Solving First-Order Constraints in the Theory of the Evaluated Trees

Thi-Bich-Hanh Dao, Khalil Djelloul

Solving First-Order Constraints in the Theory of the Evaluated Trees

Thi-Bich-Hanh Dao[1] and Khalil Djelloul[2]

[1] Laboratoire d'Informatique Fondamentale d'Orléans, France.
[2] Faculty of Computer Science, University of Ulm, Germany.

Abstract. We present in this paper a first-order extension of the solver of Prolog III, by giving not only a decision procedure, but a full first-order constraint solver in the theory T of the evaluated trees, which is a combination of the theory of finite or infinite trees and the theory of the rational numbers with addition, subtraction and a linear dense order relation. The solver is given in the form of 28 rewriting rules which transform any first-order formula φ into an equivalent disjunction ϕ of simple formulas in which the solutions of the free variables are expressed in a clear and explicit way. The correctness of our algorithm implies the completeness of a first-order theory built on the model of Prolog III.

1 Introduction

The algebra of finite or infinite trees plays a fundamental role in computer science: it is a model for data structures, program schemes and program executions. As early as 1976, G. Huet proposed an algorithm for unifying infinite terms, that is solving equations in that algebra [13]. B. Courcelle has studied the properties of infinite trees in the scope of recursive program schemes [7]. A. Colmerauer has described the execution of Prolog II, III and IV programs in terms of solving equations and disequations in that algebra [4, 3, 1].

The unification of finite terms, i.e. solving conjunctions of equations in the theory of finite trees has first been studied by A. Robinson [24]. Some algorithms with better complexities have been proposed after by M.S. Paterson and M.N.Wegman [22] and A. Martelli and U. Montanari [21]. A good synthesis on this field can be found in the paper of J.P. Jouannaud and C. Kirchner [15]. Solving conjunctions of equations on infinite trees has been studied by G. Huet [13], by A. Colmerauer [5] and by J. Jaffar [14]. Solving conjunctions of equations and disequations on finite or infinite trees has been studied by H.J. Burckert [2] and A. Colmerauer [4]. An incremental algorithm for solving conjunctions of equations and disequations on rational trees has then been proposed by V.Ramachandran and P. Van Hentenryck [23].

On the other hand, M.J. Maher has axiomatized the theory of finite or infinite trees and showed its completeness using a decision procedure which transforms any first-order formula into a Boolean combination of quantified conjunctions of atomic formulas [19]. A much more general decision procedure was recently given by K. Djelloul in the frame of decomposable theories [12].

We have then extended Maher's theory of finite or infinite trees by giving a complete first-order axiomatization of the evaluated trees which are a combination of finite or infinite trees with construction operations and the rational numbers with addition, subtraction and a linear dense order relation [9]. This theory, denoted by T, reflects essentially to Prolog III which has been modeled by A. Colmerauer [3] using a combination of trees and rational numbers. Nevertheless, the solver of Prolog III is not able to solve arbitrary quantified first-order constraints built on a combination of trees and rational numbers.

A first attempt of an extension of the solver of Prolog III was given in [11]. It consists in a decision procedure which for every proposition (formula without free variables) gives either true of false in T. Unfortunately, this decision procedure is not able to solve first-order constraints having free variables. In fact, it does not warrant that the solutions of the free variables of a solved formula are expressed in a clear and explicit way and can even produce, starting from a formula φ which contains free variables, an equivalent solved formula ϕ having free variables but being always false (or always true) in T. The appropriate solved formula of φ in this case should be the formula *false* (or the formula *true*) instead of ϕ.

Much more elaborated algorithms are then needed, specially when we want to induce solved formulas expressing solutions of complex first-order constraint satisfaction problems in T. Of course, our goal in these kinds of problems is not only to know if there exist solutions or not, but to express these solutions in the form of a solved formula which is either the formula *true* (i.e. the problem is always satisfiable) or the formula *false* (i.e. the problem is always unsatisfiable) or a simple first-order formula which is neither equivalent to *true* nor to *false* and where the solutions of the free variables are expressed in a clear and explicit way. Algorithms which are able to produce such a formula are called *first-order constraint solvers*.

We present in this paper, not only a decision procedure, but a full first-order constraint solver which gives clear and explicit solutions for any first-order constraint satisfaction problem in T. Our solver is not simply a combination of an algorithm over trees with one over rational numbers, but a powerful mechanism to solve mixed constraints. It includes full systems of typing deduction and constraint simplification and propagation. One of the major difficulties in this work resides in the fact that (i) the theory of finite or infinite trees does not accept full elimination of quantifiers, (ii) every algorithm deciding propositions in the theory of finite or infinite trees has a non-elementary complexity [25] and (iii) the function symbols $+$ and $-$ of T have two different behaviors whether they are applied on trees or rational numbers. For example $+(1,1)$ is the rational number 2, while $+(1,f_0)$ is the tree whose root is labeled $+$ and whose sons are 1 and the tree's constant f_0.

One of the practical applications of our solver is a powerful extension of the internal solver of Prolog III by allowing the user to handle general first-order constraints and solve them in T. The solver will then give the solutions of the free variables in all the models of T and present them in a clear and explicit way. As far as we know, this is the first algorithm which is able to do a such

work. Solving quantified constraints over trees and rational numbers can also be used in PSPACE-complete decision problems from areas such as planning under uncertainty, adversary game playing, and model checking. For example, in game playing we may want to find a winning strategy *for all* possible moves of the opponent. In a manufacturing problem it may be required that a configuration must be possible *for all* possible sequences of user choices. Finally, when planning in a safety critical environment, such as a nuclear power station, we require that an action is possible *for every* eventuality.

The paper is organized in four sections followed by a conclusion. This introduction is the first section. In Section 2 we present the theory of the evaluated trees and introduce an example of a complex constraint in this theory. In Section 3, we define the notions of basic formulas, blocks and solved blocks in T which are particular conjunctions of atomic formulas. We end this section by showing that every quantified solved block can be decomposed in three embedded sequences of quantifications having particular properties which enable us to eliminate some quantifiers. In Section 4, we present the working formulas, the general solved formulas and the algorithm of constraint solving in T. The algorithm is presented in the form of 28 rewriting rules and transforms an initial working formula of depth d into a final working formula of depth less than or equal to three. The main idea behind this algorithm consists in (1) a top-down simplification and propagation of constraints. In each level, quantified blocks are locally solved, decomposed and then propagated to the embedded sub-formulas. Inconsistent sub-formulas are also removed (2) a bottom-up elimination of quantifiers and working formulas' depth decrease using distribution. The disjunction ϕ of general solved formulas extracted from the final working formula is either the formula *false* or *true* or a formula having at least one free variable, being equivalent neither to *false* nor to *true* in T, and where the solutions of the free variables are expressed in a clear and explicit way. We end this section by giving an example of a constraint having two free variables but being always false in T.

The algorithm represented by a set of rewriting rules and the general solved formulas are our main contribution in this paper. The expressiveness and clearness of the solutions of the free variables in the final solved formula are our main goal in this work.

