Une Procédure de Décision pour un Problème de Satisfiabilité dans un Univers Ensembliste Hérédairement Fini*

Decision Procedure for a Satisfiability Problem in a Universe of Hereditarily Finite Sets

M. Hibti (1,2), **B. Legeard** (1) **et H. Lombardi** (2)

(1) Laboratoire d'Automatique, URA CNRS 1785, Université de Franche–Comté– ENSMM, 15 impasse des St Martin, 25000 Besançon (France).

(2) Laboratoire de Mathématiques, URA CNRS 741, Université de Franche–Comté, 25030 Besançon Cedex (France).

e-mail {hibti,lombardi}@math.univ-fcomte.fr

legeard@comte.univ-fcomte.fr

Résumé: *Nous traitons, dans cet article, du problème de satisfiabilité d'un système de contraintes ensemblistes, dans un univers d'ensembles héréditairement finis.*

Ce problème est central dans l'intégration des ensembles dans les langages de programmation, en particulier, en programmation en logique avec contraintes. Nous proposons une approche fondée sur la réduction des systèmes de contraintes ensemblistes en un système équisatisfiable d'équations et d'inéquations linéaires en entiers. La complexité de cette réduction est en $\mathcal{O}(n^3)$.

Abstract: *In this paper, we deal with the satisfiability problem for systems of constraints over hereditarily finite sets. This problem is central for integrating sets in programming languages, and particularly for constraint logic programming languages. The approach we propose here is based on reducing systems of set constraints into systems of linear integer equalities and inequalities (with bounded domain). The complexity of the reduction is on* $\mathcal{O}(n^3)$.

Mots Clés: Ensembles Héréditairement Finis, Problèmes $\mathcal{NP}$–Complets, Systèmes d'Equations et Inéquations Linéaires en Entiers, Programmation en Logique avec Contraintes.

*Une version préliminaire et courte de ce papier est apparue dans Proc. of the 4th International Conference on Logic Programming and Automated Reasonning LPAR 93, St Petersbourg (voir [17]).

Key Words: Hereditarily Finite Sets, $\mathcal{NP}$–Complete Problems, Integer Linear Programming, Constraint Logic Programming.
Classification AMS: 68Q15, 68N17, 03C15.

1 Introduction

L'automatisation de la démonstration en théorie des ensembles constitue, depuis longtemps, un thème attractif de recherche (cf. [5], [29], [6]). Mais la théorie axiomatique des ensembles usuelle offre des obstacles à la mesure de son immense pouvoir d'expression. Le théorème d'incomplétude de Gödel interdit naturellement toute procédure de décision concernant le problème général de la prouvabilité d'une formule close arbitraire. Plus précisément, des classes très simples de formules ensemblistes sont indécidables (voir par exemple [28]). De manière générale, des formules ensemblistes peuvent couramment dénoter des ensembles infinis très complexes (par exemple des parties de $\mathbb{N}$ récursivement énumérables non récursives, ou même non récursivement énumérables).

Nous nous intéressons ici à un cadre ensembliste restreint, en référence à un univers simple et bien défini, noté $\mathcal{HFSA}$, qui est formé d'ensembles héréditairement finis. Même dans ce cadre, le problème de satisfiabilité est toujours "difficile". Par exemple, le problème de satisfiabilité des propositions écrites dans le langage qui consiste en $\in$, $\vee$ et $\wedge$ comme uniques symboles (autres que les variables) est $\mathcal{NP}$–complet [14]. Dans cet article, nous étudions le problème de satisfiabilité pour des systèmes de formules ensemblistes élémentaires (en particulier sans quantificateurs). Nous montrons que le problème est $\mathcal{NP}$–complet, nous établissons des bornes explicites sur la taille d'un modèle minimal pour un tel système, et enfin nous proposons une méthode pour transformer le problème de satisfiabilité ensembliste en un problème équivalent de programmation linéaire en entiers.

Dans le cadre du projet CLPS, nous avons étudié deux méthodes complémentaires pour la satisfaction de contraintes ensemblistes. La première méthode, présentée dans [3], consiste en une mise en oeuvre incrémentale d'un test de consistance partielle assurant une consistance d'arc (cf. [24, 25]). La deuxième méthode, que nous présentons dans cet article, permet d'effectuer le test de satisfiabilité, quel que soit le système de contraintes ensemblistes considéré, que les variables soient ou non contraintes à être des éléments d'ensembles finis donnés a priori. Cette méthode consiste en une réduction du système de contraintes ensemblistes en un système équisatisfiable d'équations et d'inéquations linéaires en entiers. L'idée principale est de modèliser l'égalité et l'appartenance par des variables entières booléenes traduisant leur signification. Ensuite, il s'agit de traduire les différentes contraintes avec le maximum d'informations possibles. La signification de ces traductions et leur compatibilité seront assurées par des formules structurelles qui décrivent les propriétés de l'univers d'interprétation et donc assurent la complétude et la correction des traductions.

Les systèmes linéaires obtenus sont ensuite résolus, à l'aide d'un solveur sur les entiers. Le système de contraintes ensemblistes est déclaré insatisfiable si le système linéaire correspondant est inconsistant. Autrement, les solutions sont produites sous forme de contraintes explicitées représentant les solutions entières du système linéaire en entiers.

Cette approche est générale, et peut s'appliquer à n'importe quel système de contraintes ensemblistes établies dans des théories similaires avec ensembles héréditairement finis. La complexité de la réduction est majorée par $\mathcal{O}(n^3)$ où n est le nombre d'objets apparaissant dans les formules du système ensembliste de départ.

Notons pour terminer que notre approche réduit un problème $\mathcal{NP}$–complet (cf. [21], [30], [8]) en un problème également $\mathcal{NP}$–complet: la programation linéaire en entiers. Mais ce dernier problème est un problème classique pour lequel il existe plusieurs solveurs pratiques, souvent bien adaptés aux systèmes générés par notre réduction [35], dans lesquels toutes les inconnues sont bornées a priori. C'est ce qui à notre avis donne un réel intérêt pratique à notre travail.

Ce papier est organisé comme suit. Dans la section 2, nous allons présenter notre langage. Nous donnons aussi quelques résultats concernant l'interprétation des formules dans l'univers $\mathcal{HFSA}$. Dans la section 3, nous présentons la réduction du problème de satisfiabilité pour les formules ensemblistes au problème de programmation linéaire en entiers. La section 4 sera consacrée à l'expérimentation de l'approche à l'aide de quelques exemples. Dans la section 5, nous présentons l'intégration de notre réduction dans une approche CLP. Et finalement, la section 6 sera consacrée aux conclusions et à la présentation de travaux connexes.

2 L'Univers des ensembles héréditairement finis et un langage adapté

Dans ce travail, nous nous plaçons dans le cadre d'un univers d'ensembles héréditairement finis, construits sur une collection dénombrable d'atomes.

On considère une famille dénombrable $(a_i)_{i\in I}$ d'atomes (le mot atome doit être pris dans le sens "objet qui n'est pas un ensemble", autrement dit, "objet ne contenant aucun élément, mais différent toutefois de l'ensemble vide". En anglais, un atome est souvent appelé *urelement*). L'univers $\mathcal{HFSA}$ des ensembles héréditairement finis, construits sur la famille $(a_i)_{i\in I}$, est défini de la manière suivante:

$$\mathcal{HFSA} = \bigcup_{n\geq 0} \mathcal{HFS}(n)$$

où $\mathcal{HFS}(n)$ est défini récursivement par:

$$\mathcal{HFS}(0) = \{a_i; i \in I\} \cup \{\emptyset\}$$

et

$$\mathcal{HFS}(n) = \mathcal{HFS}(n-1) \cup P_f(\mathcal{HFS}(n-1))$$

où $P_f(X)$ dénote la classe de toutes les parties finies de X.

Deux objets de l'univers $\mathcal{HFSA}$, qui ne sont pas des atomes, sont égaux, si et seulement si ils contiennent les mêmes éléments. Chaque objet x a une profondeur n bien définie qui est le plus petit entier n tel que $x \in \mathcal{HFS}(n)$.

Exemple: L'ensemble $\{\{a, \{b\}\}, c\}$ est de profondeur 3.

Notez que l'ensemble vide est inévitable, par exemple comme intersection de deux singletons distincts, mais il a un statut d'ensemble, radicalement différent de celui des atomes.

Par abus de langage, nous appellerons parfois ensemble n'importe quel objet de $\mathcal{HFSA}$.

2.1 Le langage et les interprétations admissibles

Tout au long de cet article, nous considérerons le langage de premier ordre $\mathcal{MPLS}$ (Minimal Provisional Language for Sets) introduit dans [15].

L'alphabet consiste en:

- un ensemble dénombrable de constantes $a, b, c, \ldots$ interprétées comme des atomes deux à deux distincts de $\mathcal{HFSA}$, et la constante $\emptyset$ interprétée par l'ensemble vide,
- un ensemble de variables $x, y, z, \ldots$ représentant a priori des ensembles et occasionnellement des atomes,
- les symboles fonctionnels: $\cup$ (union), $\cap$ (intersection), $\setminus$ (différence ensembliste), et les constructeurs d'ensembles $\{., \ldots, .\}$ (il y a un constructeur de ce type pour chaque arité),
- les symboles de relation: $=$ (égalité), $\in$ (appartenance) et $\subset$ (inclusion),
- les connecteurs logiques usuels: $\Rightarrow$ (implication), $\Leftrightarrow$ (équivalence), $\wedge$ (conjonction), $\vee$ (disjonction) et $\neg$ (négation).

Les termes ensemblistes dans le langage $\mathcal{MPLS}$ sont les variables et les expressions ayant l'une des formes suivantes:

- le terme $\emptyset$,
- $\{t_1, \ldots, t_n\}$ où les t_i sont des atomes, des termes ensemblistes ou des variables,
- $t \cup t', t \cap t'$ et $t \setminus t'$ où t et t' sont des termes ensemblistes.

