1, & \text{si } u \text{ n'est pas le membre gauche d'une équation de } a \text{ ou}\\ & \text{si } u \text{ est accessible depuis elle-même dans } a, \text{ sinon}\\ 1+\gamma(v,a), & \text{si l'équation } u = v \text{ est dans } a,\\ 2+\sum_{i=1}^{n}\gamma(v_i,a), & \text{si l'équation } u = fv_1...v_n \text{ est dans } a.\end{cases}$$

Remarquons qu'une fois la conjonction des équations de a soit résolue, $\gamma(u, a)$ est fixé pour toute variable u. Sur la conjonction p de formules de travail initiales, on introduit les entiers non négatifs n_i, avec $i \in 1..10$ comme suit :

– n_1, la valeur $\alpha(p)$,
– n_2, le nombre de $\neg^0$,
– n_3, le nombre de $\neg^1$,
– n_4, le nombre de terme dans les sous-formules $\neg^1(\ldots)$,
– n_5, la somme des numéros de variables dans les sous-formules $\neg^1(\ldots)$,
– n_6, le nombre d'équations de la forme $v = u$ avec $v \prec u$ dans les sous-formules $\neg^1(\ldots)$,
– n_7, le nombre de $\neg^2$,
– n_8, la somme des $\gamma(u, a)$, pour toute variable u figurant dans une occurrence de *fini*(u) dans la conjonction a de toute sous-formule de la forme $\neg^2(\exists x\ a \wedge q)$,
– n_9, le nombre de $\neg^3$,
– n_{10}, le nombre de $\neg^4$.

Pour chaque règle, il existe un indice i telle que l'application de la règle garde inchangés les n_j, avec $1 \leq j < i$ et diminue n_i. Ces i valent 1 pour les règles 4, 9, 14 et 16, vaut 2 pour la règle 12, vaut 3 pour la règle 6, vaut 4 pour la règle 5, valent 5 pour les règles 1, 3, vaut 6 pour la règle 2,

vaut 7 pour la règle 11, valent 8 pour les règles 7, 8 et 10, vaut 9 pour la règle 13 et vaut 1 ou 10 pour la règle 15. Du fait que les n_i sont tous entiers non négatifs, ils ne peuvent être diminués infiniment. L'application des règles se termine donc.

Correction On montre maintenant que les règles conservent l'équivalence de formules et la forme de travail. La correction des règles 1-5 est due aux deux premiers axiomes de la théorie **T**∪FINI. Les règles 6 et 11 sont purement syntaxiques. Les règles 7, 8 sont correctes dans la théorie vide donc dans **T**∪FINI. Dans la règle 9, la variable u est accessible à partir d'elle-même, c'est-à-dire que a contient une sous-conjonction de la forme

$$u = t_1 \wedge u_2 = t_2 \wedge \cdots \wedge u_n = t_n$$

où u_i figure dans t_{i-1} avec $i \in 2..n$ et u figure dans t_n. D'après le quatrième axiome de la théorie **T**∪FINI, on a **T**∪FINI $\models a \rightarrow \neg \mathit{fini}(u)$, la règle 9 est donc correcte. La règle 10 est correcte dû au dernier axiome de **T**∪FINI. Les règles 12 et 14 sont correctes dans la théorie vide donc dans **T**∪FINI. La règle 13 est correcte due à la propriété 5.5.

Considérons maintenant la règle 15 :

$$\neg^4(\exists x\ a \wedge \bigwedge_{i=1}^{n} \neg^5(\exists y_i\ b_i)) \implies \neg^5(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg^5(\exists y'_i\ b'_i))$$

où aucune conjonction b_i n'est égale à a, et où les vecteurs x', y'_i et les conjonctions a', b'_i sont calculés comme suit. Décomposons $x = x'x''x'''$, $a = a' \wedge a'' \wedge a'''$ où :

- (x', a') est le couple des variables et des équations et des *fini* accessibles dans $\exists x\, a$,
- a'' est la conjonction des équations inaccessibles dans a, x'' est le vecteur des membres gauches d'équation de a'',
- a''' est la conjonction des formules *fini* non accessibles dans a,
- $b_i^\star$ est la conjonction obtenue de b_i en enlevant les *fini* figurant dans a''',
- (y'_i, b'_i) est le couple des variables et des équations accessibles dans $\exists y_i x''\ b_i^\star$,
- $K \subseteq \{1..n\}$ est tel que $i \in K$ ssi les variables de x''' ne figurent pas dans b'_i.

D'après la propriété 5.4, **T**∪FINI $\models \exists! x''\ a''$. De plus, **T**∪FINI $\models \exists x'''\ a'''$. La formule $\neg(\exists x'\ a')$ est donc équivalente à $\neg(\exists x\, a)$ dans la théorie **T**∪FINI, d'où les deux formules de la règle sont équivalentes lorsque $n = 0$.

Si $n > 0$, d'après la propriété 2.13, la formule gauche de cette règle est équivalente à

$$\neg(\exists x'x'''\ a' \wedge a''' \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i x''\ a'' \wedge b_i))$$

Du fait que l'on peut enlever des b_i les *fini* figurant dans a''' tout en gardant l'équivalence des formules, et du fait que les équations dans a'' figurent aussi dans b_i, cette formule est équivalente à

$$\neg(\exists x'x'''\ a' \wedge a''' \wedge \bigwedge_{i=1}^{n} \neg(\exists y_i x''\ b_i^\star))$$

Appliquant cette même règle dans le cas où $n = 0$ sur les sous-formules $\neg(\exists y_i x''\ b_i^\star)$, nous obtenons des formules équivalentes $\neg(\exists y'_i\, b'_i)$. La formule est donc équivalente à

$$\neg(\exists x'x'''\ a' \wedge a''' \wedge \bigwedge_{i=1}^{n} \neg(\exists y'_i\ b'_i))$$

L'ensemble des indice $1..n$ est décomposé en deux sous-ensemble K et $\{1..n\} - K$ tels que, $i \in K$ ssi aucune variable de x''' ne figure dans b'_i. Du fait que les variables de x''' ne figurent pas dans a, cette formule est équivalente à

$$\neg(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg(\exists y'_i\ b'_i) \wedge (\exists x'''\ a''' \wedge \bigwedge_{i\in\{1..n\}-K} \neg(\exists y'_i\ b'_i))) \tag{5.11}$$

Du fait que dans les formules $\neg(\exists y'_i\ b'_i)$, toutes les variables, les équations et les formules *fini* sont accessibles, la conjonction de ces formules avec $i \in \{1..n\} - K$ satisfait les conditions de la propriété 5.7. On a donc

$$\mathbf{T}\cup\text{FINI} \models \exists x'''\ a''' \wedge \bigwedge_{i\in\{1..n\}-K} \neg(\exists y'_i\ b'_i)$$

La formule (5.11) est donc équivalente à

$$\neg(\exists x'\ a' \wedge \bigwedge_{i\in K} \neg(\exists y'_i\ b'_i))$$

Dans cette formule, chaque conjonction b'_i est différente de a'. Effectivement, du fait que b_i est différent de a, b_i contient soit une équation accessible $u_i = t_i$, soit une contrainte accessible $\mathit{fini}(u_i)$ qui ne figure pas dans a. Si c'est une équation, de même raisonnement que dans la théorie **T**, on montre que cette équation figure dans b'_i. Si c'est une contrainte $\mathit{fini}(u_i)$ et que u_i ne figure pas dans y_i, alors cette contrainte est accessible dans $\exists y_i x''\ b^\star_i$, du fait que u_i ne peut pas figurer dans x''. Cette contrainte figure donc dans b'_i. Si u_i figure dans y_i, u_i ne figure pas dans a. Du fait que $\mathit{fini}(u)$ est accessible dans $\exists y_i\ b_i$, elle est accessible dans $\exists y_i x''\ b^\star_i$. Par conséquent, cette contrainte $\mathit{fini}(u_i)$ figure dans b'_i. En mettant les indices de négation appropriés, cette formule devient la formule droite de la règle 15. Cette règle est donc correcte.