2 Theory T of evaluated trees

2.1 Preliminaries

Let F be an infinite set of *function symbols* containing the symbols $+$, $-$, 0 and 1. To each element of F is associated a non-negative integer, its *arity*. The arities of $+$, $-$, 0 and 1 are respectively 2, 1, 0 and 0. Let $R = \{<, num, tree\}$ be the set of relation symbols, of respective arities 2, 1 and 1. Let V be an infinite countable set of *variables*. A *term* is an expression of the form x or $ft_1 \dots t_n$ where $n \geq 0$, f an n-ary symbol in F and the t_i's are shorter terms. A *formula* is an expression of the forms:

$s{=}t$, $rt_1..t_n$, *true*, *false*, $\neg(\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$, $\exists x \varphi$, $\forall x \varphi$,

where $x \in V$, s, t and the t_i's are terms, r is an n-ary relation symbol in R and φ and ψ are shorter formulas. The first four forms are called atomic. An occurrence of a variable x in a formula is *bound* if it occurs in a sub-formula of the form $(\exists x\varphi)$ or $(\forall x\varphi)$. It is *free* otherwise. The *free variables* of a formula are those which have at least a free occurrence in the formula. For each formula φ, we denote by $var(\varphi)$ the set of all the free variables of φ. Let $\bar{x} = x_1 \dots x_n$ and $\bar{y} = y_1 \dots y_n$ be two vectors of variables of the same length. The empty vector is denoted by ε. Let φ and $\varphi(\bar{x})$ be formulas. We write

$$\begin{array}{ll}
\exists \bar{x}\, \varphi & \text{for } \exists x_1 ... \exists x_n\, \varphi, \\
\forall \bar{x}\, \varphi & \text{for } \forall x_1 ... \forall x_n\, \varphi, \\
\exists ?\bar{x}\, \varphi(\bar{x}) & \text{for } \forall \bar{x} \forall \bar{y}\, \varphi(\bar{x}) \wedge \varphi(\bar{y}) \rightarrow \bigwedge_{i \in \{1,\dots,n\}} x_i = y_i, \\
\exists !\bar{x}\, \varphi & \text{for } (\exists \bar{x}\, \varphi) \wedge (\exists ?\bar{x}\, \varphi).
\end{array}$$

Semantically, the new quantifiers $\exists ?$ and $\exists !$ simply mean "at most one" and "one and only one".

2.2 Axiomatization of T

Let a be a positive integer and let $t_1, ..., t_n$ be terms. Let us denote by:

- $t_1 < t_2$, the term $< t_1 t_2$,
- $t_1 + t_2$, the term $+ t_1 t_2$,
- $t_1 + t_2 + t_3$, the term $+ t_1 (+ t_2 t_3)$,
- $0 t_1$, the term 0,
- $a t_1$, the term $\underbrace{t_1 + \dots + t_1}_{a}$,
- $-a t_1$, the term $\underbrace{(-t_1) + \dots + (-t_1)}_{a}$.

The theory T of the evaluated trees is the set of first-order propositions of the following forms:

$$\begin{array}{ll}
1 & \forall \bar{x} \forall \bar{y}\, ((tree\, f \bar{x}) \wedge (tree\, f \bar{y}) \wedge f\bar{x} = f\bar{y}) \rightarrow \bigwedge_i x_i = y_i, \\
2 & \forall \bar{x} \forall \bar{y}\, f\bar{x} = g\bar{y} \rightarrow num\, f\bar{x} \wedge num\, g\bar{y}, \\
3 & \forall \bar{x} \forall \bar{y}\, ((\bigwedge_{i \in I} num\, x_i) \wedge (\bigwedge_{j \in J} tree\, y_j)) \rightarrow (\exists ! \bar{z}\, \bigwedge_{k \in K} (tree\, z_k \wedge z_k = t_k(\bar{x}, \bar{y}, \bar{z}))), \\
4 & \forall x \forall y\, x < y \rightarrow (num\, x \wedge num\, y), \\
5 & \forall x \forall y\, num\, x + y \leftrightarrow num\, x \wedge num\, y, \\
6 & \forall x\, num\, -x \leftrightarrow num\, x, \\
7 & \forall \bar{x}\, tree\, h\bar{x}, \\
8 & \forall x \forall y\, (num\, x \wedge num\, y) \rightarrow x + y = y + x, \\
9 & \forall x \forall y \forall z\, (num\, x \wedge num\, y \wedge num\, z) \rightarrow x + (y + z) = (x + y) + z, \\
10 & \forall x\, num\, x \rightarrow x + 0 = x, \\
11 & \forall x\, num\, x \rightarrow x + (-x) = 0, \\
12_n & \forall x\, num\, x \rightarrow (nx = 0 \rightarrow x = 0), \\
13_n & \forall x\, num\, x \rightarrow \exists ! y\, num\, y \wedge ny = x, \\
14 & \forall x\, num\, x \rightarrow \neg x < x, \\
15 & \forall x \forall y \forall z\, num\, x \wedge num\, y \wedge num\, z \rightarrow ((x < y \wedge y < z) \rightarrow x < z), \\
16 & \forall x \forall y\, (num\, x \wedge num\, y) \rightarrow (x < y \vee x = y \vee y < x), \\
17 & \forall x \forall y\, (num\, x \wedge num\, y) \rightarrow (x < y \rightarrow (\exists z\, num\, z \wedge x < z \wedge z < y)), \\
18 & \forall x\, num\, x \rightarrow (\exists y\, num\, y \wedge x < y), \\
19 & \forall x\, num\, x \rightarrow (\exists y\, num\, y \wedge y < x), \\
20 & \forall x \forall y \forall z\, (num\, x \wedge num\, y \wedge num\, z) \rightarrow (x < y \rightarrow (x + z < y + z)), \\
21 & \forall x\, (\neg num\, x) \leftrightarrow tree\, x \\
22 & 0 < 1,
\end{array}$$

where n is a non-null integer, f and g are two distinct function symbols taken from F, $h \in F - \{+, -, 0, 1\}$, x, y, z are variables, $\bar{x}$ is a vector of variables

x_i, $\bar{y}$ is a vector of variables y_i, $\bar{z}$ is a vector of distinct variables z_i, I and J are finite possibly empty sets, and where $t_k(\bar{x}, \bar{y}, \bar{z})$ is a term which begins by a function symbol f_k element of $F - \{0, 1\}$ followed by variables taken from $\bar{x}$ or $\bar{y}$ or $\bar{z}$. Moreover, if $f_k \in \{+, -\}$ then $t_k(\bar{x}, \bar{y}, \bar{z})$ contains at least one variable taken from $\bar{y}$ or $\bar{z}$. The axiom 3 shows that all models of T contain infinite trees. In fact we have $T \models \exists! z\, z = fz \wedge \mathit{tree}\, z$ for $I = J = \emptyset$. In this case, the tree z is an infinite tree of the form $f(f(f(...)))$. Note that we have not $T \models \forall x\, \mathit{num}\, x \rightarrow (\exists! z\, z = x{+}x \wedge \mathit{tree}\, z)$, since we have $T \models \mathit{num}\, x \leftrightarrow \mathit{num}\, (x{+}x)$ according to axiom 5 which contradicts $\mathit{tree}\, z$ and $z = x + x$. This is why we have a condition if f_k belongs to $\{+, -\}$.

This theory has as model (possibly) infinite trees whose nodes are labelled by $\boldsymbol{Q} \cup F$ such that each subtree labelled by $\boldsymbol{Q} \cup \{+, -\}$ is evaluated in $\boldsymbol{Q}$ and reduced to a leaf labeled by an element of $\boldsymbol{Q}$.

Let us now introduce an example of *constraints* in T. Let us consider the following two-player game: An ordered pair (n, m) of non-negative rational numbers is given and one after another each player subtracts 1 or 2 from n or m but keeping n and m non-negative. The first player who cannot play any more has lost.