Les littéraux de $\mathcal{MPLS}$ sont toutes les relations atomiques construites sur les termes ensemblistes et les atomes, en utilisant les symboles de relation $=$, $\neq$, $\in$, $\notin$, $\subset$ et $\not\subset$. Les termes dans $\mathcal{MPLS}$ sont des variables, des constantes ou des termes ensemblistes. Finalement, les formules de $\mathcal{MPLS}$ sont toutes les expressions construites sur les littéraux de $\mathcal{MPLS}$ en utilisant les connecteurs logiques. On peut se passer de la négation sous la forme $\neg\varphi$ dans la mesure où on a introduit les trois prédicats $\neq$, $\notin$ et $\not\subset$. Les quantificateurs ne sont pas autorisés dans le langage, mais les variables apparaissant dans les formules sont implicitement quantifiées existentiellement lorsqu'on examine le problème de la satisfiabilité.

La présence des atomes parmi les termes du langage conduit —puisqu'on adopte l'univers $\mathcal{HFSA}$ comme domaine d'interprétation— à certaines restrictions dans l'utilisation des symboles fonctionnels et des symboles de relations dans les expressions ensemblistes. Par exemple, nous souhaiterions interdire les non-sens du style $a \cup \{b\}$ lorsque a et b sont des atomes. Plutôt que d'introduire des restrictions syntaxiques qui nous semblent difficiles à bien gérer, nous préférons introduire une restriction sur les interprétations des formules. Pour interpréter une formule de $\mathcal{MPLS}$, il faut assigner un objet de $\mathcal{HFSA}$ à chacune des variables présentes dans la formule. Si nous interprétons la formule $b \in y \cup \{b\}$ en assignant un atome à la variable y, nous obtenons un non-sens. Aussi, au lieu de considérer cette interprétation comme donnant la valeur $Vrai$ ou la valeur $Faux$, également problématiques, nous préférons ne pas la prendre en compte. Nous définissons, en conséquence, la notion d'*interprétation admissible.*

Définition 2.1 Une interprétation d'une formule φ est dite *admissible*, si la condition suivante est vérifiée: lorsqu'une variable z figure dans φ dans l'une des situations suivantes,

$z \cup t$, $t \cup z$, $t \cap z$, $z \cap t$, $t \setminus z$, $z \setminus t$, $t \subset z$, $z \subset t$, $t \not\subset z$, $z \not\subset t$, $t \in z$ et $t \notin z$,

l'objet de $\mathcal{HFSA}$ assigné à z n'est pas un atome.

Lorsqu'une interprétation d'une formule de $\mathcal{MPLS}$ est admissible, la formule prend la valeur $Vrai$ ou $Faux$ en considérant la signification usuelle dans $\mathcal{HFSA}$ des symboles fonctionnels et des symboles de relation de $\mathcal{MPLS}$. Une formule de $\mathcal{MPLS}$ est *satisfiable* si elle possède une interprétation admissible dans $\mathcal{HFSA}$ qui lui attribue la valeur $Vrai$.

Une *formule plate* est une formule dont tous les littéraux sont plats. Un littéral est plat s'il a l'une des formes suivantes (où x, y, z et les x_i désignent des variables ou des constantes):

$$\begin{array}{lll}
x \in y & x = y & x \subset y \\
x \notin y & x \neq y & x \not\subset y \\
x = y \cap z & x = y \cup z & x = y \setminus z \\
y = \{x_1, \ldots, x_k\} & &
\end{array}$$

Une *formule conjonctive plate* de $\mathcal{MPLS}$ est une conjonction de littéraux plats. Elle peut être vue comme un système de contraintes pour des objets de $\mathcal{HFSA}$.

Par $\mathcal{MPLS}$–SAT nous notons le problème de satisfiabilité des formules conjonctives plates de $\mathcal{MPLS}$ dans l'univers $\mathcal{HFSA}$ (seules sont considérées les interprétations admissibles).

Il est possible de transformer toute formule d'un langage de premier ordre en une formule plate équivalente en un temps presque linéaire, plus précisément en $\mathcal{O}(n\ log(n))$ (voir [16]).

Nous nous intéressons maintenant au problème de satisfiabilité des formules conjonctives plates.

2.2 Formules plates satisfiables dans $\mathcal{HFSA}$

Une interprétation d'une formule de $\mathcal{MPLS}$ est une application qui transforme les variables de la formule en des objets de $\mathcal{HFSA}$ (les atomes du langage étant interprétés comme des atomes de même nom dans $\mathcal{HFSA}$ et $\emptyset$ est interprété par l'ensemble vide). Une interprétation admissible M est appelée un modèle pour une formule conjonctive plate φ si les littéraux de cette dernière sont interprétés vrais via M.

Dans cette section, nous discutons certaines propriétés caractérisant les modèles dans $\mathcal{HFSA}$. Nous rappelons tout d'abord la représentation dite du *modèle–graphe* [27]. Cette technique consiste en la description d'une famille finie d'objets de $\mathcal{HFSA}$ à travers un graphe acyclique orienté. Les feuilles du graphe sont interprétées comme des atomes ou comme l'ensemble vide. Les flèches symbolisent la relation d'appartenance. Un noeud qui n'est pas une feuille représente un objet de $\mathcal{HFSA}$ de profondeur supérieure ou égale à 1. Néanmoins, l'extensionalité n'est pas imposée dans le graphe et deux noeuds distincts peuvent représenter le même objet. La famille définie par le graphe est parfaitement connue dès qu'on a assigné à chaque feuille un objet de $\mathcal{HFSA}$ (en pratique, ce sera toujours un atome ou l'ensemble vide): puisque le graphe est acyclique, on peut associer, reccursivement selon la profondeur, à chaque noeud qui n'est pas une feuille, l'objet de $\mathcal{HFSA}$ qui est l'ensemble des objets assignés à ses fils.

Par exemple, si $S = \{a, \{a, b\}, \{\{c, \{d\}\}\}\}$, on considère:

$u_1 = \{a, b\}$,
$u_2 = \{\{c, \{d\}\}\}$,
$u_3 = \{c, \{d\}\}$ et
$u_4 = \{d\}$

i.e. $u_1 = \{a, b\}, u_2 = \{u_3\}, u_3 = \{c, u_4\}$ et $u_4 = \{d\}$ et on associe à S le graphe $G = (V, E)$ avec:

$$V = \{S, u_1, u_2, u_3, u_4, a, b, c, d\}$$

et

$$E = \{(a, S), (u_1, S), (u_2, S), (a, u_1), (b, u_1), (u_3, u_2), (c, u_3), (u_4, u_3), (d, u_4)\}$$

Le graphe G associé à S

La stratégie consiste à fabriquer à partir d'un modèle arbitraire de φ, un modèle–graphe réduit de φ, en supprimant la plupart des informations redondantes ou superflues. Nous utilisons cette technique pour construire des modèles ensemblistes qui servent à établir les preuves d'équisatisfiabilité des formules ensemblistes avec les systèmes linéaires produits par la réduction.

Le théorème suivant montre que le graphe obtenu selon la technique proposée est de taille polynomiale en celle de la formule.

Notations et convention : On note $(x_i)_{i=1,\ldots,n}$ les variables de la formule conjonctive plate φ, et $(x_i)_{n<i\leq n+m}$ ses constantes. Par convention x_{n+1} est toujours prise égale à $\emptyset$ même si $\emptyset$ ne figure pas explicitement dans la formule, de sorte que $\emptyset$ est toujours considéré comme une constante de la formule. Dans la suite on prendra $r = n + m$.

Théorème 2.2 *Soit φ une formule conjonctive plate satisfiable de $\mathcal{MPLS}$, contenant r atomes ou variables. Alors il existe un modèle–graphe de taille polynomiale satisfaisant φ, avec les caractéristiques suivantes:*

- *le nombre de noeuds est majoré par $r + r^2$,*
- *les noeuds représentent tous, ou bien des atomes de la formule, ou bien des objets de $\mathcal{HFSA}$ correspondant aux variables de la formule, ou bien de nouveaux atomes, éléments des objets précédents.*

Preuve: Ceci est une adaptation de la preuve donnée dans [26]. Nous remplaçons seulement les ensembles de base (qui sont supposés être dans [26] de profondeur strictement positive) par des atomes. Nous rappelons que, dans notre contexte, les atomes peuvent être utilisés dans les formules, et qu'ils doivent rester invariants par les interprétations.

Soit φ une formule conjonctive plate satisfiable de $\mathcal{MPLS}$, $(x_i)_{i\in I}$ ses variables et ses constantes (y compris, rappelons le, $\emptyset$), et soit M un modèle quelconque de φ dans $\mathcal{HFSA}$. A partir de M, on construit un modèle–graphe concis $G = (V, E)$ pour φ, de la manière suivante. Tout d'abord, on définit

$$V_1 = \{x_i; i \in I\} \cup \{u_{i,j}; (i,j) \in I^2, Mx_i \backslash Mx_j \textit{ défini et non vide}\}$$

Ensuite, l'ensemble V des noeuds du graphe est défini comme un ensemble quotient de V_1: si $Mx_i = Mx_k$, on identifie x_i et x_k, et si $Mx_i = Mx_k$, $Mx_j = Mx_l$ et $Mx_i \setminus Mx_j$

défini et non vide, alors on identifie $u_{i,j}$ et $u_{k,l}$. Notons bien que V n'a pas plus de $r + r^2$ éléments.
Ensuite, à chaque noeud de V, on associe un objet de $\mathcal{HFSA}$ de la manière suivante: à un noeud x_i est associé l'objet Mx_i, et à chaque noeud $u_{i,j}$ (s'il est défini) est associé un objet $U_{i,j}$, choisi parmi les éléments de $Mx_i \backslash Mx_j$. En particulier, si $u_{i,j} = u_{k,l}$ alors $U_{i,j} = U_{k,l}$.
L'ensemble E des flèches est alors défini par

$$E = \{(u, w) \in V^2; U \in W\}$$

où U et W sont respectivement les objets de $\mathcal{HFSA}$ associés à u et w. Il est clair que le graphe obtenu est acyclique.