Considérons la règle 16 :

$$\neg^4(\exists x\ a \wedge q \wedge \neg^5(\exists y\ b \wedge \bigwedge_{i=1}^{n} \neg^5(\exists z_i\ c_i))) \implies \left[\begin{array}{l}\neg^4(\exists x\ a \wedge q \wedge \neg^5(\exists y\ b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists x_i y_i z_i\ c_i[x_i y_i] \wedge q[x_i])\end{array}\right]$$

La formule gauche est équivalente à

$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ b \wedge \neg(\bigvee_{i=1}^{n} (\exists z_i\ c_i))))$$

Du fait que b est résolue et toutes les variables de y sont accessibles dans $\exists y\ b$, d'après la propriété 5.3, $\mathbf{T}\cup\text{FINI} \models \exists ?y\ b$. D'après la propriété 2.12, la formule précédente est équivalente à

$$\neg(\exists x\ a \wedge q \wedge \neg((\exists y\ b) \wedge \neg(\exists y\ b \wedge \bigvee_{i=1}^{n} (\exists z_i\ c_i))))$$

En distribuant le $\exists$ sur le $\vee$ et en faisant descendre la deuxième occurrence de négation, la formule devient

$$\neg(\exists x\ a \wedge q \wedge (\neg(\exists y\ b) \vee \bigvee_{i=1}^{n} (\exists y z_i\ b \wedge c_i)))$$

En distribuant de $\wedge$ sur $\vee$ et en faisant descendre la première occurrence de négation, la formule devient

$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ b)) \wedge \bigwedge_{i=1}^{n} \neg(\exists xyz_i\ a \wedge b \wedge c_i \wedge q)$$

D'après la définition de forme de travail, on a $\mathbf{T} \cup \text{FINI} \models c \rightarrow a \wedge b$, la formule est donc équivalente à

$$\neg(\exists x\ a \wedge q \wedge \neg(\exists y\ b)) \wedge \bigwedge_{i=1}^{n} \neg(\exists xyz_i\ c_i \wedge q)$$

En mettant les indices de négation appropriés, cette dernière formule est la formule droite de la règle 16. Cette règle est donc justifiée. Ceci termine la preuve du théorème. □

5.1.4 Algorithme de résolution de contraintes dans T∪FINI

Algorithme de résolution

Comme dans la théorie **T**, la résolution d'une formule du premier ordre p dans $\mathbf{T} \cup \text{FINI}$ consiste à :

1. Transformer la formule p en une formule de travail initiale équivalente dans la théorie vide.
2. Appliquer des règles de réécriture autant de fois que possible. D'après la propriété 5.16, ce processus se termine et produit une formule de travail finale, équivalente à p dans $\mathbf{T} \cup \text{FINI}$, de la forme :

$$\neg^0 \bigwedge_{i=1}^{n} \neg^5(\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg^5(\exists y_{ij}\ a_{ij} \wedge b_{ij})), \qquad \text{avec } n \geq 0,\ n_i \geq 0,$$

   où les équations et les formules *fini* de a_{ij} figurent dans a_i.
3. Extraire la formule résolue équivalente, par la propriété 5.14 :

$$\bigvee_{i=1}^{n} (\exists x_i\ a_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists y_{ij}\ b_{ij}))$$

Soit $\mathcal{M}$ un modèle de la théorie $\mathbf{T} \cup \text{FINI}$. Des propriétés 5.16, 5.14, 5.11 et des corollaires 5.12 et 5.15, on conclut que :

Théorème 5.17 *Cet algorithme transforme toute formule du premier ordre en une formule résolue q équivalente dans $\mathbf{T} \cup \text{FINI}$. Cette formule q est la constante logique vrai si $\mathbf{T} \cup \text{FINI} \models p$, et la constante logique faux si $\mathbf{T} \cup \text{FINI} \models \neg p$. De plus, dans tout modèle $\mathcal{M}$ de $\mathbf{T} \cup \text{FINI}$:*

1. *Toute var(p)-base finie de p est visible dans q.*
2. *Si la formule de travail initiale est de la forme $\neg^0\neg^4\neg^0\neg^0 p'$, toute var$(p)$-base finie de $\neg p$ est visible dans $\neg q$.*

De ce théorème, il s'ensuit que

Corollaire 5.18 (complétude) *La théorie $\mathbf{T} \cup \text{FINI}$ est complète.*

Exemple de résolution d'une contrainte

Reprenons le premier jeu introduit dans le chapitre 2 et résolvons la contrainte $\mathit{fini}(u) \wedge \neg\exists v\ \mathit{coup}(u,v)$:

$$\mathit{fini}(u) \wedge \neg\exists v\ (u=sv \vee (\exists w_1\ u=sw_1 \wedge w_1=sv) \vee (u=v \wedge \neg(u=0) \wedge \neg(\exists w_2\ u=sw_2))).$$

Cette formule mise sous forme normalisée devient

$$\neg\neg(\mathit{fini}(u) \wedge \neg\exists v\neg(\neg(u=sv) \wedge \neg(\exists w_1\ u=sw_1 \wedge w_1=sv) \wedge \neg(u=v \wedge \neg(u=0) \wedge \neg(\exists w_2\ u=sw_2))))$$

puis sous forme de formule de travail initiale

$$\neg^0\neg^4\neg^0\neg^0(\mathit{fini}(u) \wedge \neg^0\exists v\neg^0 \begin{bmatrix} \neg^0(u=sv) \wedge \\ \neg^0(\exists w_1\ u=sw_1 \wedge w_1=sv) \wedge \\ \neg^0(u=v \wedge \neg^0(u=0) \wedge \neg^0(\exists w_2\ u=sw_2)) \end{bmatrix}).$$

L'application des règles 1–15, en simplifiant les trois sous-formules entre crochets, donne

$$\neg^0\neg^4\neg^4\neg^4(\mathit{fini}(u) \wedge \neg^4(\exists v\ \mathit{fini}(u) \wedge \neg^4 \begin{bmatrix} \mathit{fini}(u) \wedge \\ \neg^5(u=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^5(\exists w_1\ u=sw_1 \wedge w_1=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^5 \begin{bmatrix} v=u \wedge \mathit{fini}(u) \wedge \neg^5(v=u \wedge u=0) \wedge \\ \neg^5(\exists w_2\ v=u \wedge u=sw_2 \wedge \mathit{fini}(w_2)) \end{bmatrix} \end{bmatrix})).$$

L'application de la règle 16 sur la sous-formule $\neg^4$ la plus interne donne :

$$\neg^0\neg^4\neg^4\neg^4(\mathit{fini}(u) \wedge \neg^4 \begin{bmatrix} \exists v\ \mathit{fini}(u) \wedge \\ \neg^4 \begin{bmatrix} \mathit{fini}(u) \wedge \\ \neg^5(u=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^5(\exists w_1\ u=sw_1 \wedge w_1=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^5(v=u \wedge \mathit{fini}(u)) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} v=u \wedge u=0 \wedge \\ \neg^0(u=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^0(\exists w_{11}\ u=sw_{11} \wedge w_{11}=sv \wedge \mathit{fini}(v)) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} \exists w_2\ v=u \wedge u=sw_2 \wedge \mathit{fini}(w_2) \wedge \\ \neg^0(u=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^0(\exists w_{12}\ u=sw_{12} \wedge w_{12}=sv \wedge \mathit{fini}(v)) \end{bmatrix} \end{bmatrix}).$$

Puis les règles 1–15 s'appliquent et transforment la formule en :

$$\neg^0\neg^4\neg^4\neg^4(\mathit{fini}(u) \wedge \neg^4 \begin{bmatrix} \exists v\ \mathit{fini}(u) \wedge \\ \neg^5 \begin{bmatrix} \mathit{fini}(u) \wedge \\ \neg^5(u=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^5(\exists w_1\ u=sw_1 \wedge w_1=sv \wedge \mathit{fini}(v)) \wedge \\ \neg^5(v=u \wedge \mathit{fini}(u)) \end{bmatrix} \wedge \\ \neg^5(v=u \wedge u=0) \wedge \\ \neg^5(\exists w_2\ v=u \wedge u=sw_2 \wedge \mathit{fini}(w_2)) \end{bmatrix}).$$

L'application de la règle 16 sur la sous-formule $\neg^4$ la plus interne, donne :

$$\neg^0\neg^4\neg^4\neg^4(\mathit{fini}(u) \wedge \begin{bmatrix} \neg^4 \begin{bmatrix} \exists v\ \mathit{fini}(u) \wedge \neg^5(\mathit{fini}(u)) \wedge \\ \neg^5(v = u \wedge u = 0) \wedge \\ \neg^5(\exists w_2\ v = u \wedge u = sw_2 \wedge \mathit{fini}(w_2)) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} \exists v_1\ u = sv_1 \wedge \mathit{fini}(v_1) \wedge \\ \neg^0(v_1 = u \wedge u = 0) \wedge \\ \neg^0(\exists w_{21}\ v_1 = u \wedge u = sw_{21} \wedge \mathit{fini}(w_{21})) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} \exists v_2 w_1\ u = sw_1 \wedge w_1 = sv_2 \wedge \mathit{fini}(v_2) \wedge \\ \neg^0(v_2 = u \wedge u = 0) \wedge \\ \neg^0(\exists w_{22}\ v_2 = u \wedge u = sw_{22} \wedge \mathit{fini}(w_{22})) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} \exists v_3\ v_3 = u \wedge \mathit{fini}(u) \wedge \\ \neg^0(v_3 = u \wedge u = 0) \wedge \\ \neg^0(\exists w_{23}\ v_3 = u \wedge u = sw_{23} \wedge \mathit{fini}(w_{23})) \end{bmatrix} \end{bmatrix}).$$

L'applications des règles 1–15 sur les formules dans la grande colonne donnent :

$$\neg^0\neg^4\neg^4\neg^4(\mathit{fini}(u) \wedge \begin{bmatrix} \neg^5(\exists v_1\ u = sv_1 \wedge \mathit{fini}(v_1)) \wedge \\ \neg^5(\exists v_2 w_1\ u = sw_1 \wedge w_1 = sv_2 \wedge \mathit{fini}(v_2)) \wedge \\ \neg^5 \begin{bmatrix} \mathit{fini}(u) \wedge \\ \neg^5(u = 0) \wedge \\ \neg^5(\exists w_{23}\ u = sw_{23} \wedge \mathit{fini}(w_{23})) \end{bmatrix} \end{bmatrix}).$$