Suppose that it is the turn of player A to play. A position (n, m) is called *k-winning* if, no matter the way the other player B plays, it is always possible for A to win, after having made at most k moves. The constraint expressing that a position x is k-winning is:

$$\mathit{winning}_k(x) \leftrightarrow \left[\begin{array}{c} \exists y \mathit{move}(x, y) \wedge \neg(\exists x \mathit{move}(y, x) \wedge \\ \neg(\exists y \mathit{move}(x, y) \wedge \neg(\exists x \mathit{move}(y, x) \wedge \neg(\ldots \wedge \\ \neg(\exists y \mathit{move}(x, y) \wedge \neg(\exists x \mathit{move}(y, x) \wedge \neg(\mathit{false}\underbrace{))\ldots)}_{2k} \end{array}\right]$$

Each position (n, m) is represented by $c(i, j)$ with c a function symbol of arity 2 and $i, j \in \boldsymbol{Q}$. The constraint $\mathit{move}(x, y)$ is defined by

$$\left[\begin{array}{l} (\exists i \exists j\, x = c(i, j) \wedge y = c(i-1, j) \wedge i > 1 \wedge j > 0) \vee \\ (\exists i \exists j\, x = c(i, j) \wedge y = c(i-2, j) \wedge i > 2 \wedge j > 0) \vee \\ (\exists i \exists j\, x = c(i, j) \wedge y = c(i, j-1) \wedge i > 0 \wedge j > 1) \vee \\ (\exists i \exists j\, x = c(i, j) \wedge y = c(i, j-2) \wedge i > 0 \wedge j > 2) \vee \\ (\neg(\exists i \exists j\, x = c(i, j) \wedge \mathit{num}\, i \wedge \mathit{num}\, j) \wedge x = y) \end{array}\right]$$

By replacing the definition of *move* in the constraint $\mathit{winning}_k(x)$, we have a first-order constraint with one free variable x in the theory T of evaluated trees. Solving this constraint means finding all the positions x which are k-winning.

3 Block and quantified block in T

We will now present structured formulas called blocks and show some of their properties. Essentially a block is a conjunction of atomic formulas where all the variables are well typed and which gives enough informations to be locally solved. We will also define a mechanism to decompose each quantified block in three quantified blocks having interesting properties that will help us for solving first-order constraints on quantified blocks.

3.1 Basic formula and block in T

Suppose that the variables of V are ordered by a linear strict and dense order relation without endpoints, denoted by " $\succ$ ". For each formula φ, the bound variables are renamed such that for each sub-formula of φ we have $x \succ y$ for each bound variable x and each free variable y. We denote by $\Sigma_{i=1}^{n} t_i$ the term $\overline{t_1 + \ldots + t_n} + 0$ with $\overline{t_1 + \ldots + t_n}$ the term $t_1 + \ldots + t_n$ where all the terms 0 have been removed.

Let $f \in F - \{0, 1\}$, $a_0 \in \boldsymbol{Z}$ and $a_i \in \boldsymbol{Z}$. We call *leader* of the equation $x_0 = fx_1...x_n$ or $x_0 = x_1$ the variable x_0. We call *leader* of the formula $\Sigma_{i=1}^{n} a_i x_i = a_0 1$ the greatest variable x_k (in the order $\succ$) such that $a_k \neq 0$.

Let $f \in F$, $a_0 \in \boldsymbol{Z}$ and $a_i \in \boldsymbol{Z}$. We call *basic formula* every conjunction α of formulas of the form:

- *true*, *false*, $num\, x$, $tree\, x$,
- $x = y$, $x = fy_1...y_n$, $\Sigma_{i=1}^{n} a_i x_i = a_0 1$, $\Sigma_{i=1}^{n} a_i x_i < a_0 1$.

The formulas $num\, x$ and $tree\, x$ are called *typing constraints*. The formulas $x = y$, $x = fy_1...y_n$, $\Sigma_{i=1}^{n} a_i x_i = a_0 1$ are called *equations*. The formula $\Sigma_{i=1}^{n} a_i x_i < a_0 1$ is called *inequation*. Let α be a basic formula:

(1) We say that "$num\, x$ *is a consequence* of α" iff α contains at least one of the following sub-formulas: $num\, x$, $x = y \wedge num\, y$, $y = x \wedge num\, y$, $x = -y \wedge num\, y$, $y = -x \wedge num\, y$, $z = y + x \wedge num\, z$, $z = x + y \wedge num\, z$, $x = y + z \wedge num\, z \wedge num\, y$, $x = 0$, $x = 1$, $\Sigma_i a_i x_i = a_0 1$ or $\Sigma_i a_i x_i < a_0 1$ and x is one of the x_i's.

(2) We say that "$tree\, x$ *is a consequence* of α" iff α contains at least one of the following sub-formulas: $tree\, x$, $x = y \wedge tree\, y$, $y = x \wedge tree\, y$, $x = -y \wedge tree\, y$, $y = -x \wedge tree\, y$, $x = y + z \wedge tree\, z$, $x = z + y \wedge tree\, z$, $y = x + z \wedge tree\, y \wedge num\, z$, $y = z + x \wedge tree\, y \wedge num\, z$, $x = hy_1...y_n$, with $h \in F - \{+, -, 0, 1\}$.

(3) We call *tree-section* of α the conjunction α_t of the sub-formulas of α of the form:

- *true*, $tree\, x$,
- $x = y$ or $x = fy_1...y_n$, with $f \in F - \{0, 1\}$ and where x is such that $tree\, x$ is a sub-formula of α.

This tree-section α_t is called *formatted* iff all the left-hand sides of the equations of α_t are distinct and for each equation $x = y$ of α_t we have $x \succ y$.

(4) We call *numeric-section* of α the conjunction α_n of sub-fomulas of α of the form:

- *true*, *false*, $\Sigma_{i=1}^{n} a_i x_i = a_0 1$, $\Sigma_{i=1}^{n} a_i x_i < a_0 1$, $num\, x$,
- $x = y$, $x = -y$, $x = y + z$, where x is such that $num\, x$ is a sub-formula of α.

This numeric-section α_n is called *consistent* iff $T \models \exists \bar{x}\, \alpha_n$ with $\bar{x} = var(\alpha_n)$ and *formatted* iff

- α_n does not contain sub-formulas of the form $x = y$, $x = -y$, $x = y + z$, $0 = a_0 1$, $0 < a_0 1$, with $a_0 \in \boldsymbol{Z}$

– α_n is consistent and each leader of the equations of α_n has one occurrence in only one the equations of α_n and no occurrence in the inequations of α_n.

(5) The variable u is called *reachable* in $\exists\bar{x}\alpha$ if u is a free variable in $\exists\bar{x}\alpha$ or α has a sub-formula of the form $y = t(u) \wedge \mathit{tree}\ y$ with $t(u)$ a term containing u and y a reachable variable. In the last case, the equation $y = t(u)$ is also called reachable in $\exists\bar{x}\alpha$.
Example: In the formula $\exists xyz\, w = fxy \wedge z = v \wedge \mathit{tree}\ w$, the variables w, v, x, y are reachable because w, v are free and x and y occur in the sub-formula $w = fxy \wedge \mathit{tree}\ w$. The variable z is not reachable and since z is bound and v is free, they must be such that $z \succ v$. The equation $w = fxy$ is reachable while the equation $z = v$ is not.

We call *block* every basic formulas α such that for each variable x in α either $\mathit{num}\ x$ or $\mathit{tree}\ x$ is a sub-formula of α and α does not contain sub-formulas of the form:

– $x = 0 \wedge \mathit{tree}\ x$, $x = 1 \wedge \mathit{tree}\ x$,
– $x = y \wedge \mathit{num}\ x \wedge \mathit{tree}\ y$, $x = y \wedge \mathit{tree}\ x \wedge \mathit{num}\ y$,
– $x = -y \wedge \mathit{tree}\ x \wedge \mathit{num}\ y$, $x = -y \wedge \mathit{num}\ x \wedge \mathit{tree}\ y$
– $x = y + z \wedge \mathit{num}\ x \wedge \mathit{tree}\ y$, $x = y + z \wedge \mathit{num}\ x \wedge \mathit{tree}\ z$, $x = h\bar{y} \wedge \mathit{num}\ x$,
– $x = y + z \wedge \mathit{tree}\ x \wedge \mathit{num}\ y \wedge \mathit{num}\ z$,
– $\Sigma_{i=1}^{n} a_i x_i = a_0 1 \wedge \mathit{tree}\ x_k$, $\Sigma_{i=1}^{n} a_i x_i < a_0 1 \wedge \mathit{tree}\ x_k$

with $h \in F - \{+, -, 0, 1\}$, $k \in \{1, ..., n\}$, $a_0 \in \boldsymbol{Z}$ and $a_i \in \boldsymbol{Z}$.