Si u est une feuille de G, on définit

- $a_u = \emptyset$ lorsque $U = \emptyset$,
- et dans les autres cas, les a_u sont des atomes avec: $a_u = a_w \Longleftrightarrow U = W$.

Enfin, à partir du modèle–graphe (V, E) on définit un modèle de φ au moyen de la fonction M^* répondant aux conditions suivantes:

$$M^*u = \begin{cases} a_u & \text{si u est une feuille de G} \\ \{M^*w; (w, u) \in E\} & \text{sinon} \end{cases}$$

M^* est définie reccursivement, selon un ordre de profondeur non décroissante des noeuds, en commençant par les feuilles. On remarque que, lorsque Mx_i est un ensemble non vide, M^*x_i est également non vide et lorsque $Mx_i = \emptyset$, alors $M^*x_i = \emptyset$. Cela implique que l'interprétation M^* reste admissible.
En fait, M^* définit un modèle plus concis que M et dont la taille est polynomialement bornée en celle de φ (voir les détails qui justifient que M^* est bien un modèle dans [14]). $\square$

La construction précédente nous donne les résultats plus précis suivants:

Proposition 2.3 *Soit φ une formule conjonctive plate satisfiable de $\mathcal{MPLS}$ et p le nombre de toutes les occurrences du symbole fonctionnel $\{., \ldots, .\}$ et du symbole de relation $\in$. Alors il existe un modèle pour φ dans $\mathcal{HFSA}$, qui associe à chaque variable de la formule un ensemble de profondeur inférieure ou égale à $p + 1$. (Dans ce qui suit, cette valeur $p + 1$ sera notée α.)*

Remarque 2.4 Notons bien que cette borne ne peut pas être obtenue sans l'introduction des atomes.

Proposition 2.5 *Soit φ une formule conjonctive plate satisfiable de $\mathcal{MPLS}$ possédant n variables et m constantes. Alors il existe un modèle pour φ dans $\mathcal{HFSA}$, qui associe à chaque variable de la formule un ensemble de cardinalité inférieure ou égale à $2n + m$.*

3 Réduction à un problème de programmation linéaire en entiers

Notre objectif dans cette approche est de transformer les relations "ensemblistes" apparaissant dans une formule conjonctive plate de $\mathcal{MPLS}$ en un système d'équations et d'inéquations linéaires en entiers. Ce dernier doit être équisatisfiable avec la formule de départ. Dans un premier temps, nous étudierons le fragment des formules conjonctives plates positives, qui admettent des modèles pouvant être construits sur les seuls "objets" —variables et constantes— de la formule, puis nous traiterons les formules conjonctives plates générales. Rappelons la convention importante selon laquelle $\emptyset$ est toujours considéré comme faisant partie des constantes de n'importe quelle formule.

3.1 Préliminaires: écriture de certaines fonctions (sup et inf) dans le cadre linéaire

Dans la suite, certains systèmes linéaires en entiers font apparaître des fonctions qui ne sont pas nécessairement convexes (e.g. la fonction sup (resp. inf) pour exprimer le maximum (resp. le minimum) d'une séquence d'entiers ainsi que la fonction $\mid . \mid$ pour exprimer la valeur absolue d'un entier). L'introduction de ces fonctions ne met cependant pas en cause le caractère linéaire de nos systèmes. En effet, les entiers sur lesquels portent ces fonctions sont soit booléens, soit pseudo–booléens. Or, pour de tels entiers, on a le lemme suivant.

Lemme 3.6 *1) Soient $x_1, \ldots, x_k$ des entiers dans $\{0,1\}$ et y un entier, alors:*

$$sup(x_1, \ldots, x_k) = y \Longleftrightarrow \begin{cases} y \leq 1 \\ x_1 + \ldots + x_k \geq y \\ \bigwedge\limits_{i=1,\ldots,k} x_i \leq y \end{cases}$$

et de la même façon, puisqu'on a l'équivalence générale:

$$y = inf(x_1, \ldots, x_k) \Longleftrightarrow 1 - y = sup(1 - x_1, \ldots, 1 - x_k)$$

$$inf(x_1, \ldots, x_k) = y \Longleftrightarrow \begin{cases} y \geq 0 \\ x_1 + \ldots + x_k \leq y + k - 1 \\ \bigwedge\limits_{i=1,\ldots,k} x_i \geq y \end{cases}$$

2) Soit x un entier dans $\{-1, 0, 1\}$ et y un entier, alors:

$$y = sup(x, 0) \Longleftrightarrow \begin{cases} y \geq x \\ y \geq 0 \\ y \leq 1 \\ 2 \,.\, y \leq x + 1 \end{cases}$$

Ensuite la valeur absolue est définie au moyen de:

$$| x | = sup(x, 0) + sup(-x, 0)$$

3.2 Formules conjonctives plates positives. Modèles minimaux

Le fragment que nous étudions ici est formé des formules conjonctives plates utilisant les seuls littéraux suivants:

$$\begin{array}{lll} x \in y & x \subset y & x = y \\ x = y \cup z & & \\ y = \{x_1, \ldots, x_k\} & & \end{array}$$

où x, y, z et les x_i désignent des variables ou des constantes.
Dans la suite ces littéraux seront appelés littéraux positifs plats.

Notons que, bien qu'il n'y ait que des littéraux positifs, la formule peut contenir certaines informations négatives. Par exemple, si a et b sont deux constantes distinctes, alors $M_a \neq M_b$, pour toute interprétation admissible M.

Théorème 3.7 *Soit φ une formule conjonctive plate positive satisfiable, alors φ admet un modèle–graphe dont chaque noeud correspond à une variable ou une constante de la formule.*

Preuve. Soit M un modèle construit selon la stratégie du théorème 2.2, et obtenu à partir d'un graphe acyclique $G = (V, E)$ avec, pour chaque feuille u de G, un objet a_u de $\mathcal{HFSA}$ (un atome ou bien l'ensemble vide). Soit F l'ensemble des feuilles de G et soit F' la partie de F restreinte aux variables et constantes de la formule (y compris $\emptyset$). Supprimons dans le graphe toutes les flèches ayant pour source un élément de $F \setminus F'$. On obtient un nouveau graphe G' dont les noeuds sont les éléments de $V \setminus (F \setminus F')$ Certains noeuds de profondeur 1 du graphe G peuvent devenir des feuilles dans G'. Dans ce cas on pose $a_u = \emptyset$.
Ceci définit une nouvelle famille finie d'objets de $\mathcal{HFSA}$, associée au nouveau graphe G'. On vérifie sans peine que cela fournit une interprétation admissible et un modèle pour la formule φ. En outre, chaque noeud de G' correspond maintenant à une variable ou une constante de la formule. $\square$

3.3 Formules conjonctives plates positives. Réduction

Nous développons maintenant la réduction d'une formule conjonctive plate positive en un système linéaire en entiers.

On introduit une famille d'entiers $P_{i,j}$, $Q_{i,j}$, C_i et H_i qui seront les inconnues du système d'inéquations linéaires en entiers, de la manière suivante:

• Pour chaque $i, j = 1, \ldots, n+m$, une variable $P_{i,j} \in \{0,1\}$ est introduite et sera interprétée par:

$$P_{i,j} = 1 \Leftrightarrow x_i \in x_j \qquad (Appartenance)$$

• Pour chaque $i, j = 1, \ldots, n+m$ avec $i < j$, une variable $Q_{i,j} \in \{0,1\}$ est introduite et sera interprétée par:

$$Q_{i,j} = 1 \Leftrightarrow x_i = x_j \qquad (Egalité)$$

• Pour chaque $i = 1, \ldots, n+m$, on introduit $C_i \in \mathbb{N}$ pour représenter la cardinalité de x_i ($C_i = 0$ si x_i est interprétée par un atome a, en particulier si $i > n$).

• Pour chaque $i = 1, \ldots, n+m$, on introduit $H_i \in \mathbb{N}$ pour représenter la profondeur de x_i ($H_i = 0$ si x_i est interprétée par un atome a, en particulier si $i > n$).

Remarque 3.8 Nous signalons que pour chaque i, $P_{i,i}$ est seulement une notation pour 0, $Q_{i,i}$ est une notation pour 1. De manière similaire, si $j \geq n+1$, alors H_j, C_j et $P_{i,j}$ sont des notations pour 0. Si $k > k' \geq n+1$, alors $Q_{k,k'}$ est aussi une notation pour 0. Finalement, si $i < j$ alors $Q_{i,j}$ est une notation pour $Q_{j,i}$. Par conséquent, les seules variables seront les $P_{i,j}$ pour $i \neq j$ et $j \leq n$, les $Q_{i,j}$ pour $i < j$ et $i \leq n$, les C_i et H_i pour $i \leq n$.