Encore une fois, l'application de la règle 16 donne :

$$\neg^0\neg^4\neg^4 \begin{bmatrix} \neg^4 \begin{bmatrix} \mathit{fini}(u) \wedge \neg^5(\mathit{fini}(u)) \wedge \\ \neg^5(\exists v_1\ u = sv_1 \wedge \mathit{fini}(v_1)) \wedge \\ \neg^5(\exists v_2 w_1\ u = sw_1 \wedge w_1 = sv_2 \wedge \mathit{fini}(v_2)) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} u = 0 \wedge \\ \neg^0(\exists v_{11}\ u = sv_{11} \wedge \mathit{fini}(v_{11})) \wedge \\ \neg^0(\exists v_{21} w_{11}\ u = sw_{11} \wedge w_{11} = sv_{21} \wedge \mathit{fini}(v_{21})) \end{bmatrix} \wedge \\ \neg^4 \begin{bmatrix} \exists w_{23}\ u = sw_{23} \wedge \mathit{fini}(w_{23}) \\ \neg^0(\exists v_{12}\ u = sv_{12} \wedge \mathit{fini}(v_{12})) \wedge \\ \neg^0(\exists v_{22} w_{12}\ u = sw_{12} \wedge w_{12} = sv_{22} \wedge \mathit{fini}(v_{22})) \end{bmatrix} \end{bmatrix}$$

Les règles 1–15 simplifient la première et la dernière sous-formule $\neg^4$ en *vrai*. La formule devient :

$$\neg^0\neg^4\neg^4(\neg^5(u = 0)).$$

Les règles simplifient donc la formule en $\neg^0\neg^5(u = 0)$. La formule résolue obtenue est $u = 0$.

5.2 Résolution dans la théorie des arbres finis

Dans les parties précédentes, nous avons présenté les algorithmes de résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis **T** et dans la théorie **T**∪FINI possédant la contrainte *fini*. Nous nous intéressons maintenant à la résolution dans la théorie $\mathbf{T}_f$ des arbres finis.

5.2.1 La théorie $\mathbf{T}_f$ des arbres finis

Nous rappelons que l'ensemble **F** a au moins un symbole d'arité nulle et une infinité de symboles d'arités non nulles. La théorie $\mathbf{T}_f$ des arbres finis est l'ensemble des propositions de l'une des formes suivantes :

$$\forall x \forall y \; fx = gy \rightarrow \mathit{faux} \tag{5.12}$$

$$\forall x \forall y \; fx = fy \rightarrow \bigwedge_i x_i = y_i \tag{5.13}$$

$$\forall x \forall u \; u = t[ux] \rightarrow \mathit{faux} \tag{5.14}$$

où f et g sont des éléments distincts pris dans **F**, où x est un vecteur composé de variables x_i, où y est un vecteur composé de variables y_i, où u est une variable et où $t[ux]$ est un terme faisant intervenir au moins un symbole fonctionnel et faisant intervenir la variable u et des variables figurant dans x.

Comme dans la théorie des arbres finis ou infinis **T**, la propriété (5.12), appelée *conflit de symboles*, exprime que des opérations notées différemment produisent des individus distincts. La propriété (5.13), appelée *d'explosion*, exprime en plus que la même opération, lorsqu'elle s'applique sur des objets distincts, produit deux individus distincts. La différence de la théorie $\mathbf{T}_f$ par rapport à la théorie **T** consiste dans la propriété (5.14). Cette propriété interdit qu'un individu soit le résultat de plusieurs opérations sur lui-même. Par conséquence il n'existe pas de suites infinies d'opérations.

Suite aux propriétés (5.12) et (5.13), cette théorie $\mathbf{T}_f$ a toujours la propriété suivante :

$$\mathbf{T}_f \models \forall x \exists! u \; u = t[x] \tag{5.15}$$

où x est un vecteur de variables, u une variable ne figurant pas dans x et $t[x]$ un terme ne faisant intervenir que des variables de x.

L'algèbre des arbres finis $\langle \mathbf{C}, (f^C)_{f \in F} \rangle$ est un modèle de la théorie $\mathbf{T}_f$. De nombreux travaux ont été présentés, qui portent sur la complétude de cette théorie avec une infinité de symboles fonctionnels [31, 36] (La théorie est aussi démontrée décidable dans [38]), ou de la théorie avec un ensemble fini de symboles fonctionnels $\mathbf{T}_f \cup DCA$ (DCA l'axiome de domaine clos), par élimination de quantificateurs [12, 15, 36], ou par la méthode de complétude de modèle [47]. Une axiomatisation du premier ordre des théories de certains types particuliers d'arbres finis avec nombre de branches borné et des arbres finis avec nombre de branches fini et arbitraire a aussi été fournie dans [2].

En utilisant les algorithmes de résolution introduits précédemment, nous présentons deux approches de la résolution d'une formule p dans $\mathbf{T}_f$:

1. Première approche, on utilise la résolution dans la théorie **T**∪FINI. On construire la formule p_c à partir de p en adjoignant la contrainte $\mathit{fini}(u)$ à chaque variable u. De la formule finale q_c de la résolution de p_c dans **T**∪FINI, on extrait la formule finale q en enlevant les occurrences de la contrainte *fini*.
2. Deuxième approche, en modifiant l'algorithme de résolution dans **T** uniquement dans la partie de simplification d'équations. En fait, on introduit le fameux « occur check ».

5.2.2 Première approche

Supposons que l'on veuille résoudre une formule p. Soit V l'ensemble des variables libres de p. Soit p_c la formule

$$p_c = \bigwedge_{u \in V} \mathit{fini}(u) \wedge p'$$

où la formule p' est obtenue de p en remplaçant toute sous-formule de la forme $\neg(\exists x\ a \wedge q)$ en

$$\neg(\exists x\ a \wedge \bigwedge_{u \in X} \mathit{fini}(u) \wedge q)$$

La résolution de la formule p_c dans la théorie **T**∪FINI donne comme résultat la formule q_c. Soit q la formule obtenue depuis q_c en enlevant toutes les occurrences de la contrainte *fini*. Montrons que

$$\mathbf{T}\cup\text{FINI} \models p_c \leftrightarrow q_c \quad \text{entraîne} \quad \mathbf{T}_f \models p \leftrightarrow q$$

On va montrer que $I(p) = I(q)$ pour toute interprétation I satisfaisant $\mathbf{T}_f$. Si une interprétation I satisfait $\mathbf{T}_f$, elle satisfait aussi **T**∪FINI. D'après la construction de p_c, $I(p) = I(p_c)$. Du fait que p_c et q_c sont équivalentes dans **T**∪FINI, $I(p_c) = I(q_c)$. Du fait que $I(\mathit{fini}(u)) = \mathbf{v}$ pour toute occurrence $\mathit{fini}(u)$ dans q_c, on a $I(q_c) = I(q)$. Par conséquent, $I(q) = I(p)$. Ceci étant vrai pour toute interprétation I satisfaisant $\mathbf{T}_f$, les formules p et q sont équivalentes dans $\mathbf{T}_f$.

5.2.3 Deuxième approche

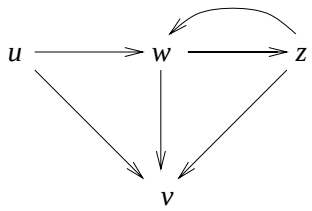
Dans cette approche, la résolution d'une conjonction d'équations est d'abord effectuée en utilisant seulement les deux axiomes (5.12) et (5.13), ensuite on vérifie si la conjonction contient des équations qui contrarient l'axiome (5.14). Chaque fois qu'une conjonction d'équations devient résolue, on vérifie si elle contient des équations qui définissent qu'une variable est accessible à partir d'elle-même.

Soit a une conjonction d'équations. À chaque variable u figurant dans a on associe un sommet u. Sur cet ensemble de sommets, nous mettons des arêtes orientés de telle façon que l'on met une arête (u, v) ssi a contient une équation de la forme $u = t$ où v figure dans t. Appelons le graphe ainsi construit le graphe d'accessibilité de a. Suivant cette construction, la variable v est accessible à partir de la variable u si et seulement s'il existe un chemin menant du sommet u au sommet v. Ainsi la conjonction a définit que la variable u est accessible à partir d'elle-même si et seulement si le sommet u appartient à un cycle. Ceci entraîne que la conjonction a est fausse dans la théorie $\mathbf{T}_f$ des arbres finis si le graphe d'accessibilité correspondant contient un cycle. La vérification qu'un graphe orienté contient un cycle peut s'effectuer par l'algorithme, linéaire en longueur de a, décrit dans le chapitre 23 du livre [17] de Thomas Cormen et al.

Exemple 5.3 Soit a la conjonction suivante

$$u = fvw \wedge v = 1 \wedge w = fvz \wedge z = fvw$$

Correspond à cette conjonction ce graphe



Dans cette conjonction, la variable w est accessible depuis elle-même. Ce graphe contient un cycle passant par les sommets w et z. Cette conjonction est satisfaisable dans **T** mais fausse dans $\mathbf{T}_f$.