Since each variable x in a block is typed i.e. occurs in a sub-formula of the form $\mathit{num}\ x$ or $\mathit{tree}\ x$, every block α can be divided into two disjoint sections: a tree-section and a numeric-section.

A block α without equations is called *relation block*. A block α without inequations and where each variable has an occurrence in at least one of the equations of α is called *equation* block. A block α is called *solved* iff its tree-section and numerical-section are formatted.

3.2 Decomposition of quantified solved blocks

Let ψ be a formula. Let $\bar{x}$ be a vector of variables and α a solved block such that for all unreachable quantified variable u in $\exists\bar{x}\alpha$ and all reachable quantified variable v in $\exists\bar{x}\alpha$ we have $u \succ v$. We call *decomposition* of the formula $\exists\bar{x}\alpha \wedge \psi$ the formula

$$\exists\bar{x}^1\, \alpha^1 \wedge (\exists\bar{x}^2\, \alpha^2 \wedge (\exists\bar{x}^3\, \alpha^3 \wedge \psi))), \tag{1}$$

obtained as follows : Let X be the set of the variables in $\bar{x}$. Let us decompose the set X into two disjoint subsets: X_r (the set of the elements of X which are reachable in $\exists\bar{x}\alpha$) and X_u. Let *Lead* be the set of the leaders of the equations of α. We have:
– $\bar{x}^1$ is the vector of the variables of X_r.
– $\bar{x}^2$ is the vector of the variables of $X_u - \mathit{Lead}$.
– $\bar{x}^3$ is the vector of the variables of $X_u \cap \mathit{Lead}$.

– α^1 is of the form $\alpha_1^1 \wedge \alpha_2^1$ where α_1^1 is the conjunction of all the equations in $\exists \bar{x} \alpha$ whose leader is reachable, α_2^1 is the conjunction of all the typing constraints of α which concern variables of $var(\alpha_1^1)$.
– α^2 is of the form $\alpha_1^2 \wedge \alpha_2^2$ where α_1^2 is the conjunction of all the inequations of α and α_2^2 is the conjunction of all the typing constraints of α which do not concern variables of $\bar{x}^3$.
– α^3 is of the form $\alpha_1^3 \wedge \alpha_2^3$ where α_1^3 is the conjunction of the other equations and α_2^3 is the conjunction of all the typing constraints of α which concern the variables of $var(\alpha_1^3)$. The restriction on the order $\succ$ of the quantified unreachable and reachable variables is due to an aim to get as leader of the equations of the numeric section of α unreachable variables. If one quantified leader is reachable then we deduce that all the quantified variables of this equation are reachable. This condition will help us for the algorithm of resolution given at Section 4. The intuitions behind this decomposition come from an aim to decompose a quantified solved block into three embedded sections each one having particular properties that enable us either to remove quantifiers or make special distributions in ψ and reduce the size of the formula $\exists \bar{x} \alpha \wedge \psi$.

Let A be the set of the solved blocks. Let A^1 be the set of the formulas of the form $\exists \bar{x}^1 \alpha^1$, where α^1 is a solved equation block and all the variables of $\bar{x}^1$ are reachable in $\exists \bar{x}^1 \alpha^1$. Let A^2 be the set of the solved relation blocks.

Property 3.2.1 *For all decomposed formula of the form (1) we have :* $\exists \bar{x}^1 \alpha^1 \in A^1$, $\alpha^2 \in A^2$, $\alpha^3 \in A$ *and* $T \models \forall \bar{x}^2 \, \alpha^2 \rightarrow \exists! \bar{x}^3 \alpha^3$.

Example 3.2.2 *Let* v, w, x, y, z *be variables such that* $w \succ y \succ z \succ x \succ v$. *Let us decompose the formula*

$$\exists wxyz \begin{bmatrix} v = fvx \wedge w + 2x + (-2)z = 1 \wedge y + 3z = 0 \wedge \\ z < 1 \wedge 3z + 2x < 0 \wedge \\ tree\ v \wedge num\ w \wedge num\ x \wedge num\ y \wedge num\ z \end{bmatrix} \tag{2}$$

The reachable variables in the formula (2) are v *and* x. *We have* $X_r = \{x, v\}$, $X_u = \{w, y, z\}$ *and* $Lead = \{v, w, y\}$. *Since* $w \succ y \succ z \succ x$ *then the formula (2) is equivalent in* T *to the decomposed formula*

$$\begin{bmatrix} \exists x\, v = fvx \wedge tree\ v \wedge num\ x \wedge \\ (\exists z\, z < 1 \wedge 3z + 2x < 0 \wedge num\ z \wedge numx \wedge tree\ v \wedge \\ (\exists wy\, w + 2x + (-2)z = 1 \wedge y + 3z = 0 \wedge num\ w \wedge num\ x \wedge num\ y \wedge num\ z)) \end{bmatrix}$$

Note that the elements of A^1 *does not accept elimination of quantifiers, this is due to the fact that all the variables of* $\bar{x}^1$ *are reachable in* $\exists \bar{x}^1 \, \alpha^1$. *Indeed in the formula* $\exists x \, v = fvx$ *the quantification* $\exists x$ *can not be eliminated in* T.

In all what follows we will use the notations $\bar{x}^1$, $\bar{x}^2$, $\bar{x}^3$, α^1, α^2,α^3 to refer to the decomposition of the formula $\exists \bar{x} \alpha$.

4 Solving first-order constraints in T

4.1 Working and general solved formulas

Definition 4.1.1 *A normalized formula φ of depth $d \geq 1$ is a formula of the form*

$$\neg(\exists \bar{x}\, \alpha \wedge \bigwedge_{i \in I} \varphi_i), \tag{3}$$

with I a finite (possibly empty) set, α a basic formula and the φ_i normalized formulas of depth d_i and $d = 1 + \max\{0, d_1, ..., d_n\}$.

Property 4.1.2 *Every formula is equivalent in T to a normalized formula.*

Definition 4.1.3 *A working formula is a normalized formula in which all the occurrences of $\neg$ are of the form $\neg^k$ with $k \in \{0, ..., 9\}$ and such that each occurrence of a sub-formula of the form*

$$\phi = \neg^k(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \bigwedge_{i \in I} \varphi_i), \tag{4}$$

has $\alpha^p = true$ if $k = 0$ and **satisfies the first k conditions** *of the following condition list if $k > 0$. Here α^p is a solved block and is called propagated constraint section, α^c is a basic formula and is called core constraint section, the φ_i are working formulas, and in the conditions: $\beta^p \wedge \beta^c$ is the conjunction of the equations and relations of the immediate top-working formula ψ of ϕ if it exists. i.e. $\psi = \neg^k(\exists \bar{y} \beta^c \wedge \beta^p \wedge \phi \wedge \bigwedge_{j \in J} \phi_j)$ where ϕ is the formula (4) and ϕ_j are any working formulas.*