Dans la section précédente, nous avons noté α la majoration de profondeur directement lisible sur la formule. Par ailleurs, nous notons $\beta = n + m - 1$ la majoration de cardinalité qui résulte du théorème 3.7. Ceci nous permet de considérer un domaine borné pour les entiers correspondants. Nous définissons les domaines à l'aide des formules suivantes:

$$(\bigwedge_{i<j} P_{i,j} + P_{j,i} \leq 1) \ \wedge \ (\bigwedge_{i \neq j} P_{i,j} \geq 0) \qquad (S1_a)$$

$$\bigwedge_{i<j} 0 \leq Q_{i,j} \leq 1 \qquad (S1_b)$$

$$\bigwedge_{i} 0 \leq C_i \leq \beta \qquad (S1_c)$$

$$\bigwedge_{i} 0 \leq H_i \leq \alpha \qquad (S1_d)$$

Ensuite, nous exprimons le fait que $Q_{i,j} = 1$ signifie l'égalité, que l'égalité est une relation déquivalence, deux objets égaux appartiennent aux mêmes ensembles et ont les mêmes éléments, deux objets égaux ont la même profondeur et la même cardinalité :

$$\bigwedge_{k} \bigwedge_{i \neq j} 1 - Q_{i,j} \geq P_{i,k} - P_{j,k} \qquad (S2_a)$$

$$\bigwedge_{k} \bigwedge_{i \neq j} 1 - Q_{i,j} \geq P_{k,i} - P_{k,j} \qquad (S2_b)$$

$$\bigwedge_{(i \neq j),(i \neq k),(j \neq k)} 1 - Q_{i,j} \geq Q_{i,k} - Q_{j,k} \qquad (S2_c)$$

$$\bigwedge_{(i \neq j),(i,j \leq n)} \alpha(1 - Q_{i,j}) \geq H_i - H_j \qquad (S2_d)$$

$$\bigwedge_{(i \neq j),(i,j \leq n)} \beta(1 - Q_{i,j}) \geq C_i - C_j \qquad (S2_e)$$

Ces inégalités structurelles ont les significations suivantes:

La formule $(S2_a)$ (resp $(S2_b)$) exprime le fait que deux objets égaux doivent appartenir aux mêmes ensembles (resp. avoir les mêmes éléments).

La formule $(S2_c)$ exprime la transitivité de l'égalité.

Les formules $(S2_d)$ et $(S2_e)$ expriment le fait que deux objets égaux doivent avoir les mêmes cardinalités et les mêmes profondeurs.

Maintenant, nous exprimons le fait que deux objets de profondeur supérieure ou égale à 1 sont égaux s'ils ont les mêmes éléments (*extensionalité*).

$$\bigwedge_{(i,j \leq n),(i \neq j)} 1 + Q_{i,j} + \sum_k \mid P_{k,j} - P_{k,i} \mid \geq (D_i + D_j) \qquad (S3)$$

où pour chaque $j \leq n$, D_j est défini par

$$(\alpha\ D_j \geq H_j \geq D_j) \ \wedge \ (D_j \leq 1)$$

de sorte que $D_j = 1$ signifie que x_j est de profondeur supérieure ou égale à 1, et où les termes $\mid P_{k,j} - P_{k,i} \mid$ sont exprimés par l'intermédiaire de nouvelles variables entières:

$$R_{k,i,j} = sup(0, P_{k,i} - P_{k,j})$$

Finalement, nous exprimons que la relation $x_i \in x_j$ implique $H_i + 1 \leq H_j$:

$$\bigwedge_{i \neq j} (\alpha + 1)(1 - P_{i,j}) \geq H_i - H_j + 1 \qquad (S4)$$

Les contraintes $S4$ sont cruciales car ce sont elles qui garantissent l'acyclicité du modèle–graphe qui sera construit.

Après ces inégalités structurelles, nous devons traduire les formules conjonctives plates positives en traduisant chacun de leurs littéraux. D'abord, nous remarquons que, sans aucune perte de généralité, nous pouvons supposer qu'aucune relation $x_i = x_j$ n'apparaît dans la conjonction φ (sinon elle peut être simplifiée), et donc on n'a pas à la traduire. Les autres traductions sont faciles:

- Le littéral $x_i \in x_j$ sera traduit par:

$$\begin{cases} P_{i,j} = 1 \\ H_i + 1 \leq H_j \\ C_j \geq 1 \end{cases} \qquad (T_1)$$

- On traduit $x_i \subset x_j$ par :

$$\begin{cases} C_i \leq C_j \\ H_i \leq H_j \\ ((H_i + Q_{i,n+1}) \geq 1) \;\wedge\; ((H_j + Q_{j,n+1}) \geq 1) \\ \bigwedge\limits_{1 \leq k \leq n+m} (P_{k,i} \leq P_{k,j}) \end{cases} \tag{T_2}$$

- On traduit $x_i = x_j \cup x_k$ par:

$$\begin{cases} C_i \leq C_j + C_k \\ (C_i \geq C_j) \wedge (C_i \geq C_k) \\ (H_i \geq H_j) \wedge (H_i \geq H_k) \\ ((H_i + Q_{i,n+1}) \geq 1) \;\wedge\; ((H_j + Q_{j,n+1}) \geq 1) \;\wedge\; ((H_k + Q_{k,n+1}) \geq 1) \\ \bigwedge\limits_{1 \leq s \leq n+m} P_{s,i} = sup(P_{s,j}, P_{s,k}) \end{cases} \tag{T_3}$$

- On traduit $\{x_{i_1}, \ldots, x_{i_t}\} = x_k$ par :

$$\begin{cases} 1 \leq C_k \leq t \\ \bigwedge\limits_{1 \leq s \leq t} P_{i_s,k} = 1 \\ \bigwedge\limits_{1 \leq s \leq t} H_k \geq H_{i_s} + 1 \\ \bigwedge\limits_{1 \leq j \leq n+m} P_{j,k} = sup_{s \leq t}(Q_{j,i_s}) \end{cases} \tag{T_4}$$

Chaque inégalité du type $((H_i + Q_{i,n+1}) \geq 1)$ est écrite de manière à interdire que la variable x_i puisse être interprétée comme un atome, ce qui garantira l'admissibilité de l'interprétation obtenue.

Les contraintes (T_i) présentent certaines redondances par rapport aux contraintes structurelles, par exemple dans $(T1)$ on écrit $P_{i,j} = 1$ puis $H_i + 1 \leq H_j$ alors que la deuxième contrainte résulte de la première par $(S4)$. La raison pour laquelle nous introduisons ces redondances est que lors de la résolution du système de contraintes, les (T_i) sont a priori moins encombrantes et sont mises en tête du système ce qui peut aider à detecter rapidement une eventuelle inconsistance.

Le système S_φ est obtenu en joignant les inégalités structurelles et les traductions des littéraux de φ. Nous mentionnons que dans l'étape résolution, les contraintes sont posées suivant l'ordre: *Déclarations de domaines* $(S1_a - S1_d)$, *Traductions de littéraux* $(T1 - T4)$ *et Inégalités structurelles* $(S2 - S4)$. Le théorème suivant montre que le système S_φ est équisatisfiable avec φ.

Théorème 3.9 *Une formule conjonctive positive plate φ est satisfiable dans le modèle $\mathcal{HFSA}$, si et seulement si le système correspondant S_φ décrit ci–dessus admet des solutions entières. La réduction de φ à S_φ est de complexité polynomiale. En particulier, la taille de S_φ est en $\mathcal{O}(Nr) + \mathcal{O}(r^3)$, où $r = n + m$ est le nombre de variables et de constantes apparaissant dans φ et N est le nombre de littéraux de φ.*

Preuve: D'abord, dans cette réduction, chaque littéral a une traduction T_i dont la taille est en $\mathcal{O}(r)$. La traduction des littéraux de φ est donc en $\mathcal{O}(Nr)$. Les formules structurelles $(S1_i)$ et $(S4)$ sont en $\mathcal{O}(r^2)$, les formules $(S2_i)$ sont en $\mathcal{O}(r^3)$. Quant à $(S3)$ elle est formée de r^2-1 inéquations de taille en $\mathcal{O}(r)$. Il s'ensuit que la taille de S_φ est en $\mathcal{O}(Nr)+\mathcal{O}(r^3)$ ou encore en $\mathcal{O}(Nr^2)$.
Noter que si si on tient compte de l'espace occupé par l'écriture des variables, il faut rajouter un facteur $log(r)$ à la majoration donnée dans le théorème.

Maintenant, supposons φ satisfiable. D'après le théorème 3.7, il existe un modèle M' de φ dont les seuls objets "représentent" des variables ou des constantes de φ. Pour obtenir une solution de S_φ, il suffit alors de poser

- $C_i = \#(M'x_i)$,
- $H_i = la\ profondeur\ de\ M'x_i$,
- $P_{i,j} = Val(M'x_i \in M'x_j)$,
  ($Val(M'x_i \in M'x_j)$ étant la valeur de vérité de $(M'x_i \in M'x_j)$.)
- $Q_{i,j} = Val(M'x_i = M'x_j)$.

Puisque M' est un modèle (dans $\mathcal{HSFA}$), les formules structurelles et les traductions de littéraux sont manifestement vérifiées.

Pour la réciproque, supposons que S_φ admet des solutions. Alors un modèle M pour φ peut être obtenu à travers les étapes suivantes:
On commence par "fusionner" les objets "égaux" en un seul objet. Pour cela nous introduisons une relation d'équivalence $\equiv$ sur les variables ou constantes x_i apparaissant dans φ par:

$$x_i \equiv x_j \Longleftrightarrow Q_{i,j} = 1$$

Lemme 3.10 *La relation $\equiv$ est une relation d'équivalence. De plus, pour chaque $i,j=1,\ldots,r$, $x_i \equiv x_j$ entraine:*
i) $H_i = H_j$,
ii) $C_i = C_j$,
iii) $P_{k,i} = P_{k,j}, P_{i,k} = P_{j,k}$ et $Q_{k,i} = Q_{k,j}$ pour chaque $k = 1,\ldots,r$.

Preuve: La réflexivité et la symétrie sont garanties par les conventions concernant les $Q_{i,j}$ données dans la remarque 3.8.
La transitivité ainsi que les propriétés $(i-iii)$ sont garanties par les formules structurelles du système. □

Considèrons ensuite $y_1,\ldots,y_m$ comme étant les représentants de chaque classe d'équivalence de $\equiv$.

On définit un graphe $G=(V,E)$ qui rend compte des relations exprimées dans le système S_φ, et qui servira de support pour un modèle de φ:

$$V = \{y_1,\ldots,y_m\}\ et\ E = \{(y_i,y_j) \in V^2; P_{i,j} = 1\}$$

Vu les contraintes ($S4$) portant sur les H_i, $P_{i,j} = 1$ implique $H_j \geq H_i + 1$ et ceci interdit tout cycle dans le graphe. Pour la même raison, comme $0 \leq H_i \leq \alpha$ est une contrainte dans S_φ la profondeur du modèle graphe G est inférieure ou égale à α.