Rappelons que pour démontrer la correction de la résolution de formules dans la théorie des arbres finis ou infinis **T**, les propriétés spécifiques de **T** utilisées sont les propriétés 4.5, 4.2, 4.6 et 4.12. Les autre propriétés sont prouvées en utilisant ces propriétés spécifiques et des propriétés générales d'une théorie quelconque.

En ce qui concerne la théorie $\mathbf{T}_f$, montrons que si l'on dit qu'une conjonction d'équations élémentaires est résolue lorsque

- ses membres gauches d'équations sont tous distincts,
- ses équations de la forme $u = v$ sont telles que $v \prec u$, et
- il n'y a pas de variable accessible à partir d'elle-même,

alors les propriétés 4.5, 4.2, 4.6 et 4.12 sont aussi vraies dans $\mathbf{T}_f$. En effet, l'axiome (5.13) entraîne que la propriété 4.5 reste vraie dans $\mathbf{T}_f$. Du fait que les conjonctions résolues ne contiennent pas de cycle, (5.15) entraîne que la propriété 4.2 soit vraie dans $\mathbf{T}_f$. Par l'hypothèse que l'ensemble **F** soit infini et contienne au moins une constante, les propriétés 4.6 et 4.12 sont aussi vraies dans $\mathbf{T}_f$.

Le changement des règles de résolution dans la théorie **T** aux règles de résolution dans la théorie $\mathbf{T}_f$ n'est donc que sur la règle 6. Cette règle en **T** étant

$$6 \quad \neg^1(\exists x\ a \wedge q) \quad \Longrightarrow \quad \neg^2(\exists x\ a \wedge q)$$

avec a une conjonction résolue, devient deux règles suivantes :

$$\begin{array}{lll} 6a & \neg^1(\exists x\ a \wedge q) & \Longrightarrow \quad \textit{vrai} \\ 6b & \neg^1(\exists x\ a \wedge q) & \Longrightarrow \quad \neg^2(\exists x\ a \wedge q) \end{array}$$

où dans la règle $6a$, le graphe d'accessibilité de la conjonction a contient un cycle et dans la règle $6b$, ce graphe ne contient pas de cycle. Par conséquent, dans toute sous-formule de la forme $\neg^k(\exists x\ a \wedge q)$, avec $k \geq 2$, la conjonction a est résolue et ne contient pas de variables accessibles à partir d'elle-même. Le test si une conjonction contient un cycle est fait en temps linéaire en longueur d'écriture de a. Ce changement conserve donc la terminaison des règles de réécriture. Nous pouvons alors conclure que pour toute formule p et toute formule q obtenue de p après une application de ces règles de réécriture,

$$\mathbf{T}_f \models p \leftrightarrow q.$$

De plus, la formule q possède les propriétés d'une formule résolue, c'est-à-dire qu'elle, soit est la constante *vrai*, soit est la constante *faux*, soit contient une variable libre et n'est équivalente ni à *vrai* ni à *faux* dans $\mathbf{T}_f$. Si p ou $\neg p$ a un ensemble fini de solutions dans un modèle $\mathcal{M}$ de $\mathbf{T}_f$ alors ces solutions et non-solutions sont représentées explicitement dans q.

Avec les résultats précédents, nous avons donc une autre façon de montrer la complétude de la théorie $\mathbf{T}_f$ des arbres finis.

Dans [1], L. Albert, R. Casas et F. Fages ont étudié la complexité en moyenne des algorithmes d'unification des arbres finis. S. Vorobyov, en [48] a montré que le problème de décider la validité d'une proposition dans cette théorie $\mathbf{T}_f$ est aussi non-élémentaire, c'est-à-dire que la complexité des algorithmes qui le résolvent n'est bornée par aucune composition finie de

fonctions élémentaires. Il a étudié ensuite la théorie bornée des arbres finis avec l'ensemble **F** fini contient au moins une constante, où l'égalité est remplacée par une famille infinie d'égalités approximatives (deux arbres sont égaux jusqu'à une profondeur donnée). Il a montré que le problème de décider de la validité d'une proposition dans cette théorie a une complexité en espace $2^{2^{cn}}$, avec n la taille de l'entrée et $c > 0$ une constante [49].

Chapitre 6

Implantation

Sommaire

Du système de règles de réécriture de sous-formules présenté dans le chapitre 4, nous pouvons dériver toute une famille d'algorithmes de résolution de contraintes en fixant la stratégie d'application des règles. Nous allons présenter dans ce chapitre la mise en œuvre d'un tel algorithme et les résultats expérimentaux obtenus.

6.1 Détails de l'implantation

6.1.1 Structure de formules

Nous savons qu'une formule normalisée est de la forme

$$\neg(\exists x\ a \wedge p_1 \wedge \cdots \wedge p_n)$$

où x est un vecteur de variables, a une conjonction d'équations et les p_i des formules normalisées de tailles plus courtes. Cette forme normalisée est représentée par une structure imbriquée définie récursivement. Chaque sous-formule p est caractérisée par

- un numéro : la profondeur de p dans la formule à résoudre,
- un pointeur vers sa sur-formule directe,
- un vecteur de variables : les variables quantifiées, où à chaque variable est associé un nombre distinct, sa position dans le vecteur,
- une conjonction d'équations, toujours maintenue sous forme résolue, et
- une conjonction de formules : les sous-formules p_i.

À chaque variable u ayant une occurrence dans p est associé un couple d'entiers non négatifs (i, j), où i est la profondeur de la sous-formule p' dont le vecteur de variables quantifiées x' contient la variable u, et j est le numéro d'identificateur différent de tous les autres. Les variables libres de la formule totale ont $i = 0$. L'ordre $\prec$ sur les variables est définit comme l'ordre lexicographique sur les couples d'entiers correspondants, c'est-à-dire que pour toute variable u et v avec les couples (i, j) et (k, l) associés, on a $u \prec v$ ssi $i < k$ ou $i = k$ et $j < l$.

Rappelons que nous avons imposé les conditions suivantes sur une formule p :

1. Les variables quantifiées sont aussi distinctes que possible.
2. Pour toutes variables u, v et toute sous-formule q de p, si u et v ont des occurrences respectivement libres et liées dans q alors $u \prec v$.

Les deux conditions sont satisfaites. En effet, le deuxième élément du couple associé à chaque variable permet de distinguer la variables des autres. Quant à la deuxième condition, on constate que le premier entier associé à une variable quantifiée est toujours plus grand que celui d'une variable libre. La condition est donc satisfaite.

D'ailleurs, la profondeur d'une variable nous permet de décider immédiatement si une variable est libre dans une formule. Si la profondeur de la variable est égale à la profondeur de la formule, elle doit être définie dans la liste des variables quantifiées de la formule donc elle n'est pas libre. Si la profondeur est plus petite, la variable est libre dans la formule. Remarquons que la profondeur d'une formule ne dépasse jamais celle de la formule dans laquelle elle apparaît.

De plus, associé à chaque variable u un pointeur mbd permettant d'accéder temporairement au terme t, tel que l'équation $u = t$ est définie dans une sous-formule suivant une branche d'imbrication de négations. L'accès est temporaire, car il peut y avoir différents termes t suivant différentes branche d'imbrications. Lors de l'initialisation, ces pointeurs mbd sont NUL. Avec l'utilisation de ces pointeurs, l'organisation d'une conjonction n'est choisie que pour faciliter les parcours des équations de la conjonction. On peut représenter donc une conjonction d'équations par une liste d'équations.

6.1.2 Application des règles

Avec les indices de négation dans les règles de réécriture, la stratégie d'application des règles ne dépend que du choix de la sous-formule sur laquelle on appliquera une règle sauf pour la règle 11

$$\neg^3 \left[\begin{array}{l} \exists x\ a \wedge q[x] \wedge \\ \neg^4 \left[\begin{array}{l} \exists y\ b \wedge \\ \bigwedge_{i=1}^{n} \neg^4(\exists z_i\ c_i[xy]) \end{array} \right] \end{array} \right] \implies \left[\begin{array}{l} \neg^3(\exists x\ a \wedge q \wedge \neg^4(\exists y\ b)) \wedge \\ \bigwedge_{i=1}^{n} \neg^3(\exists x_i y_i z_i\ c_i[x_i y_i] \wedge q[x_i]) \end{array} \right]$$

pour laquelle intervient aussi le choix de $q[x]$. Nous avons choisi la stratégie suivante :

- Dans la règle 11, la formule q est une conjonction de formules de travail de profondeur inférieure ou égale à 2, où les négations sont toutes d'indice 4.
- Pour choisir une sous-formule sur laquelle on appliquera une règle, on prend la sous-formule la plus à gauche, s'il n'y a pas de sous-formule de la forme $\neg(\exists x\, vrai)$.

Simplification des équations

La simplification de la conjonction d'équations d'une formule p est représentée par "Algorithm 1" (voir page 75) et "Algorithm 2" (voir page 76) donnés ci-dessous. Les équations

d'une formule ne sont pas physiquement propagées dans les sous-formules. On simplifie les équations d'une sous-formule avec celles de ses sur-formules à l'aide des pointeurs vers des sur-formules directes et à l'aide de la pile $restaure$, qui mémorise l'état initial des équations modifiées et donc permet de restaurer ces équations une fois que la simplification est terminée. On utilise également la pile $ajout$, qui mémorise les équations nouvellement créées et qui, lorsque la simplification se termine, permet d'ajouter ces nouvelles équations dans la conjonction d'équation de la sous-formule. Cette pile se révèle aussi utile pour éliminer des redondances, comme on le verra dans la prochaine section.