1. *if ψ exists then $T \models \alpha^p \wedge \alpha^c \rightarrow \beta^p \wedge \beta^c$, and the tree-sections of α^p and $\beta^c \wedge \beta^p$ have the same set of left-hand side of equations,*
2. *the tree-section of $\alpha^p \wedge \alpha^c$ is formatted and the formula $\alpha^p \wedge \alpha^c$ does not contain tree $x \wedge$ num x for any variable x,*
3. *$\alpha^p \wedge \alpha^c$ is a block,*
4. *the numeric-section of $\alpha^p \wedge \alpha^c$ is consistent, and we have $u \succ v$ for u any unreachable variable in $\bar{x}$ and v any reachable variable in $\bar{x}$,*
5. *$\alpha^p \wedge \alpha^c$ is a solved block,*
6. *α^p is the formula $\beta^c \wedge \beta^p$ if ψ exists, and is the formula true otherwise. The formula α^c is a solved block and for each relation num x (or tree x) in α^p, if x does not occur in an equation or inequation of α^c then num x (resp. tree x) does not occur in α^c,*
7. *$(\exists \bar{x}\, \alpha^c)$ is decomposable into $(\exists \bar{x}^1\, \alpha^{c1} \wedge (\exists \bar{x}^2\, \alpha^{c2} \wedge (\exists \varepsilon\, true)))$,*
8. *$(\exists \bar{x}\, \alpha^c)$ is decomposable into $(\exists \bar{x}^1\, \alpha^{c1} \wedge (\exists \varepsilon\, \alpha^{c2} \wedge (\exists \varepsilon\, true)))$,*
9. *$(\exists \bar{x}\, \alpha^c)$ is decomposable into $(\exists \bar{x}^1\, \alpha^{c1} \wedge (\exists \varepsilon\, true \wedge (\exists \varepsilon\, true)))$.*

We use k in order to be able to control the execution of our rewriting rules on working formulas. We strongly insist on the fact that $\neg^k$ does not mean that the normalized formula satisfies only the k^{th} condition but all the conditions i with $1 \leq i \leq k$. We call *initial* working formula a working formula of the form

$$\neg^6(\exists \varepsilon\, true \wedge \bigwedge_{i \in I} \varphi_i)$$

with φ_i working formulas where all negation symbols $\neg^k$ have $k = 0$ and all propagated constraint sections are reduced to *true*. We call *final* working formula a formula of the form

$$\neg^7(\exists\varepsilon\ true \wedge \bigwedge_{i\in I} \neg^8(\exists\bar{x}_i\ \alpha_i^c \wedge \alpha_i^p \wedge \bigwedge_{j\in J_i} \neg^9(\exists\bar{y}_{ij}\ \beta_{ij}^c \wedge \beta_{ij}^p))), \tag{5}$$

where all the β_{ij}^c are different from *true*.

Definition 4.1.4 *A* general solved formula *is a formula of the form*

$$\exists\bar{x}^1\ \alpha^1 \wedge \alpha^2 \wedge \bigwedge_{i\in I} \neg(\exists\bar{y}_i^1\ \beta_i^1), \tag{6}$$

where $\exists\bar{x}^1\ \alpha^1 \in A^1$, $\alpha^2 \in A^2$, $\exists\bar{y}_i^1\ \beta_i^1 \in A^1$, *all the* $\alpha^1 \wedge \alpha^2 \wedge \beta_i^1$ *are solved blocks and all the* β_i^1 *are different from true.*

According to the properties of $\neg^8$ and $\neg^9$, in the final working formula (5), $\alpha_i^p = true$ and $\beta_{ij}^p = \alpha_i^p \wedge \alpha_i^c$. Thus the formula (5) is equivalent in T to the following disjunction of general solved formulas

$$\bigvee_{i\in I}(\exists\bar{x}_i\ \alpha_i^c \wedge \bigwedge_{j\in J_i} \neg(\exists\bar{y}_{ij}\ \beta_{ij}^c)) \tag{7}$$

Property 4.1.5 *Let* φ *be a general solved formula of the form (6). If* φ *has no free variables then* φ *is the formula true, otherwise neither* $T \models \varphi$ *nor* $T \models \neg\varphi$ *and the solutions of the free variables of* φ *are explicit.*

This result is very important because it shows that for each solved formula φ containing at least one free variable there exists a set of solutions and a set of non-solutions, i.e. φ is neither true nor false in T. A similar result has been shown for the finite trees of J. Lassez [17] and the rational trees of M. Maher [20]. Note also that in all our proofs [8] we have not used the famous independence of inequations [4, 16, 6, 18] but only the condition that the signature of T is infinite (F is infinite) which implies in this case the independence of the inequations.

4.2 Main idea

The general algorithm for solving first-order constraints in T uses a system of rewriting rules. The main idea is to transform an initial working formula of depth d into a final working formula of depth less than or equal to three. The transformation is done in two steps:

(1) The first step is a top-down simplification and propagation. In each sub-working formula, $\alpha^c \wedge \alpha^p$ is transformed to a solved block, then $\exists\bar{x}\alpha^c$ is decomposed into three parts as in subsection 3.2. The third part is eliminated and added to the core-constraint section of the immediate sub-working formulas using a special property of the quantifier $\exists!$. The constraints of the two other parts

in α^p are propagated to the propagated-constraint section of the immediate sub-working formulas. In this step, the rules 1 to 24 are applied and transform the initial working formula into a working formula where each negation symbol is of the form $\neg^7$.

(2) The second step is a bottom-up simplification and elimination of quantifiers. This step is done by the rules 25 to 28. In each sub-working formula of depth one or two, the rule 25 eliminates quantified variables of the second part of the decomposition (the third one had been already removed in the first step). The rule 26 eliminates the constraints of the second part in the deepest level. Each sub-working formula of depth 3 is transformed step by step to a conjunction of working formulas of depth 2 by the rule 28 using a property of the quantifier $\exists?$. The transformations in this step can create new sub-working formulas where the first step needs to be done. At the end of the transformation, we obtain a final working formula of depth less than or equal to 3.

4.3 Rewriting rules

We present in Figure 1 the rewriting rules which transform an initial working formula into an equivalent final working formula. To apply the rule $p_1 \Longrightarrow p_2$ to the working formula p means to replace in p, a sub-formula p_1 by the formula p_2, by considering that the connector $\wedge$ is associative and commutative.

In all these rules, α is a basic formula, φ and ψ are conjunctions of working formulas.

In the rules 1 to 14, the equations and relations in α^c and α^p are mixed by considering the connector $\wedge$ associative and commutative. In these rules, except the rule 6, all modifications in the right hand side are done in α^c, since α^p is a solved block.

In the rule 2, f and g are two distinct function symbols taken from F. The rules 4, 6, 7, are applied only if $x \succ y$. This condition prevents infinite loops and makes the procedure terminating. In the rule 5, the equation $x = fz_1...z_n$ does not belong to α^p. In the rule 6, if the equation $x = fz_1...z_n$ belongs to α^p, then $x = y \wedge \mathit{tree}\ y$ is moved to α^p. In the rule 7, the equation $x = z$ does not belong to α^p.

In the rule 9, $a_0 > 0$. In the rules 13 and 14 the variable x_k is the leader of the equation $\Sigma_i a_i x_i = a_0 1$ and $b_k \neq 0$. Moreover, the equation $\Sigma_j b_j x_j = b_0 1$ does not belong to α^p. In the rule 14, the relation $\Sigma_j b_j x_j < b_0 1$ does not belong to α^p and $\lambda = 1$ if $a_k > 0$ and $\lambda = -1$ otherwise.

