Theory of Finite or Infinite Trees Revisited

KHALIL DJELLOUL
Faculty of computer science
University of Ulm
Germany

THI-BICH-HANH DAO
Laboratoire d'informatique fondamentale d'Orleans
Universite d'Orleans
France

THOM FRÜHWIRTH
Faculty of computer science
University of Ulm
Germany

Abstract

We present in this paper a first-order axiomatization of an extended theory T of finite or infinite trees, built on a signature containing an infinite set of function symbols and a relation *finite*(t) which enables to distinguish between finite or infinite trees. We show that T has at least one model and prove its completeness by giving 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. The solver is given in the form of 16 rewriting rules which transform any first-order constraint φ into an equivalent disjunction ϕ of simple formulas such that ϕ is either the formula *true* or the formula *false* or a formula having at least one free variable, being equivalent neither to *true* nor to *false* and where the solutions of the free variables are expressed in a clear and explicit way. The correctness of our rules implies the completeness of T. We also describe an implementation of our algorithm in CHR (Constraint Handling Rules) and compare the performance with an implementation in C++ and that of a recent decision procedure for decomposable theories.

KEYWORDS: Logical first-order formula, Theory of finite or infinite trees, Complete theory, Rewriting rules.

Contents

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 1930, J. Herbrand (Herbrand 1930) gave an informal description of an algorithm for unifying finite terms, that is solving equations in finite trees. A. Robinson (Robinson 1965) rediscovered a similar algorithm when he introduced the resolution procedure for first-order logic in 1965. Some algorithms with better complexities have been proposed after by M.S. Paterson and M.N.Wegman (Paterson and Wegman 1978) and A. Martelli and U. Montanari (Martelli and Montanari 1982). A good synthesis on this field can be found in the paper of J.P. Jouannaud and C. Kirchner (Jouannaud and Kirchner 1991). Solving conjunctions of equations on infinite trees has been studied by G. Huet (Huet 1976), by A. Colmerauer (Colmerauer 1982) and by J. Jaffar (Jaffar 1984). Solving conjunctions of equations and disequations on finite or infinite trees has been studied by H.J. Burckert (Burkert 1988) and A. Colmerauer (Colmerauer 1984). An incremental algorithm for solving conjunctions of equations and disequations on rational trees has then been proposed by V.Ramachandran and P. Van Hentenryck (Ramachandran and Van Hentenryck 1993) and a quasi-linear incremental algorithm for testing entailment and disentailment over rational trees has been given by A. Podelski and P. Van Roy (Podelski and Van Roy 1994).

On the other hand, K.L. Clark has proposed a complete axiomatization of the equality theory, also called Clark equational theory CET, and gave intuitions about a complete axiomatization of the theory of finite trees (Clark 1978). B. Courcelle has studied the properties of infinite trees in the scope of recursive program schemes (Courcelle 1983; Courcelle 1986) and A. Colmerauer has described the execution of Prolog II, III and IV programs in terms of solving equations and disequations in the algebra of finite or infinite trees (Colmerauer 1984; Colmerauer 1990; Benhamou et al. 1996).

Concerning quantified constraints, solving universally quantified disequations on finite trees has been studied by D.A. Smith (Smith 1991) and there exist some decision procedures which transform any first-order formula into a Boolean combination of quantified conjunctions of atomic formulas using elimination of quantifiers. In the case of finite trees we can refer to A. Malcev (Malcev 1971), K. Kunen (Kunen 1987) and H. Comon (Comon 1988; Comon 1991b; Comon and Lescanne 1989). For infinite trees, we can refer to the work of H. Comon (Comon 1988; Comon 1991a) and M. Maher (Maher 1988).

M. Maher has axiomatized all the cases by complete first-order theories (Maher 1988). In particular, he has introduced the theory $\mathcal{T}$ of finite or infinite trees built on an infinite set F of function symbols and showed its completeness using a decision procedure which transforms any first-order formula φ into a Boolean combination ϕ of quantified conjunctions of atomic formulas. If φ does not contain free variables then ϕ is either the formula *true* or *false*.

K. Djelloul has then presented in (Djelloul 2006a) the class of decomposable theories and proved that the theory of finite or infinite trees is decomposable. He has also given a decision procedure in the form of five rewriting rules which, for any decomposable theory, transforms any first-order formula φ into an equivalent conjunction ϕ of solved formulas easily transformable into a Boolean combination of existentially quantified conjunctions of atomic formulas. In particular, if φ has no free variables then ϕ is either the formula *true* or $\neg$*true*.