On définit sur V une application $y \mapsto M_y$ en suivant un ordre non décroissant des H_{y_i} comme suit:

$$My_i = \begin{cases} \{My_j; (y_j, y_i) \in E\} & \text{si ce dernier est non vide} \\ x_{n+j} & \text{si } Q_{i,n+j} = 1 \text{ avec } j \geq 1 \\ a_{y_i} & \text{sinon} \end{cases}$$

(où les a_{y_i} sont de nouveaux atomes)
L'application M est bien définie puisque le graphe est acyclique. Enfin, si y est le représentant de la classe de x on pose $Mx = My$.

Maintenant, nous allons voir que M est un modèle pour φ. Tout d'abord, l'interprétation est admissible en raison des contraintes imposées sur les profondeurs dans les traductions (T_i). Comme φ est une conjonction de littéraux positifs plats, et que sans perte de généralité aucun littéral de φ n'est une égalité du type $x_i = x_j$, il suffit de vérifier le lemme suivant:

Lemme 3.11 *L'application M satisfait les conditions suivantes (qui doivent être lues comme suit: une implication $A \Longrightarrow B$ dans ce lemme signifie que si le littéral A apparaît dans φ alors on a la relation B pour l'interprétation M).*
i) $x_i \in x_j \Longrightarrow Mx_i \in Mx_j$
ii) $x_i \subset x_j \Longrightarrow Mx_i \subset Mx_j$
iii) $x_i = x_j \cup x_k \Longrightarrow Mx_i = Mx_j \cup Mx_k$
iv) $\{x_{i_1}, \ldots, x_{i_r}\} = x_k \Longrightarrow \{Mx_{i_1}, \ldots, Mx_{i_r}\} = Mx_k$

Pour prouver ce lemme, on a besoin du lemme suivant:

Lemme 3.12 *Pour chaque $i, j = 1, \ldots, n+m$, on a:*
a) $Mx_i = Mx_j \Longleftrightarrow Q_{i,j} = 1$
b) $Mx_i \in Mx_j \Longleftrightarrow P_{i,j} = 1$

Preuve du lemme 3.12: *a*) L'implication $Q_{i,j} = 1 \Longrightarrow Mx_i = Mx_j$ est assurée par la définition de M constante sur les classes d'équivalence. La réciproque est garantie par la définition de M pour les objets de profondeur 0, et par les inégalités ($S3$) pour les objets de profondeur ≥ 1.
b) L'implication $P_{i,j} = 1 \Longrightarrow Mx_i \in Mx_j$ est vraie par définition des My. Réciproquement, supposons $Mx_i \in Mx_j$. Alors, il existe x_k tel que $Mx_k = Mx_j$ et $P_{i,k} = 1$. Mais, comme $Mx_k = Mx_j$ il s'en suit que $Q_{k,j} = 1$ (par *a*)). On obtient $P_{i,j} = 1$ par la formule ($S2_a$) . □

Preuve du lemme 3.11: Les conditions ($i - ii$) sont évidentes (par défintion de M).

iii) Supposons que $x_i = x_j \cup x_k$ soit un littéral de φ. Supposons en outre $Mx_s \in Mx_i$, alors $P_{s,i} = 1$. Comme $x_i = x_j \cup x_k$ est un littéral de φ, il existe dans S_φ une équation $P_{s,i} = sup(P_{s,j}; P_{s,k})$. Il s'en suit que $P_{s,j} = 1$ ou $P_{s,k} = 1$ et donc, par le lemme 3.12, $Mx_s \in Mx_j \cup Mx_k$.
Les mêmes arguments sont valables pour (iv). □

Ceci achève la preuve du théorème. □

Remarque 3.13

1) Les entiers H_i et C_i dans une solution du système S_φ ne représentent pas, a priori, les profondeurs et les cardinalités dans le modèle produit par la solution du système, mais seulement des majorations. Ceci n'est pas surprenant puisqu'une formule de $\mathcal{MPLS}$ peut être satisfaite par deux modèles qui assignent différentes valeurs à la profondeur d'une variable.

2) Toutes les contraintes sur les cardinalités peuvent être omises sans remettre en cause le théorème 3.9. Cependant nous estimons utile l'introduction de telles contraintes car elles peuvent rendre le système incompatible en rationnels. Ce qui est beaucoup plus facilement détectable que l'inconsistance en entiers.

3) Signalons également qu'on pourra introduire des contraintes supplémentaires sur les cardinalités des objets dans une formule (i.e. dans les formules, on peut tolérer des littéraux de la forme $f(C_1, \ldots, C_k) \leq N$ où f est une forme linéaire à coefficients entiers et N un entier naturel). C'est par exemple le cas avec le problème des pigeonniers (voir section 4). Cependant, dans ce cas, pour garantir la complétude et la correction de la traduction des systèmes ainsi étendus, il faudra introduire une nouvelle formule structurelle S_5 garantissant que les C_i sont exactement les cardinalités des objets de la formule.

$$\bigwedge_{i \leq n} C_i = \sum_k (P_{k,i} - R_{k,i}) \qquad (S5)$$

où

$$R_{k,i} = \inf(P_{k,i}, \sup_{j<k}(Q_{k,j})).$$

Pour chaque i, la formule structurelle concernant la cardinalité C_i est écrite en comptant les éléments de l'ensemble x_i de la manière suivante:

• Si $x_1 \in x_i$ ($P_{1,i} = 1$) on compte un élément, ($C_i = P_{1,i} + ...$), sinon on ne compte rien.

• Si $x_2 \in x_i$, alors on compte deux éléments si $x_1 \neq x_2$ ($Q_{1,2} = 0$) on écrit donc:

$$(C_i = P_{1,i} + P_{2,i} - inf(Q_{1,2}\ ,\ P_{2,i}) + ...)$$

le terme $\inf(Q_{1,2}, P_{2,i})$ est introduit pour ne compter qu'une seule fois x_1 si $x_1 = x_2$, et pour ne rien supprimer si jamais $x_1 = x_2$ et aucun d'eux n'est dans x_i.
On continue ainsi ce raisonnement pour les autres éléments et cette fois–ci on compare chaque nouvel élément à ceux qui l'ont précédé. De cette manière, $S5$ peut s'écrire:

$$
\begin{array}{rlll}
C_i = & P_{1,i} & + & \\
& P_{2,i} & - & \inf(P_{2,i}\ ,\ Q_{1,2}) \\
& P_{3,i} & - & \inf(P_{3,i}\ ,\ \sup(Q_{1,3}, Q_{2,3})) \\
& \vdots & & \\
& P_{N,i} & - & \inf(P_{N,i}\ ,\ \sup_{j<N}(Q_{j,N}))
\end{array}
$$

3.4 Formules conjonctives plates générales. Modèles minimaux

Dans un cadre plus général, il est souhaitable de traiter des formules présentant des informations négatives plus explicites. Ceci nous amène à considérer le problème de satisfiabilité d'une formule conjonctive plate générale.

L'introduction de la négation repose le problème des objets servant pour l'interprétation des formules. Alors que pour le fragment positif, les formules peuvent être interprétées via un modèle où ne figurent que les Mx_i, ce n'est plus le cas pour les formules conjonctives plates générales. Considérons, par exemple, les formules

$$\varphi \equiv (\emptyset \in x) \wedge (\emptyset \in y) \wedge (x \neq y) \wedge (x \notin y) \wedge (y \notin x)$$

$$
\begin{array}{c}
\psi \equiv (x = \{u\}) \wedge\ (y = \{v\}) \wedge\ (z = \{u, v\}) \wedge\ (t = \{x, y\}) \wedge (\emptyset \in u) \wedge\ (\emptyset \in v) \\
(x\ \cap\ y = \emptyset) \wedge\ (u\ \cap\ z = \emptyset) \wedge\ (v\ \cap\ z = \emptyset) \wedge\ (v\ \cap\ t = \emptyset) \wedge\ (u\ \cap\ t = \emptyset)
\end{array}
$$

Il apparait que φ est satisfiable dans $\mathcal{HFSA}$, cependant elle n'admet pas de modèle construit seulement sur ses variables et constantes. En effet, dans φ l'interprétation de x doit être différente de celle de y, par exemple, en certifiant que x contient un élément distinct de x, y et $\emptyset$.

Quant à ψ, vu $x \cap y = \emptyset$, un modèle de ψ doit satisfaire $M_u \neq M_v$, cette inégalité nécessite un nouvel élement dans u ou dans v. Mais, aucun des Mx_i pour $x_i = u, v, x, y, z, t$ ne peut convenir (voir le dessin et prendre en compte les intersections vides).

Un "réservoir" d'objets supplémentaires est donc nécessaire pour construire un modèle pour une formule "négative", et il est intéressant de minimiser la taille de ce réservoir pour en tenir compte pendant la réduction, et rendre cette réduction moins coûteuse.
Connaissant les objets de la formule en question on peut toujours prévoir une majoration du nombre d'objets supplémentaires nécessaires. Une première estimation est obtenue par la stratégie du modèle–graphe et a été donnée dans le théorème 2.2.

Ce résultat peut être amélioré:

Théorème 3.14 *Soit φ une formule conjonctive plate de $\mathcal{MPLS}$ avec n variables et k occurrences du symbole $\not\subseteq$. Si φ est satisfiable, alors elle admet un modèle construit sur au plus $k+n$ atomes supplémentaires de $\mathcal{HFSA}$. En outre, chaque Mx_i ou bien est de profondeur 0, ou bien admet pour seuls éléments des atomes et des Mx_j.*

Preuve: Soit φ une formule satisfiable de $\mathcal{MPLS}$ et $(x_i)_{i\in I}$ ses variables et constantes. Le but de ce théorème est de montrer que, partant d'un modèle de φ, on peut en supprimer certaines informations sans causer aucune perturbation aux relations présentes dans la formule. D'autre part, on ne garde que des atomes, en petit nombre, comme informations supplémentaires.