In the rule 15, the tree section of $\alpha^c \wedge \alpha^p$ is formatted and there is no sub-formula in $\alpha^c \wedge \alpha^p$ of the form $\mathit{num}\ x \wedge \mathit{tree}\ x$. In the rule 16 respectively 17, the typing constraint $\mathit{num}\ z$, respectively $\mathit{tree}\ z$ is not in $\alpha^c \wedge \alpha^p$ and is a consequence of $\alpha^c \wedge \alpha^p$. In the rule 18, z does not have typing constraints in $\alpha^c \wedge \alpha^p$ and neither $\mathit{num}\ z$ nor $\mathit{tree}\ z$ is a consequence of $\alpha^c \wedge \alpha^p$.

In the rule 19, $\alpha^c \wedge \alpha^p$ is a block. In the rule 20, the numeric section of $\alpha^c \wedge \alpha^p$ is inconsistent. In the rule 21, the unreachable variables in $\bar{x}$ are renamed if necessary such that $u \succ v$ for each unreachable variable u and each reachable variable v in $\bar{x}$ and the numeric section of $\alpha^c \wedge \alpha^p$ is consistent. The consistency

Fig. 1. The rewriting rules

$$
\begin{array}{lll}
1 & \neg^1(\exists \bar{u}\, num\, x \wedge tree\, x \wedge \alpha \wedge \varphi) & \Longrightarrow true \\
2 & \neg^1(\exists \bar{u}\, x = f\bar{y} \wedge x = g\bar{z} \wedge tree\, x \wedge \alpha \wedge \varphi) & \Longrightarrow true \\
3 & \neg^1(\exists \bar{u}\, x = x \wedge \alpha \wedge \varphi) & \Longrightarrow \neg^1(\exists \bar{u}\, \alpha \wedge \varphi) \\
4 & \neg^1(\exists \bar{u}\, y = x \wedge tree\, x \wedge \alpha \wedge \varphi) & \Longrightarrow \neg^1(\exists \bar{u}\, x = y \wedge tree\, x \wedge \alpha \wedge \varphi) \\
5 & \neg^1 \left[\begin{array}{l} \exists \bar{u}\, x = fy_1 ... y_n \wedge x = fz_1 ... z_n \wedge \\ tree\, x \wedge \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^1 \left[\begin{array}{l} \exists \bar{u}\, x = fy_1 ... y_n \wedge \bigwedge_i y_i = z_i \wedge \\ tree\, x \wedge \alpha \wedge \varphi \end{array}\right] \\
6 & \neg^1 \left[\begin{array}{l} \exists \bar{u}\, x = y \wedge x = fz_1 ... z_n \wedge \\ tree\, x \wedge tree\, y \wedge \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^1 \left[\begin{array}{l} \exists \bar{u}\, x = y \wedge y = fz_1 ... z_n \wedge \\ tree\, x \wedge tree\, y \wedge \alpha \wedge \varphi \end{array}\right] \\
7 & \neg^1(\exists \bar{u}\, x = y \wedge x = z \wedge tree\, x \wedge \alpha \wedge \varphi) & \Longrightarrow \neg^1(\exists \bar{u}\, x = y \wedge y = z \wedge tree\, x \wedge \alpha \wedge \varphi)
\end{array}
$$

$$
\begin{array}{lll}
8 & \neg^4(\exists \bar{u}\, 0 = 0 \wedge \alpha \wedge \varphi) & \Longrightarrow \neg^4(\exists \bar{u}\, \alpha \wedge \varphi) \\
9 & \neg^4(\exists \bar{u}\, 0 < a_0 1 \wedge \alpha \wedge \varphi) & \Longrightarrow \neg^4(\exists \bar{u}\, \alpha \wedge \varphi) \\
10 & \neg^4 \left[\begin{array}{l} \exists \bar{u}\, x = y \wedge \\ num\, x \wedge num\, y \wedge \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^4 \left[\begin{array}{l} \exists \bar{u}\, x + (-1y) = 0 \wedge \\ num\, x \wedge num\, y \wedge \alpha \wedge \varphi \end{array}\right] \\
11 & \neg^4 \left[\begin{array}{l} \exists \bar{u}\, x = -y \wedge \\ num\, x \wedge num\, y \wedge \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^4 \left[\begin{array}{l} \exists \bar{u}\, x + y = 0 \wedge \\ num\, x \wedge num\, y \wedge \alpha \wedge \varphi \end{array}\right] \\
12 & \neg^4 \left[\begin{array}{l} \exists \bar{u}\, x = y + z \wedge num\, x \wedge \\ num\, y \wedge num\, z \wedge \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^4 \left[\begin{array}{l} \exists \bar{u}\, x + (-1y) + (-1z) = 0 \wedge \\ num\, x \wedge num\, y \wedge num\, z \wedge \alpha \wedge \varphi \end{array}\right] \\
13 & \neg^4 \left[\begin{array}{l} \exists \bar{u}\, \Sigma_{i=1}^n a_i x_i = a_0 1 \wedge \\ \Sigma_{i=1}^n b_i x_i = b_0 1 \wedge \\ \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^4 \left[\begin{array}{l} \exists \bar{u}\, \Sigma_{i=1}^n a_i x_i = a_0 1 \wedge \\ \Sigma_{i=1}^n (b_k a_i - a_k b_i) x_i = (b_k a_0 - a_k b_0) 1 \wedge \\ \alpha \wedge \varphi \end{array}\right] \\
14 & \neg^4 \left[\begin{array}{l} \exists \bar{u}\, \Sigma_{i=1}^n a_i x_i = a_0 1 \wedge \\ \Sigma_{i=1}^n b_i x_i < b_0 1 \wedge \\ \alpha \wedge \varphi \end{array}\right] & \Longrightarrow \neg^4 \left[\begin{array}{l} \exists \bar{u}\, \Sigma_{i=1}^n a_i x_i = a_0 1 \wedge \\ \Sigma_{i=1}^n \lambda (b_k a_i - a_k b_i) x_i < (b_k a_0 - a_k b_0) 1 \wedge \\ \alpha \wedge \varphi \end{array}\right]
\end{array}
$$

$$
\begin{array}{lll}
15 & \neg^1(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \neg^2(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) \\
16 & \neg^2(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \neg^1(\exists \bar{x}\, num\, z \wedge \alpha^c \wedge \alpha^p \wedge \varphi) \\
17 & \neg^2(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \neg^1(\exists \bar{x}\, tree\, z \wedge \alpha^c \wedge \alpha^p \wedge \varphi) \\
18 & \neg^2(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \left[\begin{array}{l} \neg^1(\exists \bar{x}\, num\, z \wedge \alpha^c \wedge \alpha^p \wedge \varphi) \wedge \\ \neg^1(\exists \bar{x}\, tree\, z \wedge \alpha^c \wedge \alpha^p \wedge \varphi) \end{array}\right] \\
19 & \neg^2(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \neg^3(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) \\
20 & \neg^3(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow true \\
21 & \neg^3(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \neg^4(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) \\
22 & \neg^4(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi) & \Longrightarrow \neg^5(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi)
\end{array}
$$