En plus des règles de simplification d'équations, on utilise la règle

$$u = v \wedge v = w \implies u = w \wedge v = w$$

Avec cette règle, une suite d'enchaînement d'équations

$$u_1 = u_2 \wedge u_2 = u_3 \wedge \cdots \wedge u_{n-1} = u_n \tag{6.1}$$

devient

$$u_1 = u_n \wedge u_2 = u_n \wedge \cdots \wedge u_{n-1} = u_n \tag{6.2}$$

En maintenant cette forme, toute comparaison entre deux variables dans la suite est faite en un nombre constant de réécriture. La conjonction de deux suites

$$\begin{array}{l} u_1 = u_n \wedge u_2 = u_n \wedge \cdots \wedge u_{n-1} = u_n \wedge \\ v_1 = v_m \wedge v_2 = v_m \wedge \cdots \wedge v_{m-1} = v_m \end{array}$$

avec ajout d'une équation entre deux variables de ces deux suites demande un nombre de réécriture linéaire en $\max(n, m)$, du même ordre que sans utilisation de cette règle. Cette règle nous permet d'éviter le cas où la formule résolue contient des suites d'équations de la forme (6.1), où les variables u_i sont quantifiées, u_1 est accessible et les variables $u_2, \ldots, u_n$ ne figure pas dans d'autres équations. En mettant ces suites sous la forme (6.2), les variables $u_2, \ldots, u_{n-1}$ deviennent non accessibles. L'application de cette règle sur une conjonction d'équations a est réalisée par l'appelle de la fonction Compacte(a).

Résolution d'une formule

La résolution d'une formule, adoptant la stratégie où dans la règle 11, q est une conjonction de formules de travail finales, est détaillée dans "Algorithm 3" (voir page 77) et "Algorithm 4" (voir page 77).

6.1.3 Élimination de redondances

Le test de redondance peut s'effectuer sur la formule courante considérée. Supposons que la formule courante soit $\neg(\exists x\, a \wedge \neg(\exists y\, b \wedge r) \wedge q)$ et que l'on veuille vérifier si :

$$\mathbf{T} \models \neg(\exists x\, a \wedge \neg(\exists y\, b \wedge r) \wedge q) \leftrightarrow \neg(\exists x\, a \wedge q) \tag{6.3}$$

Ici a et b sont des conjonctions d'équations, q et r des conjonctions de formules. D'après la propriété 4.20, on vérifie si :

$$\mathbf{T} \models \neg(\exists xy\, a \wedge b \wedge r \wedge q)$$

Algorithm 1 Réduire_Équations(p)

Require: Formule $p = \neg(\exists x\ a \wedge q)$

Ensure: Simplifier a avec les équations des sur-formules de p

for all équation $u = t$ dans a **do** {Tenir compte d'équations dans a}
  if $u \rightarrow mbd = NUL$ **then**
    $u \rightarrow mbd = t$
  end if
end for
while $p \neq \neg(vrai)$ et il y a une équation $u = t$ dans a avec $u \rightarrow mbd \neq t$ **do**
  Ajouter_Équation$(u = t, p)$
end while
for all équation $u = t$ dans $restaure$ **do**
  $u \rightarrow mgd = t$ {Restaurer les équations modifiées des sur-formules}
end for
if $p \neq vrai$ **then**
  for all équation $u = t$ dans $ajout$ **do** {Ajouter les nouvelles équations}
    mettre $u = t$ dans a
  end for
  for all équation $u = t$ dans a et $u \rightarrow mbd \neq t$ **do** {Enlever les équations de trop}
    enlever $u = t$ de a
  end for
  Compacte(a)
end if

Algorithm 2 Ajouter_Équation($u = t$,p)

$s = u \rightarrow mbd$
if $s = NUL$ **then** {Ajouter une équation nouvelle dans $ajout$}
  mettre $u = t$ dans $ajout$
  $u \rightarrow mbd = t$
end if
if $s = f u_1..u_n$ **then**
  if $t = g v_1..v_m$ **then**
    if $f \neq g$ ou $m \neq n$ **then** {règle 4}
      $p = \neg(vrai)$
    else {règle 5}
      for $i = 1$ à n **do**
        Ajouter_Équation($u_i = v_i$,p)
      end for
    end if
  else {règle 3, t est une variable, l'équation $u = s$ est définie dans une sur-formule de p}
    mettre $u = s$ dans $restaure$
    $u \rightarrow mbd = t$
    Ajouter_Équation($t = s$,p)
  end if
else {s est une variable, règle 3}
  if $t = g v_1..v_m$ ou t est une variable telle que $t \prec s$ **then**
    Ajouter_Équation($s = t$,p)
  else
    Ajouter_Équation($t = s$,p)
  end if
end if

Algorithm 3 Résoudre(p)

Require: Formule $p = \neg(\exists x\ a \wedge q)$

Ensure: Résoudre p, la formule finale sera la première formule de la conjonction de formules résolues équivalente à p

```
Réduire_Équations(p)
if p ≠ ¬(vrai) then
  Marquer_Variables_Accessibles(a, x)
  Résoudre_Conjonction(q)
  if q = ¬(vrai) then
    p = ¬(¬(vrai))
  else
    if p est de profondeur 1 ou 2 then
      Éliminer_Quantification(p){règle 10}
    else
      while p est de profondeur 3 do {règle 11}
        Distribuer(p) { des nouvelles formules sont ajoutées à la conjonction de formules qui contient p}
      end while
    end if
    Réinitialiser_Pointeurs(a){ remettre à NUL les pointeurs mbd des membres gauches des équations de a qui ne figurent pas dans x }
  end if
end if
```

Algorithm 4 Résoudre_Conjonction(q)

Require: Conjonction de formules $q = p_1 \wedge \cdots \wedge p_n$

Ensure: Résoudre q, la conjonction finale est équivalente à q

```
while q ≠ vrai et il y a une formule p_i non résolue do {Parcours les formules de q}
  Résoudre(p_i)
  if p_i = ¬(¬(vrai)) then
    enlever p_i de q
  else
    if p_i = ¬(vrai) then
      q = ¬(vrai)
    end if
  end if
end while
```

Dans notre implantation, nous n'éliminons des redondances qu'avec q une conjonction de formules de profondeur 1, c'est-à-dire des formules de la forme $\neg(\exists z\, c)$. Voici les marches à suivre :

- Initialiser les pointeurs des variables constituant les membres gauches des équations de b si la variable n'est pas définie dans y. Ces variables sont celles qui ont une profondeur plus petite que celle de la formule $\neg(\exists y\, b \wedge r)$.
- Modifier la profondeur et le numéro des variables dans y. La profondeur est diminuée de un, pour être de la même profondeur que celle de x. Les numéros des variables dans y sont augmentés de telle sorte qu'ils soient plus grands que ceux des variables dans x. Ce changement ne modifie pas l'ordre entre les variables de y, donc les propriétés des équations de $b \wedge r$. Du fait qu'aucune variable de y ne figure dans a ni dans q, ce changement ne modifie pas non plus les propriétés des équations dans ces formules.
- Simplifier chaque formule $\neg(\exists z\, c)$ de q et vérifier si la formule simplifiée est $\neg(\exists z \bigwedge_i u_i = t_i)$, avec les u_i figurent dans z (la forme $\neg(\exists z\, \mathit{vrai})$ est un cas spécial de cette forme). Si oui, l'implication (6.3) est vraie. Si toute simplification des formules de q ne donne pas $\neg(\exists z \bigwedge_i u_i = t_i)$, avec les u_i figurent dans z, l'implication (6.3) est fausse. La simplification se fait comme dans "Algorithm 1", sauf dans le dernier test si $cf \neq 0$, on vérifie si la conjonction des équations dans c et dans $ajout$ sont de la forme $u = t$ avec u figurant dans z. Ensuite, la pile $ajout$ est vidée.
- Restaurer la profondeur et le numéro des variables dans y.
- Remettre à zéro les pointeurs des variables membres gauches des équations de b qui ne sont pas définies dans y.

6.1.4 Extension dans la théorie des arbres finis

Disposant déjà une implantation d'un algorithme pour résoudre des contraintes dans la théorie des arbres finis ou infinis, nous adoptons la deuxième approche (voir section 5.2) pour passer à la théorie des arbres finis.

Le seul changement dans l'implantation par rapport à celle de la résolution dans **T** consiste dans "Algorithm 3". Après avoir réduit la conjonction d'équations a, l'appel de Vérifier(p) vérifie si a contient une variable accessible depuis elle-même, c'est-à-dire si le graphe d'accessibilité G_a contient un cycle. Si la conjonction a contient une telle variable, la formule p devient $\neg(\mathit{vrai})$. Pour ceci on est amené à colorer chaque variable par la couleur blanc, gris, noir. La fonction Vérifier est détaillée dans "Algorithme 5" (page 79) et "Algorithme 6" (page 79).