Le point réellement important est que, pour préserver la vérité des littéraux de φ, lorsqu'on supprime les objets "superflus", il faut essentiellement préserver les relations de différence $Mx_i \neq Mx_j$. Le lemme suivant montre qu'on n'a pas besoin de plus de n atomes supplémentaires pour préserver les relations de différence.

Lemme 3.15 *Soit φ une formule conjonctive plate de $\mathcal{MPLS}$, M un modèle pour φ ayant les propriétés du modèle construit au théorème 2.2, et $(a_i)_{i\in I}$ la famille d'atomes supplémentaires nécessaires pour M, et soit n le nombre de variables. Alors, il existe un sous ensemble J de I tel que:*

1) $\mid J \mid\leq n$, et

2) Pour chaque paire (x,y) de variables apparaissant dans la formule, $Mx \neq My$ implique qu'il existe $i \in J$ tel que

$$a_i \in (Mx\backslash My) \cup (My\backslash Mx)$$

ou z apparaissant dans φ tel que

$$Mz \in (Mx\backslash My) \cup (My\backslash Mx).$$

Preuve. Soit $(x_i)_{i\leq n}$ la famille des variables apparaissant dans la formule. On peut supposer sans perte de généralité que $Mx_i \neq Mx_j$ pour $i \neq j$. Nous allons procéder par induction sur n. Pour $n=1$ et $n=2$, le lemme est évident.

Soit $(x_i)_{i\leq k}$ la famille de variables (pour le pas k), soit J_k un ensemble tel que 1) et 2) soient vérifiées pour la formule "restreinte" aux variables $x_1,\ldots,x_k$. Posons

$$A_{J_k} = \{a_h; h \in J_k\} \cup \{Mx_j; j \leq n\}$$

on a $Mx_{k+1} \neq Mx_j$ pour chaque $j \leq k$. S'il n'existe aucun indice j tel que

$$Mx_{k+1} \cap A_{J_k} = Mx_j \cap A_{J_k} \qquad (*)$$

alors J_k est l'ensemble désiré pour $(x_i)_{i \leq k+1}$. Autrement (si un tel indice existe), alors par l'hypothèse de récurrence, il est facile de voir qu'il est unique. Alors il suffit de choisir un atome a_r ou une variable x_r dans $Mx_{k+1} \backslash (Mx_j \cup A_{J_k})$ ou dans $Mx_j \backslash (Mx_{k+1} \cup A_{J_k})$, et de poser

$$J_{k+1} = \begin{cases} J_k \cup \{r\} & \text{si c'est } a_r \\ J_k & \text{si c'est } x_r \end{cases}$$

pour avoir le résultat désiré.

Maintenant, nous allons construire un nouveau modèle–graphe construit sur un nombre minimisé d'atomes. Soit donc J un sous ensemble de I satisfaisant les propriétés du lemme. Nous allons considérer un graphe $G = (V, E)$ défini de la manière suivante:

Les noeuds de G sont caractérisés par leurs ensembles associés et sont de l'un des deux types suivants:

i) Pour chaque $i \in I, x_i \in V$ avec Mx_i comme ensemble associé, et pour chaque littéral $x_i \not\subset x_j$ apparaissant dans la formule, on conserve un noeud u de l'ancien modèle–graphe tel que $u \in Mx_i \backslash Mx_j$.

ii) Les noeuds du deuxième type sont ceux de la famille $(a_j)_{j \in J}$, ils sont associés à eux même.

Observation: Le nombre de noeuds introduits ne dépasse pas $n + k$.

L'ensemble des flèches E est défini par:

$$E = \{(u, v) \in V^2; Mu \in Mv\}$$

Avec pour chaque atome u, $Mu = u$.
Soit M^* l'application définie sur V par:

$M^*v = v$ si v est un atome ou une variable interprétée par un atome dans M.
$M^*v = \{M^*u; (u, v) \in E\}$ dans les autres cas.

M^* est bien définie puisque M est un modèle de φ dans $\mathcal{HFSA}$ ce qui assure l'acyclicité de G. De plus M^* est un modèle pour φ comme le montre le lemme suivant.

Lemme 3.16 *La fonction M^* satisfait les conditions suivantes (Pour les implications $iii) - viii)$, $(A \Longrightarrow B)$ doit être lu comme suit: si A apparaît dans la formule, alors on a B):*

i) $Mu = Mv \Longleftrightarrow M^*u = M^*v$ pour tout $(u, v) \in V^2$;
ii) $Mu \in Mv \Longleftrightarrow M^*u \in M^*v$ pour tout $(u, v) \in V^2$;
iii) $x_i \subset x_j \Longrightarrow M^*x_i \subset M^*x_j$;

iv) $x_i \not\subset x_j \Longrightarrow M^*x_i \not\subset M^*x_j$;
v) $x_i = x_j \cap x_k \Longrightarrow M^*x_i = M^*x_j \cap M^*x_k$;
vi) $x_i = x_j \cup x_k \Longrightarrow M^*x_i = M^*x_j \cup M^*x_k$;
vii) $x_i = x_j \backslash x_k \Longrightarrow M^*x_i = M^*x_j \backslash M^*x_k$;
viii) $x_k = \{x_{i_1}, \ldots, x_{i_r}\} \Longrightarrow M^*x_k = \{M^*x_{i_1}, \ldots, M^*x_{i_r}\}$.

Preuve: i) Le sens $Mu = Mv \Longrightarrow M^*u = M^*v$ est trivial par définition de M^*. Réciproquement, supposons qu'il existe u et v tels que $M^*u = M^*v$ et $Mu \neq Mv$. Et supposons que u soit *E–minimal* au sens suivant: s'il existe d'autres éléments w et w' satisfaisant les mêmes hypothèses alors il ne peut exister de chemin conduisant w ou w' à u. Alors il existe un noeud $a \in V$ tel que $(Ma \in Mu)$ et $(Ma \notin Mv)$ ou $(Ma \in Mv)$ et $(Ma \notin Mu)$. Supposons, par exemple, que $Ma \in Mu$ et $Ma \notin Mv$. Alors, $M^*a \in M^*u$, mais comme $M^*u = M^*v$ on a $M^*a \in M^*v$ et donc il existe $b \in V$ tel que $M^*a = M^*b$ et donc $Ma \neq Mb$ ce qui contredit la E–minimalité de u.

ii) Supposons $Mu \in Mv$ alors $M^*u \in M^*v$ par définition de M^*. Pour la réciproque, supposons $Mu \notin Mv$ et $M^*u \in M^*v$, alors il existe u' dans V tel que $Mu' \in Mv$ et $M^*u' = M^*u$. Il s'en suit, que $Mu = Mu'$ (absurde).
Les autre propriétés peuvent être démontrées en utilisant i) et ii) avec des arguments similaires. □

3.5 Formules conjonctives plates générales. Réduction

Le résultat précédent nous permet d'adapter la procédure de réduction au nouveau cadre et donc de la généraliser aux formules négatives.

Soit $(a_i)_{i=1,\ldots,h}$ la collection d'atomes supplémentaires utiles comme prévus par le théorème précédent.

La nouvelle procédure de réduction est une extension de la première de façon à tenir compte de $(a_i)_{i=1,\ldots,h}$.

Premièrement, nous introduisons de nouveaux entiers $P_{i,j}$, $Q_{i,j}$ et H_i relatifs aux objets obtenus en ajoutant à ceux de φ la famille $(a_i)_{i=1,\ldots,h}$, nous traduisons chaque littéral $x_i \neq x_j$ par $Q_{i,j} = 0$. D'autre part si on considère $(x_i)_{i=n+1,\ldots,n+m'}$ comme étant la nouvelle collection d'objets, il suffit de remplacer m par m' dans chaque indexation dans les formules du système de la section 1, pour avoir un autre système équisatisfiable à φ. Par exemple, le littéral $x_i = \{x_j\}$ sera traduit de la manière suivante (on a seulement rajouté les égalités pour l'indice k entre $n+m+1$ et $n+m'$):

$$\begin{cases} P_{j,i} = 1 \\ C_i = 1 \\ H_i = H_j + 1 \\ \displaystyle\bigwedge_{1 \leq k \leq n+m'} P_{k,i} = Q_{k,j} \end{cases} \qquad (T_4)$$

Le littéral $x_i \notin x_j$ est traduit par

$$\begin{cases} P_{i,j} = 0 \\ H_j + Q_{j,n+1}) \geq 1 \end{cases} \qquad (T_5)$$

Comme dans le cas de la traduction des littéraux positifs, une inégalité du type $((H_j + Q_{j,n+1}) \geq 1)$ est écrite de manière à interdire que la variable x_j puisse être interprétée comme un atome, ce qui garantira l'admissibilité de l'interprétation obtenue. La même remarque s'applique aux traductions suivantes.