Unfortunately, all the preceding decision procedures are not able to solve complex first-order constraint satisfaction problems in $\mathcal{T}$. In fact, these algorithms are only basic decision procedures and not full first-order constraint solvers: they do 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 $\mathcal{T}$. The appropriate solved formula of φ in this case should be the formula *false* or the formula *true* instead of ϕ. If we use for example the decision procedure of (Djelloul 2006a) to solve the following formula φ

$$\neg(\exists y\, x = f(y) \wedge \neg(\exists zw\, x = f(z) \wedge w = f(w))),$$

then we get the following solved[1] formula ϕ

$$\neg(\exists y\, x = f(y) \wedge \neg(\exists z\, x = f(z))).$$

The problem is that this formula contains free variables but is always true in the theory of finite or infinite trees. In fact, it is equivalent to

$$\neg(\exists y\, x = f(y) \wedge \neg(\exists z\, x = f(y) \wedge x = f(z))),$$

[1] ϕ is solved according to Definition 4.2.4 of (Djelloul 2006a)

i.e. to

$$\neg(\exists y\, x = f(y) \wedge \neg(x = f(y) \wedge (\exists z\, z = y))),$$

thus to

$$\neg(\exists y\, x = f(y) \wedge \neg(x = f(y))),$$

which is finally equivalent to *true*. As a consequence, the solved formula of φ should be *true* instead of ϕ. This is a good example which shows the limits of the decision procedures in solving first-order constraints having at least one free variable.

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 the theory of finite or infinite. 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 first-order formula ϕ 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 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 ϕ are called *first-order constraint solvers.*

We have then presented in (Djelloul and Dao 2006b), not only a decision procedure, but a full first-order constraint solver in the theory $\mathcal{T}$ of finite or infinite trees, in the form of 11 rewriting rules, which gives clear and explicit solutions for any first-order constraint satisfaction problem in $\mathcal{T}$. The intuitions behind this algorithm come from the works of T. Dao in (Dao 2000) where many elegant properties of the theory of finite or infinite trees were given. As far as we know, this is the first algorithm which is able to do a such work in $\mathcal{T}$.

This is an extended and detailed version with full proofs of our previous work on the theory $\mathcal{T}$ of finite or infinite trees (Djelloul and Dao 2006b). Moreover, in this paper we extend the signature of $\mathcal{T}$ by the relation *finite*(t) which forces the term t to be a finite tree. Then we extend Maher's axiomatization by two new axioms and show its completeness by giving an extended version of our previous first-order constraint solver (Djelloul and Dao 2006b). We also describe a CHR (Constraint Handling Rules) implementation of our rules and compare the performances with those obtained using a C++ implementation of our solver and the decision procedure for decomposable theories (Djelloul 2006a).

Overview of the paper

This paper is organized in five sections followed by a conclusion. This introduction is the first section. In section 2, we introduce the structure of finite or infinite trees and give formal definitions of trees, finites trees, infinite trees and rational trees. We end this section by presenting particular algebras which handle finite or infinite trees.

In section 3, after a brief recall on first-order logic, we present the five axioms of our extended theory[2] T of finite or infinite trees built on a signature containing not only an infinite set of function symbols, but also a relation *finite*(t) which enables to distinguish between finite or infinite trees. We then extend the algebras given at the end of section 2 by the relation *finite*(t) and show that these extended algebras are models of T. In particular, we show that the models of sets of nodes, of finite or infinite trees and of rational trees are models of T.

In section 4, we present structured formulas that we call *working formulas* and give some of their properties. These working formulas are extensions of those given in (Djelloul 2006a). We also introduce the notion of reachable variables and show that there exist particular formulas which have only quantified reachable variables, do not accept elimination of quantifiers and cannot be simplified any further. Such formulas are called *general solved formulas*. We then present 16 rewriting rules which handle working formulas and transform an initial working formula into an equivalent conjunction of final working formulas from which we can extract easily an equivalent conjunction of general solved formulas. We end this section by a full first-order constraint solver in T. This algorithm uses, among other things, our 16 rules and transforms any first-order formula φ into a disjunction ϕ of simple formulas such that ϕ is either the formula *true* or the formula *false* or a formula having at least one free variable, being equivalent neither to *true* nor to *false* and where the solutions of the free variables are expressed in a clear and explicit way. The correctness of our algorithm implies the completeness of T.

Finally, in section 5, we give a series of benchmarks. Our algorithm was implemented in C++ and CHR (Fruehwirth 1998; Fruehwirth and Abdennadher 2003; Schrijvers and Fruehwirth 2006). The C++ implementation is able to solve formulas of a two player game involving 80 nested alternated quantifiers. Even if the C++ implementation is fastest, we found interesting to see how we can translate our algorithm into CHR rules. Using this high-level approach, we will be able to quickly prototype optimizations and variations of our algorithm and hope to parallelize it. We also compare the performances with those of C++ implementation of the decision procedure for decomposable theories[3] (Djelloul 2006a).

The axiomatization of T, the proof that T has at least one model, the 16 rewriting rules, the proof of the correctness of our rules, the first-order constraint solver in T, the completeness of T, the CHR implementation, the two player game and the benchmarks are new contributions in this paper.