6.2 Benchmarks

Toute formule en entrée est de la forme

$$\neg(\exists u_1 \ldots u_n\; v_1 = t_1 \wedge \cdots \wedge v_m = t_m \wedge p_1 \wedge \cdots \wedge p_l) \tag{6.4}$$

avec les t_i des termes et les p_i des formules. Cette forme est représentée par

```
!([U1 ... Un] V1=t1, ..., Vm=tm, p1, ..., pl)
```

Les variables sont des chaînes de caractères dont le premier est une majuscule. Les symboles fonctionnels sont des chaînes de caractères dont le premier est un minuscule. Si $n = 0$, on peut omettre les crochets `[ ]` et si $m + l = 0$, la conjonction est représentée par `vrai`.

Algorithm 5 Vérifier(p)

Require: Formule $p = \neg(\exists x\ a \wedge q)$
Ensure: Si G_a contient un cycle alors $p = \neg(\mathit{vrai})$
  for all variable $u \in V$ **do** {Initialisation :colorer en blanc les variables}
    $u \rightarrow couleur =$ blanc
  end for
  while $p \neq \neg(\mathit{vrai})$ et l'équation $u = t$ dans a n'est pas considérée **do**
    if $u \rightarrow couleur =$ blanc **then** {u n'est pas visité}
      Visiter(u, p) {Parcours en profondeur}
    end if
  end while

Algorithm 6 Visiter(G, u) Visite en profondeur partant de la variable u

  $u \rightarrow couleur =$ gris {u vient d'être visité}
  while $p \neq \neg(\mathit{vrai})$ et v figurant dans $u \rightarrow mbd$ n'est pas considéré **do**
    if $v \rightarrow couleur =$ gris **then** {v appartient à un cycle}
      $p = \neg(\mathit{vrai})$
    else if $v \rightarrow couleur =$ blanc **then** {v n'est pas visité}
      Visiter(v, p)
    end if
  end while
  $u \rightarrow couleur =$ noir {Colorer en noir la variable u, la visite termine}

Pour être plus à l'aise, nous acceptons des occurrences de prédicats. Les noms de prédicats sont des chaînes de caractères dont le premier est un minuscule. Chaque prédicat est défini par une formule de la forme (6.4). Une occurrence de prédicat est donc considérée comme une sous-formule. Toutes les occurrences dans une formule seront remplacées par leur formule de définition avant la phase de résolution. La formule à résoudre ne fait donc intervenir que la relation d'égalité. Les commentaires sont délimités par `/*` et `*/`.

Constatons que, bien que les formules p et $\neg\neg p$ soient équivalentes, la résolution de ces formules ne donne pas forcément la même formule. Nous choisissons donc de résoudre une formule telle qu'elle est, c'est à l'utilisateur d'assurer que la formule entrée est bien une formule de travail initiale.

Les résultats expérimentaux dépendent bien entendu des performances de l'ordinateur utilisé. Nos programmes sont écrits en langage C++ et sont exécutés sur un processeur Pentium II 350Mhz, de 512Ko de mémoire cache et sous le système Linux 2.1.

6.2.1 Positions gagnantes et perdantes dans un jeu

Les contraintes $gagnant_k(x)$ et $perdant_k(x)$, représentées respectivement par `gagnant[K](X)` et `perdant[K](X)`, sont exprimées comme suit.

```
gagnant[0](X)  :  !(vrai).
gagnant[1+K](X):  !(!(
                      [Y] coup(X,Y),!(
                      [X] coup(Y,X),!(
                      gagnant[K](X))))).

perdant[K](X)  :  ![Y] coup(X,Y),!(gagnant[K](Y))).
```

La contrainte `coup(X,Y)` est définie dépendant de chaque jeu.

Premier jeu

La contrainte `coup(X,Y)` est définie comme suit :

```
coup(X,Y)      :  !(!(X=s(Y)),
                    ![U] X=s(U),U=s(Y)),
                    !(X=Y,!(X=0),![U] X=s(U)))).
```

La résolution de la formule `gagnant[1](X)` donne

```
!(vrai,
  ![U165] X=s(U165), U165=0, vrai),
  ![U167,U168] X=s(U168), U168=s(U167), U167=0, vrai))
```

qui se présente bien $x = s(0) \vee x = s(s(0))$. Celle de la formule `!(!(perdant[1](X)))` donne

```
!(vrai,
  ![U274,U275,U276]
     U274=s(U276), U276=s(U275), U275=0, X=s(U274), vrai),
  !(X=0, vrai))
```

ce qui représente $x = s(s(s(0))) \vee x = 0$.

Deuxième jeu

```
coup(X,Y)  :
    !(
      !(X=Y,!([U,V] X=c(U,V))),
      ![U,V,W]
        !(!(X=c(U,V),Y=c(U,W)),
          !(X=c(V,U),Y=c(W,U))),
        !(!([I] U=g(I),
            !(!([J] V=g(J),W=f(V)),
              !(W=g(V),!(!(V=0),![J] V=f(J)))),
              !(W=V,!(V=0),![J] V=f(J)),![J] V=g(J)))))),
          !(!([I] U=g(I)),
            !(!(W=V,!(V=0),![J] V=f(J)),![J] V=g(J))),
              ![J] V=f(J),
                !(!([K] J=f(K),W=V),
                  !(J=0,W=V),
                  !(W=J, !(J=0),![K] J=f(K)))),
              ![J] V=g(J),
                !(!([K] J=g(K),W=V),
                  !(W=J, !([K] J=g(K))))))))))).
```

La résolution de la formule `gagnant[1](X)` donne

```
!(vrai,
  ![U23385,U23386,U23387]
     U23387=g(U23386), X=c(U23385,U23387), U23385=0, U23386=0, vrai),
  ![U23389,U23390,U23391]
     U23391=g(U23389), X=c(U23391,U23390), U23389=0, U23390=0, vrai))
```

qui se présente $x = c(0, g(0)) \vee x = c(g(0), 0)$. La résolution de `!(!(perdant[1](X)))` donne

```
!(vrai,
  ![U37342,U37343]
     U37342=0, U37343=0, X=c(U37342,U37343), vrai),
  ![U37345,U37346,U37347,U37348]
     U37348=0, U37347=g(U37348), U37345=f(U37347), U37346=0,
     X=c(U37345,U37346), vrai),
  ![U37350,U37351,U37352,U37353]
     U37353=0, U37352=g(U37353), U37350=0, U37351=f(U37352),
     X=c(U37350,U37351), vrai))
```

qui se présente $x = c(0, 0) \vee x = c(f(g(0)), 0) \vee x = c(0, f(g(0)))$.

Troisième jeu

```
coup(X,Y) :
    !(
      !(X=Y,!([U,V] X=c(U,V))),
```

```
!([U1,V1,U2,V2]
  X=c(U1,V1),Y=c(U2,V2),
  !(
    !(V1=0,V2=V1,
      !(!([W] U1=f(W),
          !(!([K] W=g(K),U2=W),!(U2=U1,!([K] W=g(K))))),
        !([W] U1=g(W),
          !(!([K] W=g(K),U2=U1),!(U2=W,!([K] W=g(K))))),
        !(U2=U1,!([W] U1=f(W)),!([W] U1=g(W)),!(U1=0)))),
    !(V1=1,
      !(!([W] U1=g(W),!(!(U2=f(U1),V2=V1),!(U2=U1,V2=0))),
        !(U2=g(U1),!([W] U1=g(W)),!(!(V2=V1),!(V2=0))))),
    !(U2=U1,V2=V1,!(V1=0),!(V1=1))))).
```

La résolution de la formule `gagnant[1](X)` donne

```
!(vrai,
  ![U5435,U5436,U5437]
     U5436=g(U5435), U5437=0, X=c(U5436,U5437), U5435=0, vrai))
```

qui représente $x = c(g(0), 0)$. La résolution de `!(!(perdant[1](X)))` donne

```
!(vrai,
  ![U8544,U8545,U8546,U8547]
     U8547=0, U8546=g(U8547), U8544=f(U8546), U8545=0,
     X=c(U8544,U8545), vrai),
  ![U8549,U8550]
     U8549=0, U8550=0, X=c(U8549,U8550), vrai))
```

qui représente $x = c(f(g(0)), 0) \vee x = c(0, 0)$.