Le littéral $x_i = x_j \cap x_k$ est traduit par:

$$\begin{cases} (C_i \leq C_j) \wedge (C_i \leq C_k) \\ (H_i \leq H_j) \wedge (H_i \leq H_k) \\ ((H_i + Q_{i,n+1}) \geq 1) \ \wedge \ ((H_j + Q_{j,n+1}) \geq 1) \ \wedge \ ((H_k + Q_{k,n+1}) \geq 1) \\ \bigwedge\limits_{1 \leq s \leq n+m'} P_{s,i} = inf(P_{s,k}, P_{s,j}) \end{cases} \qquad (T_6)$$

Le littéral $x_i = x_j \backslash x_k$ est traduit par:

$$\begin{cases} C_i \leq C_j \\ H_i \leq H_j \\ ((H_i + Q_{i,n+1}) \geq 1) \ \wedge \ ((H_j + Q_{j,n+1}) \geq 1) \ \wedge \ ((H_k + Q_{k,n+1}) \geq 1) \\ \bigwedge\limits_{1 \leq s \leq n+m'} P_{s,i} = P_{s,j} - inf(P_{s,k}, P_{s,j}) \end{cases} \qquad (T_7)$$

Nous traduisons enfin le littéral $x_i \not\subset x_j$ par:

$$\begin{cases} \sup\limits_{1 \leq s \leq n+m'} (P_{s,i} - P_{s,j}) = 1 \\ (H_i \geq 1) \ \wedge \ (C_i \geq 1) \ \wedge \ ((H_j + Q_{j,n+1}) \geq 1) \end{cases} \qquad (T_8)$$

Théorème 3.17 *Soit φ une formule conjonctive de $\mathcal{MPLS}$ (avec symboles négatifs), alors φ est satisfiable dans $\mathcal{HFSA}$ si et seulement si le système correspondant S'_φ, obtenu via la nouvelle procédure, admet au moins une solution. De plus, la taille de S'_φ est polynomiale en celle de φ, plus précisément en $\mathcal{O}(n^3)$.*

Preuve: Les lemmes 3.10–3.11 restent vrais, de plus si $x \notin y$ (resp $x \neq y$, $x \not\subset y$) apparait dans φ alors on aura nécessairement $Mx \notin My$ (resp $Mx \neq My$, $Mx \not\subset My$). Une démonstration entièrement analogue à celle du théorème 3.9 peut alors être faite. □

4 Expérimentation

Nous avons développé un environnement d'expérimentation qui automatise la réduction d'un système de formules de $\mathcal{MPLS}$ en un système équisatisfiable d'équations et d'inéquations linéaires en entiers bornés (cf. *figure ci–dessous*).

Le système linéaire obtenu est ensuite résolu avec un *solveur d'arithmétique linéaire sur des domaines finis d'entiers*, dans la version actuelle, soit celui du langage CHIP [9], soit celui du langage PROLOG III [7].

Si la réponse de l'environnement est l'échec, cela montre l'insatisfiabilité du système linéaire en entiers et donc celle du système ensembliste initial. Dans le cas où la résolution réussit, l'environnement rend la solution sous une forme compacte, codée par les variables entières. Si on désire obtenir les solutions de manière explicite, un processus d'énumération des solutions entières doit être déclenché, et il faut dans chaque cas reconstruire les objets de $\mathcal{HFSA}$ à partir du système linéaire résolu, comme indiqué dans la preuve du théorème 3.9: fusion des objets égaux (en utilisant les égalités $Q_{i,j} = 1$), puis construction du graphe (en utilisant les égalités $P_{i,j} = 1$) qui décrit les objets de $\mathcal{HFSA}$ associés aux variables de la formule.

Exemples:

1) Soit φ formule conjonctive plate suivante:

$$x = \{z, y\} \wedge y = \{u, v\} \wedge z = \{p, q\} \wedge t = \{x, p, u\} \wedge$$
$$r = \{s, w\} \wedge w = \{s\} \wedge r \subset t \wedge x \in r$$

Cette formule est non satisfiable. En effet un raisonnement sur les objets de φ permet de conduire à la découverte de chaines circulaires d'inclusion ce qui met en défaut l'axiome de fondation ou de régularité. Or, on sait que ce genre de raisonnements implique ordinairement la distinction de nombreux cas, ce qui oblige la création de points de choix supplémentaires. Dans notre approche, ce type d'inconsistance est facilement détecté grace au contrôle de profondeur lors de la traduction des différents littéraux.

2) Dans cet exemple, nous allons considérer le problème des pigeons ($n = 8$) qui peut être exprimé sous forme ensembliste de la manière suivante:

$$x = \{x_i; i = 1, \ldots 8\} \wedge \bigwedge_{j=1,\ldots,9} y_j \subset x \wedge \bigwedge_{j=1,\ldots,9} C_{y_j} = 7 \wedge \bigwedge_{1 \leq j < k \leq 9} y_j \neq y_k$$

Ce problème n'a pas de solution. Mais, il est à noter que la détection de l'inconsistance dans ce cas n'est pas une affaire facile, puisque nécessitant un raisonnement sur les cardinalités. Or, dans le cas où les x_i sont des variables la cardinalité des y_j ne peut être cernée facilement. Dans les approches à base de consistance d'arc [35] pour détecter l'inconsistance, il est nécessaire de parcourir tous les cas possibles [3]. Dans notre approche, les contraintes sur les cardinalités permettent de bien gérer ces situations. Si les contraintes sur les cardinalités figurent dans la formule de départ la formule structurelle des cardinalités sera prise en compte (voir remarque 3.13).

5 Integration dans une approche CLP

Dans le contexte des langages de programmation, l'intégration des ensembles s'effectue sur la base de fragments de la théorie axiomatique des ensembles, en particulier les ensembles finis, ou héréditairement finis comme proposé dans cet article. La structure d'ensemble avec ses opérateurs et relations permet ainsi d'améliorer le niveau d'abstraction et le pouvoir d'expression des langages de programmation, que ce soit dans le style impératif (voir le langage SETL [34], par exemple), ou dans le style déclaratif. Ainsi, en programmation en logique, plusieurs propositions d'extensions ensemblistes sont apparues récemment (i.e. LPS [22], LDL1 [4], {Log} [10] ou [32]). Ces approches se présentent comme une extension des signatures des langages de type Prolog par l'introduction des constructeurs de termes ensemblistes, et donc l'introduction d'expressions ensemblistes dans les clauses de Horn.

Dans ces langages, la résolution SLD est étendue par une procédure d'unification ensembliste. Le principal problème provient du caractère $\mathcal{NP}$–complet de l'unifiabilité de structures ensemblistes [21] et de l'absence d'unicité de l'unificateur le plus général. L'introduction d'un algorithme d'unification ensembliste dans la résolution SLD, comme par exemple dans le langage $\{log\}$ [10], conduit à une multiplication de points de choix qui viennent s'ajouter à ceux provenant des disjonctions de clauses. Ceci rend cette approche peu réaliste du point de vue de l'efficacité.

Dans le cadre du projet CLPS –Constraint Logic Programming with Sets– développé depuis 1990 à l'université de Franche–Comté, nous avons proposé de contourner ce problème en utilisant le paradigme de programmation en logique avec contraintes (cf.[19, 36, 7, 20]). Dans notre approche (cf. [23, 3]), l'unification ensembliste est traitée comme une contrainte d'égalité dont la satisfaction est différée. Cette contrainte est integrée dans le système courant de contraintes, sur lequel est mis en oeuvre un test de consistance. Ce retardement de la résolution de la contrainte peut permettre de bénéficier de l'acquisition de nouvelles informations lors de l'exécution du programme, ce qui permet de réduire le nombre de solutions. Ainsi, la résolution de l'équation $X \cup Y \cup Z = \{a, b, c\}$ rend 343 solutions différentes, mais l'acquisition de la contrainte $X = \{a\}$, fait passer ce nombre à 36. Si la contrainte $\{d\} \subset Y$ est acquise, le système devient inconsistant et l'équation de départ n'a pas de solution. De la même façon, toutes les formules ensemblistes sont traitées comme des contraintes sur lequelles sont mises en oeuvre des techniques de tests de satisfiabilité, puis une génération de solutions.

Dans le cadre du projet CLPS, au sein duquel se situent les travaux présentés dans cet article, nous mettons en oeuvre des techniques de résolution des contraintes ensemblistes qui associent les techniques "domaines finis" avec la réduction en programmation linéaire en entiers. Les techniques sur les "domaines finis" utilisées consistent principalement en une extension des algorithmes de consistance d'arc, avec des mecanismes d'énumération de type "forward checking". En CLPS, le système de contraintes est modelisé sous forme de graphe étiquetté, où les noeuds représentent les objets du système (i.e. ses variables et atomes) et où les flèches représentent les différentes relations qui lient les objets du système (cf. [3]). Le graphe est construit incrémentalement durant l'acquisition des contraintes. Quand une contrainte est rencontrée, elle subit un test de consistance de type avant d'être transformée en une conjonction de contraintes élémentaires (i.e. contraintes de la forme: $x \in y, x \notin y, x \subset y, x \not\subset y$ et $x \neq y$). Ces dernières correspondent à l'ensemble des informations que l'on peut dériver de la contrainte courante, et que l'on exprime à l'aide des symboles $\in, \notin, \subset, \not\subset$ et $\neq$. Les formules élémentaires sont ensuite réintégrées dans le système de contraintes courant, et sont testées en utilisant des procédures généralisant les algorithmes de consistance d'arcs [35].

Dans de nombreux problèmes, les contraintes peuvent faire apparaitre des variables sans aucune spécification de domaine. Et donc ces techniques se trouvent être incomplètes. L'algorithme par énumération et propagation, ne peut dans ce cas, permettre de montrer la satisfiabilité.

En l'absence de domaines, l'approche de réduction que nous présentons ici se substitue à la génération des valeurs par énumération et propagation. En effet, là où il y a une ambiguité —c'est-à-dire dans le cas de contraintes faisant intervenir des objets sans domaines— nous traduisons le sous graphe connexe correspondant en un système d'inéquations et d'équations linéaires en entiers. Ce dernier sera résolu par un solveur arithmétique (qui peut très bien être un solveur bien inscrit dans le cadre CLP et donc parfaitement integrable dans notre cadre). Les résultats obtenus seront alors explicités sous forme de contraintes élémentaires, qui sont à nouveau réintegrées au système de contraintes.

6 Conclusion

En théorie des ensembles, plusieurs résultats de décidabilité ont été prouvés dans les dernières années (nous mentionnons, par exemple, la série de papiers intitulée *"Decision Procedures for Elementary Sublanguages of Set Theory"* introduite par [11], on peut aussi se reférer à [6] pour une bibliographie étendue). Cependant, les efforts entrepris pour développer un démonstrateur automatique dans des fragments de la théorie des ensembles se heurtent à des problèmes de complexité importante, et les techniques utilisées jusqu'à présent ne sont pas suffisamment puissantes pour traiter efficacement des opérations sur des structures ensemblistes.