$$
\begin{array}{lll}
23 & \neg^7 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \\ \neg^5(\exists \bar{y}\, \beta^c \wedge \beta^p \wedge \psi) \end{array}\right] & \Longrightarrow \neg^7 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \\ \neg^6(\exists \bar{y}\, \gamma^c \wedge \gamma^p \wedge \psi) \end{array}\right] \\
24 & \neg^6 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \\ \bigwedge_i \neg^0(\exists \bar{y}_i \beta_i^c \wedge \beta_i^p \wedge \varphi_i) \end{array}\right] & \Longrightarrow \neg^7 \left[\begin{array}{l} \exists \bar{x}^1 \bar{x}^2\, \alpha^{c1} \wedge \alpha^{c2} \wedge \alpha^p \wedge \\ \bigwedge_i \neg^1(\exists \bar{y}_i \bar{x}^3 \gamma_i^c \wedge \gamma_i^p \wedge \varphi_i) \end{array}\right] \\
25 & \neg^7 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \\ \bigwedge_{i \in I} \neg^9(\exists \bar{y}_i \beta_i^c \wedge \beta_i^p) \end{array}\right] & \Longrightarrow \neg^8 \left[\begin{array}{l} \exists \bar{x}^1 \alpha^{c1} \wedge \alpha^{c2*} \wedge \alpha^p \wedge \\ \bigwedge_{i \in I'} \neg^9(\exists \bar{y}_i \beta_i^c \wedge \beta_i^{p*}) \end{array}\right] \\
26 & \neg^7 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \\ \neg^8(\exists \bar{y} \beta^c \wedge \beta^p) \end{array}\right] & \Longrightarrow \left[\begin{array}{l} \neg^7(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \neg^9(\exists \bar{y} \beta^{c1} \wedge \beta^p)) \wedge \\ \bigwedge_{i \in I} \neg^1(\exists \bar{x} \bar{y} \beta^p \wedge \beta^{c1} \wedge \beta_i^{c2*} \wedge \varphi_0) \end{array}\right] \\
27 & \neg^7 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \\ \neg^9(\exists \varepsilon true \wedge \beta^p) \end{array}\right] & \Longrightarrow true \\
28 & \neg^7 \left[\begin{array}{l} \exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \\ \neg^8 \left[\begin{array}{l} \exists \bar{y}\, \beta^c \wedge \beta^p \wedge \\ \bigwedge_{i \in I} \neg^9(\exists \bar{z}_i\, \gamma_i^c \wedge \gamma_i^p) \end{array}\right] \end{array}\right] & \Longrightarrow \left[\begin{array}{l} \neg^7(\exists \bar{x}\, \alpha^c \wedge \alpha^p \wedge \varphi \wedge \neg^8(\exists \bar{y}\, \beta^c \wedge \beta^p)) \wedge \\ \bigwedge_{i \in I} \neg^6(\exists \bar{x} \bar{y} \bar{z}_i\, \delta_i^c \wedge \delta_i^p \wedge \varphi_0) \end{array}\right]
\end{array}
$$

can be verified for example by using the first step of the Simplex. In the rule 22, $\alpha^c \wedge \alpha^p$ is a solved block.

In the rule 23, γ^c is obtained from β^c as follows: for every variable $x \in var(\beta^c)$, we add all the relations *num x* or *tree x* which are in β^p but not in β^c, and for all the variables y which do not occur in an equation or inequation of β^c we remove all relations *num y* or *tree y* which are both in β^c and β^p. The formula γ^p is the formula $\alpha^p \wedge \alpha^c$.

In the rule 24, $\exists \bar{x} \alpha^c$ is decomposed into $\exists \bar{x}^1 \alpha^{c1} \wedge (\exists \bar{x}^2 \alpha^{c2} \wedge (\exists \bar{x}^3 \alpha^{c3}))$, $\gamma_i^c = \beta_i^c \wedge \alpha^{c3}$ and $\gamma_i^p = \beta_i^p \wedge \alpha^{c1} \wedge \alpha^{c2} \wedge \alpha^p$.

The four rules 25, 26, 27 and 28 cannot be applied on the occurrence of $\neg^7$ of the first level of the general working formula. In the rule 25, all the $\beta_c i$ are different from *true*, I' is the set of $i \in I$ such that β_i^c does not contain occurrences of any variables in $\bar{x}^2$. The formula α^{c2*} is such that $T \models (\exists \bar{x}^2 \alpha^{c2}) \leftrightarrow \alpha^{c2*}$ and is computed using the Fourier quantifier elimination. The propagated-constraint section $\beta_i^{p*} = \alpha^{c1} \wedge \alpha^{c2*} \wedge \alpha^p$.

In the rule 26, φ is such that every negation symbol $\neg^k$ has $k \geq 6$, φ_0 is obtained from φ by replacing all occurrences of $\neg^k$ by $\neg^0$ and all propagated-constraint sections by *true*. Let β^2 be the formula obtained from β^{c2} by removing the multiple occurrences of typing constraints and for all the variables y which do not occur in an inequation of β^{c2} we remove all relations *num y* or *tree y* which are both in β^{c1} and β^{c2}. If β^2 is the formula *true* then $I = \emptyset$, otherwise the β_i^{c2*} with $i \in I$ are obtained from β^2 as follows: Since $\beta^2 \in A^2$ then it is of the form

$$\begin{bmatrix} (\bigwedge_{\ell \in L} num\ z_\ell) \wedge (\bigwedge_{k \in K} tree\ v_k) \wedge \\ ((\bigwedge_{j \in J} \sum_{i=1}^n a_{ij} x_i < a_{0j}) \wedge \bigwedge_{m=1}^n num\ x_m) \end{bmatrix},$$

thus $\neg\beta^2$ is of the form

$$\begin{bmatrix} (\bigvee_{\ell \in L} tree\ z_\ell) \vee (\bigvee_{k \in K} num\ v_k) \vee (\bigvee_{m=1}^n tree\ x_m) \vee \\ \bigvee_{j \in J} ((\sum_{i=1}^n a_{ij} x_i = a_{0j} 1 \wedge \bigwedge_{m=1}^n num\ x_m) \vee \\ (\sum_{i=1}^n (-a_{ij}) x_i < (-a_{0j}) 1 \wedge \bigwedge_{m=1}^n num\ x_m)) \end{bmatrix}$$

Each element of this disjunction is a block and represents a formula β_i^{c2*}. Of course we have $T \models (\neg\beta^2) \leftrightarrow \bigvee_i \beta_i^{c2*}$.

In the rule 28, $I \neq \emptyset$, φ is such that every negation symbol $\neg^k$ has $k \geq 6$, φ_0 is obtained from φ by replacing all occurrences of $\neg^k$ by $\neg^0$ and all propagated-constraint sections by *true*. Moreover $\delta_i^p = \alpha^p$ and $\delta_i^c = \gamma_i^c \wedge \beta^c \wedge \alpha^c$.

Property 4.3.1 *Every repeated application of the precedent rewriting rules on an inital working formula terminates and produces an equivalent final working formula which does not contain new free variables.*

Corollary 4.3.2 *Every formula is equivalent in T either to true or to false or to a disjunction of general solved formulas, having at least one free variable, being equivalent neither to true nor to false in T and where the solutions of the free variables are expressed in a clear and explicit way.*

In fact, solving a constraint φ in T proceeds as follows:

1. Transform φ into a normalized formula, then into an initial working formula ϕ, which is equivalent to φ in T.
2. Transform ϕ into a final working formula ψ using the rewriting rules defined in the subsection 4.3.
3. Extract from ψ the equivalent disjunction of general solved formulas. If this disjunction contains the general solved formula *true*, then it is reduced to *true*.