La résolution de ces formules sont effectuées avec l'élimination de redondances dans les sous-formules de profondeur 1. Voici le temps d'exécution en milisecondes de la résolution de formule $gagnant_k(x)$

k	Jeu 1	Jeu 2	Jeu 3
1	0	150	40
2	10	360	70
3	10	610	110
4	20	840	170
5	30	1 180	210
10	300	5 970	600
20	4 270	236 350	2 890
40	89 870	-	30 180
80	3 841 220	-	-

Notons qu'avec $k = 80$, la formule d'entrée contient plus de 160 quantifications imbriquées. L'élimination de redondances permet de réduire le temps d'exécution ainsi que l'espace utilisé. D'autre part, la résolution de ces formules dans la théorie des arbres finis $\mathbf{T}_f$ nécessite un temps

légèrement moins important et un espace plus important. Nous l'illustrons avec l'exemple des temps de résolution des formules $gagnant_k(x)$ du premier jeu dans le tableau suivant :

k	**T** sans élimination	**T** avec élimination	$\mathbf{T}_f$
10	420	300	280
20	8 330	4 270	4 060
40	373 930	89 870	86 150
60	abandonner	688 590	668 840
80	abandonner	3 841 220	abandonner

6.2.2 Contrainte représentant un arbre énorme

```
enorme[K](X) :
        !(!([T,Y,W,Z] T=trois,Y=0,W=0,triangle[K](T,X,Z,Y))).

triangle[0](T,X,Z,Y):
        !(!(Z=X,Z=Y)).

triangle[1+K](T,X,Z,Y):
        !(!([U1,U2] Z=f(X,U1,U2,Y),
            ![T1,Y1,Z1]
              !(!(T1=un),!(T1=deux)),
              triangle[K](T1,Z,Z1,Y1),
              !(T1=deux,forme2(Y1)),
              ![U,V]
                T1=un, forme1(U,Y1,V),
                !(T=un,!(trans1(U,V,Y))),
                !(T=deux,!(trans2(U,V))),
                !(T=trois,!(trans3(U,V,Y)))))))).

forme1(X,Z,Y) :
        !(!([U1,U2,U3,U4,U5,U6,V1,V2]
            Z=f(U1,V1,V2,U6),V1=f(U1,U2,U3,X),V2=f(Y,U4,U5,U6))).

forme2(X) :
        !(!([U1,U2,U3,U4,V1,V2]
            X=f(U1,V1,V2,U4),V1=f(U2,U2,U2,U2),V2=f(U3,U3,U3,U3))).

trans1(X,Y,Z) :
        !(!(X=Z),!(trans2(X,Y))).

trans2(X,Y) :
        !(!([U1,U2,U3,U4] X=f(U1,U2,U3,U4),!(!(Y=U2),!(Y=U3)))).

trans3(X,Y,Z) :
        !(!(X=s(Y))).
```

La formule *énorme*$_k(x)$ est alors équivalente à $x = s^{\delta(k)-1}(0)$. La résolution de `enorme[2](X)` donne

```
!(vrai,
  ![U2568,U2569,U2570]
    U2570=s(U2568), X=s(U2569), U2569=s(U2570), U2568=0, vrai))
```

ainsi que celle de `enorme[3](X)` donne

```
!(vrai,
  ![U34257,U34258,U34259,U34260,U34261,U34262,U34263,
     U34264,U34265,U34266,U34267,U34268,U34269,U34270,U34271]
    U34271=s(U34257), U34263=s(U34270), U34269=s(U34258),
    U34261=s(U34268), U34267=s(U34262), U34259=s(U34266),
    U34265=s(U34260), X=s(U34264), U34270=s(U34271),
    U34268=s(U34269), U34266=s(U34267), U34264=s(U34265),
    U34262=s(U34263), U34260=s(U34261), U34258=s(U34259),
    U34257=0, vrai))
```

Le temps d'exécution en milisecondes de *énorme*$_k(x)$ est

k	0	1	2	3	4
énorme$_k(x)$	$x = 0$	$x = s(0)$	$x = s^3(0)$	$x = s^{15}(0)$	$x = s^{65535}(0)$
temps (ms)	0	0	10	230	-

6.2.3 Autres contraintes

Puzzle d'anniversaire

Ce puzzle est considéré par T. Keisu [30], donné par Michael Fitzpatrick en *The Newsletter of the Association for Logic Programming*, Vol 6/4, Nov 1993, appelé *Birthday Puzzle*.

Sachant que les cadeaux sont une vidéo cassette (vt), un CD (cd), une cassette (ct) et un disque (rt). Alan ne reçoit le cd que si Barry reçoit le vt et David reçoit le ct. Barry ne reçoit le ct que si Carl reçoit le rt et Alan reçoit le vt. David ne reçoit le rt que si Alain reçoit le cd et Barry reçoit le vt. Alan ne reçoit le vt que si Carl reçoit le rt et David reçoit le ct. Barry ne reçoit le vt que si Alan reçoit le rt et David reçoit le ct. Alan ne reçoit le rt que si Barry reçoit le cd et Carl reçoit le vt. Carl ne reçoit le rt que si Barry reçoit le cd et Alan reçoit le ct. Alan ne reçoit le ct que si Barry reçoit le rt et David reçoit le cd. Carl ne reçoit le vt que si David reçoit le cd.

```
cadeau(X) :   !(!(X=vt),!(X=cd),!(X=ct),!(X=rt)).
```

Comme en [30], le puzzle s'exprime par la formule suivante :

```
!(!(cadeau(A),cadeau(B),cadeau(C),cadeau(D),
    !(A=B),!(A=C),!(A=D),!(B=C),!(B=D),!(C=D),
    !(A=cd,!(B=vt,D=ct)),
    !(B=ct,!(C=rt,A=vt)),
    !(D=rt,!(A=cd,B=vt)),
```

```
    !(A=vt,!(C=rt,D=ct)),
    !(B=vt,!(A=rt,D=ct)),
    !(A=rt,!(B=cd,C=vt)),
    !(C=rt,!(B=cd,A=ct)),
    !(A=ct,!(B=rt,D=cd)),
    !(C=vt,!(D=cd)))).
```

La résolution de cette formule donne (en un temps négligeable : 0ms) :

```
!(vrai,
  !(C=vt, B=rt, D=cd, A=ct, vrai))
```

Puzzle du zèbre

On s'intéresse au fameux puzzle du zèbre, aussi appelé le puzzle de cinq maisons par Pascal Van Hentenryck [46]. En adoptant son idée, le puzzle se représente par le programme suivant :

```
/* Les variables utilisées (i=1..5) :
Ni (nationalité) : anglais, espagnol, japonnais, italien, norvégien
Ci (couleur)     : rouge, vert, blanc, jaune, bleu
Ai (animal)      : chien, escagots, renard, cheval, zèbre
Bi (boisson)     : thé, café, lait, jus, eau
Pi (profession)  : peintre, sculpteur, diplomate, violoniste,
                   docteur */

diff(X1,X2,X3,X4,X5):
    !(!(!(X1=X2),!(X1=X3),!(X1=X4),!(X1=X5),
        !(X2=X3),!(X2=X4),!(X2=X5),
        !(X3=X4),!(X3=X5),
        !(X4=X5))).

valeur(X) :
    !(!(X=0),
      !([U] X=s(U),U=0),
      !([U1,U2] X=s(U2),U2=s(U1),U1=0),
      !([U1,U2,U3] X=s(U3),U3=s(U2),U2=s(U1),U1=0),
      !([U1,U2,U3,U4] X=s(U4),U4=s(U3),U3=s(U2),U2=s(U1),U1=0)).
.

/* Formule à résoudre */

!(!([U]
    N1=C1, /* L'anglais vit dans une maison rouge. */
    N2=A1, /* L'espagnol possède un chien.*/
    N3=P1, /* Le japonnais est un peintre. */
    N4=B1, /* L'italien boit le thé. */
    N5=0,  /* Le norvégien vit dans la première maison à gauche.*/
    C2=B2, /* Le propriétaire de la maison verte boit le café. */
```

```
    C2=s(C3),/* La maison verte est à droite de la maison blanche.*/
    P2=A2, /* Le sculpteur élève des escagots.*/
    P3=C4, /* Le diplomate vit dans la maison jaune.*/
    B3=s(U),U=s(N5), /* Le lait est bu dans la maison au milieu.*/
    P4=B4, /* Le violoniste boit jus de fruit.*/
    !(!(N5=s(C5)),!(C5=s(N5))), /* La maison du norvégien est à
                                   côté de la maison bleue.*/
    !(!(A3=s(P5)),!(P5=s(A3))), /* Le renard est dans une maison à
                                   côté de celle du docteur.*/
    !(!(A4=s(P3)),!(P3=s(A4))), /* Le cheval est dans une maison à
                                   côté de celle du diplomate.*/
    valeur(A1),valeur(A2),valeur(A3),valeur(A4),valeur(A5),
    valeur(B1),valeur(B2),valeur(B3),valeur(B4),valeur(B5),
    valeur(C1),valeur(C2),valeur(C3),valeur(C4),valeur(C5),
    valeur(P1),valeur(P2),valeur(P3),valeur(P4),valeur(P5),
    valeur(N1),valeur(N2),valeur(N3),valeur(N4),valeur(N5),
    diff(A1,A2,A3,A4,A5),
    diff(B1,B2,B3,B4,B5),
    diff(C1,C2,C3,C4,C5),
    diff(P1,P2,P3,P4,P5),
    diff(N1,N2,N3,N4,N5)
)).
```

Le temps de résolution est 8690ms. Bien entendu ce temps est beaucoup plus important que ce qu'a pris le solveur de contraintes sur les entiers avec des techniques de réduction d'intervalles [50], mais signalons que les domaines des contraintes sont différents, et que le domaine des entiers est plus adapté à ce genre de contraintes.

Chapitre 7

Conclusion

Concluons par quelques remarques sur l'expressivité de nos contraintes, la complexité des algorithmes pour les résoudre et les extensions qu'on pourrait leur apporter.