Dans cet article, nous avons présenté un schéma de réduction de tout système de contraintes ensemblistes élémentaires en un système équisatisfiable de programmation linéaire en entiers. Notre réduction est polynomiale (majorée par $\mathcal{O}(n^3)$). Elle permet ainsi de réutiliser, dans le contexte des ensembles, les travaux et les outils développés pour la résolution des systèmes linéaires.

Nous utilisons directement ce résultat dans le cadre du développement d'un langage de programmation en logique avec contraintes sur les ensembles héréditairement finis. La réduction est mise en oeuvre sur le système de contraintes ensemblistes associées aux variables ne possédant pas de domaine de valeurs. Cela permet ainsi d'assurer la complétude de la procédure de résolution utilisée dans ce langage.

Il faut noter que d'autres travaux, dans des contextes différents suivent une voie similaire. Par exemple, certains auteurs traitent certains problèmes ensemblistes comme un cas particulier de l'unification associative-commutative et idempotente. Voir, par exemple, les travaux de Siekmann [31]. Dans le système setCAL [33], Y. Sato, K. Sakai et S. Menju ont proposé une autre approche pour traiter des contraintes ensemblistes. Cette proposistion consiste à écrire les contraintes ensemblistes en termes d'anneaux de polynômes (booléens) qui sont résolus en utilisant les bases de Groebner.

D'"autres contraintes ensemblistes" sont au coeur d'un autre courant de recherche (cf. [13, 1, 2, 12]). Dans ce cadre, les contraintes ensemblistes désignent des inclusions (non inclusions) formelles entre sous ensembles de l'ensemble T_Σ des termes instanciés construits sur un alphabet Σ qui consiste en: – un ensemble de variables, – des opérateurs de la théorie des ensembles à savoir 0 et 1 dénotant respectivement l'ensemble vide et celui de tous les termes, $\cup$, $\cap$, $\sim$ et – pour chaque opérateur n-aire f de Σ, un opérateur ensembliste correspondant défini par

$$f(A_1, \ldots, A_n) = \{ft_1 \ldots t_n \mid t_i \in A_i, 1 \leq i \leq n\}$$

Ce type de contraintes a plusieurs applications dans le domaine de l'analyse de programme et de l'inférence de type. Cependant, cette approche traite un problème radicalement différent de celui que nous abordons.

Dans ce contexte actif de recherche sur les contraintes ensemblistes, nous poursuivons les travaux présentés dans cet article dans trois directions. Il s'agit:

– d'étendre les schémas de réduction au cas de multi–ensembles et séquences [18],

– d'utiliser les caractéristiques des systèmes en entiers générés (matrices creuses, cadre assez borné avec plusieurs variables booléennes ou pseudo booléennes...) pour développer une procédure de résolution bien adaptée,

– d'étendre ce résultat à la combinaison, dans le système de départ, de contraintes ensemblistes et de contraintes numériques pouvant porter sur les mêmes objets (e.g. $x \in z \wedge x \geq 3$, si $\mathbb{N}$ est contenu dans l'ensemble des atomes).

Remerciements

Nous remercions les rapporteurs anonymes pour les différentes corrections et suggestions.

References

[1] A. Aiken and E. Wimmers, *Solving Systems of Set Constraints*, In 7^{th} Symposium on LICS, 1992.

[2] A. Aiken, D. Kozen, E. Wimmers, *Decidability of Systems of Set Constraints with Negative Constraints*, Technical Report 93–1362, Computer Science Departement, Cornell University, June 1993.

[3] F. Ambert, B. Legeard and E. Legros, *Constraint Logic Programming on Sets and Multisets*, Workshop on Constraint Languages and their use in Problem modelling, in conjunction with ILPS'94, Ithaca, USA,pp. 151–165, November 18, 1994.

[4] C. Beeri, S. Naqvi, R. Ramakrishnan, O. Shmueli, S. Tsur, *Sets and negation in a logic database language (LDL1)*, Proceedings of the 6th Annual ACM SIGMOD Symposium on principles of Database Systems, Vol 16, N3, pp 21-37, 1987.

[5] F. M. Brown, *Towards the Automation of Set Theory and its Logic*, Artificial Intelligence, 10 (1978) 281–316.

[6] D. Cantone, A. Ferro and E. Omodeo, *Computable Set Theory*, Academic Press 1990.

[7] A. Colmerauer, *An Introduction to PROLOG III*, In Communication of the A.C.M Vol.33 No 7. July 1990.

[8] D. Cantone, E. Omodeo and A. Policriti, *The Automation of Syllogistic II. Optimization and Complexity Issues.* Journal of Automated Reasoning, 6: 173-187, 1990.

[9] M. Dincbas, P. Van Henteryck, H. Simonis, A. Aggoun, T. Graf, F.Berthier, *The Constraint Logic Programming Language CHIP*, Proceeding of the International Conference on Fifth genaration Computer system (Tokyo 88).

[10] A. Dovier, E. Omodeo, E Pontelli and G.F. Rossi, *log: A Logic Programming Language With Finite Sets*, Proceedings of The Eighth International Conference in Logic Programming, (K. Furukawa, ed), The MIT Press, 1991, p111-124.

[11] A. Ferro, E. Omodeo and J. T. Schwartz, *Decision Procedures for Elementary Sublanguages of Set Theory. I: Multi–level syllogistic and some Extensions*, Comm. Pure and Appl. Math.vol.XXXIII 1980, 599-608.

[12] R. Gilleron, S. Tison, and M. Tommasi, *Solving Systems of Set Constraints using Tree Automata*, In Proc. STACS, LNCS 665, pages 505-514, Springer-Verlag, Fevrier 1993.

[13] N. Heintz and J. Jaffar, *A Decision Procedure for a Class of Set Constraints*, LICS 90.

[14] M. Hibti, *NP-Complétude du langage MLS*, Mémoire de DEA de Mathématiques, Université de Franche-Comté, Septembre 1991.

[15] M. Hibti, *Satisfiabilité dans Certains Langages Ensemblistes*, Actes de La Journée Ensemble, Rapport de recherche LIFO Orléans 9 Avril 92.

[16] M. Hibti, *Décidabilité et Complexité de Systèmes de Contraintes Ensemblistes*, Thèse de Doctorat de Mathématiques, Université de Franche–Comté, N d'ordre 464, Juin 1995.

[17] M. Hibti, H. Lombardi and B. Legeard, *Deciding in HFS–Theory via Linear Integer Programming with Applivation to Set–Unification*, in Proc. of the 4th International Conference on Logic Programming and Automated Reasonning LPAR 93, St Petersbourg, pp 170–181, Springer LNCS 698.

[18] M. Hibti, B. Legeard and H. Lombardi , *Decision Procedure for Constraints over Sets Multisets and Sequences*, Research report LAB –RRIAP 9402.

[19] J. Jaffar and J. L. Lassez, *Constraint Logic Programming*, Proceedings of the 14^{th} ACM Conference on Principle of Programming Languages, POPL, pp 111–119, Munich 1987.

[20] J. Jaffar and M. J. Maher, *Constraint Logic Programming: A Survey*, J. of Logic Programming Vol 19/20 May/July, pp. 503–582,1994.

[21] D. Kapur and P. Narendran, *NP-Completeness of The Set Unification and Matching Problems*, Proc. of the ICAD, Oxford, July 1986, Springer LNCS 230, 489-495.

[22] G.M. Kuper, *Logic Programming with Sets*, Research Report IBM Yorktown Heights, RC 12378, Dec. 1987.

[23] B. Legeard, H. Lombardi, E. Legros and M. Hibti, *A Satisfaction Approach to Set Unification*, in Proceedings of the 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language, EC2, Vol 1, pp 265–276, Avignon 1993.

[24] A. K. Mackworth, *"Consistency in Network of Relations"* , Journal of Artificial Intelligence, vol. 8, N 1, pp 99-118, 1977.

[25] B. A. NADEL, *Constraints Satisfaction Algorithms*, Journal of Computer Intelligence, vol 5, pp 188-224, 1989.

[26] A. POLICRITI, *NP–completeness of MLS*, technical report, University of Udine, 1990.

[27] F. PARLAMENTO AND A. POLICRITI, *Decision Procedures for Elementary Sublanguages of Set Theory. XIII: Model Graph, Reflexion and Decidability*, Journal of Automated Reasonning, vol.7, 1991.

[28] F. PARLAMENTO AND A. POLICRITI, *Undecidability Results for Restricted Universallu Quantified Formulae of Set Theory*, Com. on Pure and Applied Mathematics, Vol. XLVI, pp 57–73, 1993.

[29] D. PASTRE, *Automatic Theorem Proving in Set Theory*, Artificial Intelligence, 10 (1978) 1–27.

[30] K.J. PERRY, K.V. PALEM, K. MCALOON, G.M KUPER, *The Complexity of Logic Programming with Sets*, Research Report IBM Yorktown Heights, RC 12887,1987.

[31] J. H. SIEKMANN, *Unification Theory*, In Unification, Edited by C. Kirchner, Academic Press, pp 1-68, 1990.

[32] R. SIGAL, *Desiderata for Logic Programming with Sets*, GULP, Proceedings of the 4th National Conference on Logic Programming, pp 127-141, Bologna, 1989.

[33] Y. SATO, K. SAKAI AND S. MENJU, *SetCAL- a Solver of Set Constraints in CAL System*, Technical Report TM- 0963, ICOT, 1990.

[34] J. T. SCHWARTS, R. B. DEWAR, E. DUBINSKY, E. SCHONBERG, *Programming with Sets – An Introduction to SETL*, 493 pages, Springer–Verlag Editions, Berlin, 1986.

[35] E. TSANG, *Foundations of Constraint Satisfaction*, Academic Press, 1993.

[36] P. VAN HENTENRYCK, *Constraint Satisfaction in Logic Programming*, The MIT Press, 224 Pages, 1989.