Example: Let φ be the following constraint having i, j as free variables:

$$\exists x\, x = fij \wedge i > 0 \wedge tree\, x \wedge num\, i \wedge num\, j \wedge \neg(\exists k\, j = 2k \wedge num\, k).$$

We can see that $num\, j \wedge \neg(\exists k\, j = 2k \wedge num\, k)$ is always false in T since for every variable j, there exists a unique variable k such that $j = 2k$ (axiom 13_n). Let us transform φ into an initial working formula (the propagated-constraint sections are underlined):

$$\neg^6\neg^0(\exists x\, x = fij \wedge i > 0 \wedge tree\, x \wedge num\, j \wedge \underline{true} \wedge \neg^0(\exists k\, j = 2k \wedge num\, k \wedge \underline{true}))$$

After having applied the rules 24, 15, 16, 15, 19, 21, 22, 23 in this order, we obtain:
$\neg^7\neg^6(\exists x\, x = fij \wedge i > 0 \wedge tree\, x \wedge num\, i \wedge num\, j \wedge \underline{true} \wedge \neg^0(\exists k\, j = 2k \wedge num\, k \wedge \underline{true}))$
The rule 24 being applied changes the formula to:

$$\neg^7\neg^7 \left[\begin{array}{l} i > 0 \wedge num\, i \wedge num\, j \wedge \underline{true} \wedge \\ \neg^1 \left[\begin{array}{l} \exists xk\, x = fij \wedge j = 2k \wedge num\, k \wedge tree\, x \wedge \\ \underline{i > 0 \wedge num\, i \wedge num\, j} \end{array}\right] \end{array}\right]$$

After having applied on the sub-working formula $\neg^1(...)$ the rule 15, 19, 21, 12, 22, 23

$$\neg^7\neg^7 \left[\begin{array}{l} i > 0 \wedge num\, i \wedge num\, j \wedge \underline{true} \wedge \\ \neg^6 \left[\begin{array}{l} \exists xk\, x = fij \wedge j - 2k = 0 \wedge num\, k \wedge tree\, x \wedge \\ \underline{i > 0 \wedge num\, i \wedge num\, j} \end{array}\right] \end{array}\right]$$

The rule 24 is applied then we obtain:

$$\neg^7\neg^7(i > 0 \wedge num\, i \wedge num\, j \wedge \underline{true} \wedge \neg^7(true \wedge \underline{i > 0 \wedge num\, i \wedge num\, j}))$$

The rules 25, 26 are applied in this order, giving:

$$\neg^7\neg^7(i > 0 \wedge num\, i \wedge num\, j \wedge \underline{true} \wedge \neg^9(true \wedge \underline{i > 0 \wedge num\, i \wedge num\, j}))$$

Finally, by application of the rule 27, we obtain the final working formula $\neg^7 true$, which is equivalent to the empty disjunction of general solved formulas, i.e. *false*. Thus, the initial constraint φ is false in T and does not depend on the values of its free variables i and j. A such phenomena is impossible to detect using a decision procedure instead of a first-order constraint solver.

5 Conclusion

Quantified formulas over trees and rational numbers provide an expressive constraint language that is essential in applications such as program analysis and model checking. We have presented in this paper a first-order constraint solver in the theory of the evaluated trees. The algorithm is given in the form of 28 rewriting rules and its correctness implies the completeness of a theory built on the model of Prolog III. Our aim in this work was not only to decide proposition i.e. to decide if a formula without free variables is true or false in T but to

express the solutions of any first-order constraint having free variables in a clear and explicit way.

S. Vorobyov [25] has shown that the problem of deciding if a proposition is true or not in the theory of trees is non-elementary, i.e. the complexity of all algorithms which solve it cannot be bound by a tower of powers of $2's$ (with a top down evaluation) with a fixed height. Thus, our algorithm must not escape this kind of complexity in the worst case. This is why we have used two strategies in the algorithm: a top down propagation of constraints and a bottom-up elimination of quantifiers and distribution. This technique can quickly detect (using propagation and local solving) sub-formulas which are equivalent to false and prevents us from solving a big working formula (i.e. a working formula of huge depth) which contradicts its top-working formula. We have recently programmed a similar algorithm only on the theory of finite or infinite trees and in spite of the high complexity we can solve formulas on two partners games involving 160 nested quantifiers [10].

Currently, we are trying to find other classes of theories T_i such that we can apply a similar technique to solve first-order constraints in the hybrid theories $T_i + Trees$. We are also working on a possibly CHR (Constraint Handling Rules) implementation of our solver.

Acknowledgements We thank Alain Colmerauer for our many discussions and his help in this work. We dedicate to him this paper.

References

1. Benhamou F, Colmerauer , Van Caneghem M. Le manuel de Prolog IV , PrologIA, France, 1996.
2. Bürckert H. Solving disequations in equational theories. In Proc. 9th Conf. on Automated Deduction, LNCS 310, p 517-526. 1988.
3. Colmerauer A. An introduction to Prolog III. *Communication of the ACM*, 33(7):68–90,1990.
4. Colmerauer A. Equations and disequations on finite and infinite trees. In Proc of the 5th conf on generation of computer systems Tokyo, 1984. P. 85–99.
5. Colmerauer, A. 1982. Prolog and infinite trees. In K.L. Clark and S-A. Tarnlund, editors, Logic Programming. Academic Press. pp. 231–251.
6. Comon H. Résolution de contraintes dans des algèbres de termes. Rapport d'Habilitation, Université de Paris Sud, 1992.
7. Courcelle B. Fundamental Properties of Infinite Trees, TCS vol. 25, no 2, 1983, pp. 95–169.
8. Dao, T. and Djelloul, K. Solving first-order constraints in evaluated trees. Technical report, Laboratoire d'informatique fondamentale d'Orlans, RR-2006-05, http://www.univ-orleans.fr/lifo/rapports.php. Full version of this paper with full proofs.
9. Djelloul, K. About the combination of trees and rational numbers in a complete first-order theory. 2005. Proceeding of the 5th International workshop on frontiers of combining systems (FroCoS'05). Lecture Notes in Artificial Intelligence, LNAI, vol 3717, pp. 106–122.
10. Djelloul, K. and Dao, T. 2006. Solving First-Order formulas in the Theory of Finite or Infinite Trees. Proceeding of the 21st ACM Symposium on Applied Computing (SAC'06). ACM press, pp. 7–14.

11. Djelloul, K. and Dao, T. 2006. Extension into trees of first order theories. Proceeding of the 8th International conference on artificial intelligence and symbolic computation (AISC'06). Lecture notes in artificial intelligence. LNAI, vol 4120, pp. 53–67.
12. Djelloul, K. 2006. Decomposable Theories. Journal of Theory and practice of Logic Programming. (to appear)
13. Huet G. Resolution d'equations dans les langages d'ordre 1, 2,... ω. These d'Etat, Universite Paris 7. France,1976.
14. Jaffar J. Efficient unification over infinite terms. New Generation Computing, 2(3):207-219, 1984.
15. Jouannaud, J.P. and Kirchner, C. 1991. Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. Computational Logic - Essays in Honor of Alan Robinson, MIT press, pp: 257-321.
16. Lassez, J., Maher, M. and Marriott, K. 1986. Unification revisited. In proceedings of the workshop on the foundations of deductive database and logic programming, pp. 587-625.
17. Lassez, J. and Marriott, K. 1987. Explicit representation of terms defined by counter examples. Journal of automated reasonning. 3:301-317.
18. Lassez, J. and McAloon, K. 1989. Independence of negative constraints. In proceedings of TOPSOFT, LNCS 351, pp. 19-27.
19. Maher M. Complete axiomatization of the algebra of finite, rational and infinite trees. *Technical report, IBM - T.J.Watson Research Center*, 1988.
20. Maher, M. and Stuckey, P. 1995. On inductive inference of cyclic structures. Annals of mathematics and artificial intelligence, 15(2):167-208.
21. Martelli A. and Montanari U. An efficient unification algorithm. ACM Trans. on Languages and Systems, 4(2):258-282, 1982.
22. Paterson M and Wegman N. Linear unification. Journal of Computer and Systems Science, 16:158-167, 1978.
23. Ramachandran V. and Van Hentenryck P. Incremental algorithms for constraint solving and entailment over rational trees. In Proc. of the 13th Conf. Foundations of Software Technology. LNCS 761, 205-217, 1993.
24. Robinson J.A. A machine-oriented logic based on the resolution principle. JACM, 12(1):23-41, 1965.
25. Vorobyov S. An Improved Lower Bound for the Elementary Theories of Trees, Proceeding of the 13th Conf on Automated Deduction (CADE'96). LNAI 1104, pp. 275- 287, 1996.