En ce qui concerne l'expressivité de nos contraintes, si les exemples du chapitre 3 ont mis en évidence l'apport des quantifications et des négations, ils n'ont pas montré l'apport des arbres infinis. En effet, ces exemples sont aussi corrects dans l'algèbre des arbres finis. Il serait donc intéressant de mettre en avant des exemples plus pointus faisant intervenir des structure cycliques, telles que les automates à ensembles d'états finis, les grammaires hors contexte ou les λ-expressions, ainsi qu'il a été fait en [7, 18] dans le cadre de la programmation logique.

En ce qui concerne la complexité des algorithmes pour résoudre nos contraintes, nous avons vu que dans le pire des cas, elle est proche de l'indécidabilité. Cependant les benchmarks présentés dans le chapitre 6 montrent qu'il existe des classes de problèmes, avec beaucoup de quantifications imbriquées, que l'on peut résoudre en pratique. Nous n'avons malheureusement pas pu caractériser ces classes, mais peut-être que notre thèse sera le point de départ d'une telle étude.

En ce qui concerne les extensions possibles de l'algèbre des arbres, nous nous sommes limité dans le chapitre 5 à introduire le prédicat unaire *fini*. Nous avons cependant aussi envisagé l'introduction du prédicat $\mathit{nat}(u)$ qui impose que l'arbre u soit un entier naturel, c'est-à-dire qu'il soit de la forme $s^n(0)$, avec $n \geq 0$. Cela revient probablement à ajouter l'axiome suivant dans la théorie des arbres :

$$\forall u\ \mathit{nat}(u) \ \leftrightarrow\ (u = 0) \vee (\exists v\ u = sv \wedge \neg(u = v) \wedge \mathit{nat}(v))$$

De cet axiome on doit pouvoir déduire les modifications à apporter à nos règles de réécriture. Une autre extension possible serait de munir l'algèbre des arbres d'opérations d'ajout d'un élément à gauche ou à droit d'une liste, dans le même esprit de ce qui a été fait par T. Rybina et A. Voronkov [44].

Bibliographie

[1] Luc Albert, Rafael Casas, and François Fages. Average-case analysis of unification algorithms. *Theoretical Computer Science*, 113:3–34, 1993.

[2] Rolf Backofen, James Rogers, and K. Vijay-Shanker. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language and Information*, 4(1):5–39, 1995.

[3] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 146(1-2):243–268, July 1995.

[4] Jon Barwise. An introduction to first-order logic. In Jon Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 5–46. North Holland, 1977.

[5] F. Benhamou, P. Bouvier, A. Colmerauer, H. Garetta, B. Giletta, J.L. Massat, G.A. Narboni, S. N'Dong, R. Pasero, J.F. Pique, Touraïvane, M. Van Caneghem, and E. Vétillard. *Le manuel de Prolog IV*. PrologIA, Marseille, Juin 1996.

[6] Hans-Jurgen Bürckert. Solving disequations in equational theories. In *Proc. 9th Conf. on Automated Deduction*, LNCS 310, pages 517–526. Springer-Verlag, 1988.

[7] Alain Colmerauer. Prolog and infinite trees. In K.L. Clark and S-A. Tarnlund, editors, *Logic Programming*, pages 231–251, New York, 1982. Academic Press.

[8] Alain Colmerauer. Equations and inequations on finite and infinite trees. In *Proceeding of the International Conference on Fifth Generation Computer Systems (FCGS-84)*, pages 85–99, Tokyo, 1984. ICOT.

[9] Alain Colmerauer. An introduction to Prolog III. *Communications of the ACM*, 33(7):68–90, 1990.

[10] Alain Colmerauer and Thi Bich Hanh Dao. Expressiveness of full first order constraints in the algebra of finite and infinite trees. In *6th International Conference of Principles and Practice of Constraint Programming, CP'2000*, LNCS 1894, pages 172–186. Springer, 2000.

[11] Alain Colmerauer, Henri Kanoui, and Michel Van Caneghem. Prolog, theoretical principles and current trends. *Technology and Science of Informatics*, 2(4), August 1983. Version anglaise de la revue *TSI*, AFCET-Bordas.

[12] Hubert Comon. *Unification et disunification : Théorie et applications*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1988.

[13] Hubert Comon. Disunification: a survey. In J.L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT Press, 1991.

[14] Hubert Comon. Résolution de contraintes dans des algèbres de termes. Rapport d'Habilitation, Université de Paris Sud, 1992.

[15] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.

[16] Kevin J. Compton and C. Ward Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Annals of Pure and Applied Logic*, 48(1):1–79, 1990.

[17] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. MIT Press, 1990.

[18] Solange Coupet-Grimal and Olivier Ridoux. On the use of advanced logic programming features in computational linguistics. *The Journal of Logic Programming*, 24(1-2):121–159, 1995.

[19] Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25(2):95–169, March 1983.

[20] Bruno Courcelle. Equivalences and transformations of regular systems - applications to program schemes and grammars. *Theretical Computer Science*, 42:1–122, 1986.

[21] Thi Bich Hanh Dao. Résolution de contraintes du premier ordre dans la théorie des arbres finis ou infinis. In *Programmation en logique avec contraintes, JFPLC'2000*, pages 225–240. Hermes Science Publications, 2000.

[22] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[23] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69–115, February 1987.

[24] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[25] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, August 1979.

[26] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley publishing company, 1979.

[27] Gérard Huet. *Résolution d'équations dans les langages d'ordre* $1, 2, \ldots, \omega$. Thèse d'etat, Université Paris 7, 1976.

[28] Joxan Jaffar. Efficient unification over infinite terms. *New Generation Computing*, 2(3):207–219, 1984.

[29] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstrack algebras: A rule-based survey of unifications. In J.L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 257–331. MIT Press, 1991.

[30] Torbjörn Keisu. *Tree Constraints*. PhD thesis, The Royal Institute of Technology,Sweden, 1994.

[31] Kenneth Kunen. Negation in logic programming. *Journal of Logic Programming*, 4:289–308, 1987.

[32] Jean-Louis Lassez, Michael J. Maher, and Kim Marriott. Unification revisited. In *Proc. Workshop on the Foundations of Deductive Database and Logic Programming*, pages 587–625, 1986.

[33] Jean-Louis Lassez and Kim Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasonning*, 3:301–317, 1987.

[34] Jean-Louis Lassez and Ken McAloon. Independence of negative constraints. In *Proc. TAPSOFT 89*, LNCS 351, pages 19–27, Barcelona, 1989.

[35] Roger C. Lyndon. *Notes on Logic*. Van Nostrand Mathematical Studies, 1964.

[36] Michael J. Maher. Complete axiomatization of the algebra of finite, rational and infinite trees. Technical report, IBM - T.J. Watson Research Center, 1988.

[37] Michael J. Maher and Peter J. Stuckey. On inductive inference of cyclic structures. *Annals of Mathematics and Artificial Intelligence*, 15(2):167–208, 1995.

[38] Anatolii Malcev. Axiomatizable classes of locally free algebras of various types. In B. Wells III, editor, *The Metamathematics of Algebraic Systems. Anatolii Ivanovic Malcev. Collected Papers: 1936-1967*, volume 66, chapter 23, pages 262–281. North Holland, 1971.

[39] Alberto Matelli and Ugo Montanari. An efficient unification algorithm. *ACM Trans. on Languages and Systems*, 4(2):258–282, 1982.

[40] Pawel Mielniczuk. Basic theory of feature trees. Submitted to Journal of Symbolic Computation, 1998. Also available at `http://www.tcs.univ.wroc.pl/~mielni`.

[41] Michael S. Paterson and Mark N. Wegman. Linear unification. *Journal of Computer and Systems Science*, 16:158–167, 1978.

[42] Viswanath Ramachandran and Pascal Van Hentenryck. Incremental algorithms for constraint solving and entailment over rational trees. In *Proc. of the 13th Conference Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *LNCS*, pages 205–217, 1993.

[43] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965.

[44] Tatiana Rybina and Andrei Voronkov. A decision procedure for term algebras with queues. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, pages 279–290, 2000.

[45] Donald A. Smith. Constraint operations for CLP($\mathcal{FT}$). In *Logic Programming: Proceedings of the 8th International Conference*, pages 760–774, Paris, 1991.

[46] Pascal Van Hentenryck. *Constraint satisfaction in logic programming*. Logic Programming Series. The MIT Press, Cambridge, Massachussetes, 1989.

[47] Sergei Vorobyov. Theory of finite trees revisited: Application of model-theoretic algebra. Technical report, Centre de recherche en informatique de Nancy, 1994. CRIN-94-R-135.

[48] Sergei Vorobyov. An improved lower bound for the elementary theories of trees. In *Proceeding of the 13th International Conference on Automated Deduction, CADE'96*, volume 1104 of *Springer Lecture Notes in Artificial Intelligence*, pages 275–287, 1996.

[49] Sergei Vorobyov. On the bounded theories of finite trees. In *Proc. of 2nd Asian Computing Science Conference*, volume 1179 of *LNCS*, pages 152–161, 1996.

[50] Jianyang Zhou. *Calcul de plus petits produits cartésiens d'intervalles : Application au problème d'ordonnancement d'atelier*. PhD thesis, Université de la Méditerranée Aix-Marseille II, 1997.