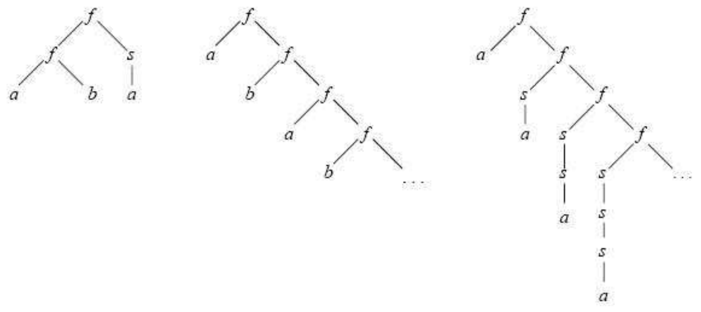
2 The structure of finite or infinite trees

2.1 What is a tree?

Trees are well known objects in the computer science world. Here are some of them:

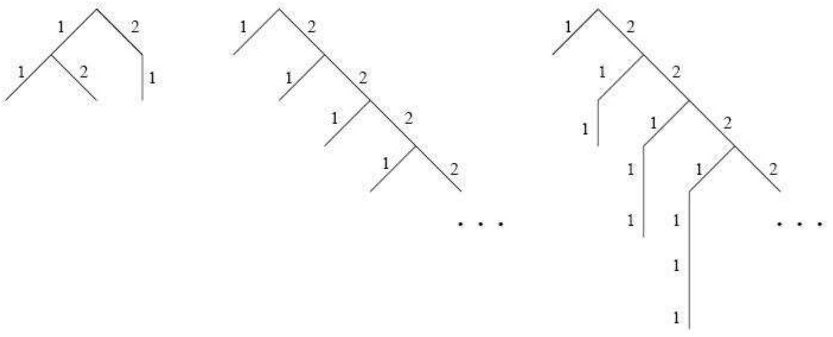
[2] We have chosen to denote by $\mathcal{T}$ the Maher's theory of finite or infinite trees and by T our extended theory of finite or infinite trees.

[3] In (Djelloul 2006a), we have shown that the Maher's theory $\mathcal{T}$ of finite or infinite trees is decomposable. We can show easily using a similar proof that our extended theory T is also decomposable.



their nodes are labeled by the symbols a,b,f,s of respective arities 0,0,2,1. While the first tree is a *finite tree*, i.e. it has a finite set of nodes, the two others are *infinite trees*, i.e. they have an infinite set of nodes.

Let us now number from 1 to n and from left to right the branches that connect each node l to his n sons. We get:



Each node c labeled by l can now be seen as a pair (p, l) where p is the position of the node, i.e. the smallest series of positive integers that we meet if we move from the root of the tree to the node c. Thus, the preceding trees can be represented by the following sets of nodes:

$$\{(\varepsilon, f), (1, f), (2, s), (11, a), (12, b), (21, a)\}$$
$$\{(\varepsilon, f), (1, a), (2, f), (21, b), (22, f), (221, a), (222, f), (2221, b), ...\}$$
$$\left\{ \begin{array}{c} (\varepsilon, f), (1, a), (2, f), (21, s), (22, f), (211, a), (221, s), (222, f), \\ (2211, s), (2221, s), (2222, f), (22111, a), (22211, s), (222111, s), (2221111, a), ... \end{array} \right\}$$

Let us now formalize all the preceding statements. Let L be a (possibly infinite) set. Its elements are called *labels.* To each label $l \in L$ is linked a non-negative integer called *arity of* l. An n-ary label is a label of arity n. A *position* is a word built on strictly positive integers (the empty word is denoted by ε). Let p be a position and l a label. The pair (p, l) is called *node* and its *depth* is the length[4] of p. An n-ary node is a node whose label is of arity n. A *root* is a node of depth 0. The *row* of an n-ary node, with $n \neq 0$, is the last integer of its position. We say that c is the father of c' or c' is the son of c if c and c' are nodes whose positions are respectively of the form $i_1...i_k$ and $i_1...i_k i_{k+1}$, where the i_j's are strictly positive integers and k

[4] As usual, the length of the empty word ε is 0.

a (possibly null[5]) positive integer. Let us denote by N the set of the nodes labeled by elements of L.

Definition 2.1.1
A node c of N is called *arborescent* in a sub-set N_1 of N if $N_1 \neq \emptyset$ and either $c \notin N_1$, or $c \in N_1$ and the two following conditions hold:

- $N_1 - \{c\}$ does not contain any node whose position is the same than those of c,
- c is either a root or the son of an n-ary node of N_1 which has exactly n sons in N_1 of respective rows $1, ..., n$.

We can now define formally a *tree*:

Definition 2.1.2
A *tree* tr is a sub-set of N such that each element of N is arborescent in tr. A *finite* tree is a tree whose set of nodes is finite. An *infinite* tree is a tree whose set of nodes is infinite.

Let us now define the notion of subtree:

Definition 2.1.3
Let tr be a tree. The subtree linked to a node $(i_1...i_k, l)$ of tr is the set of the nodes of the form $(i_{k+1}...i_{k+n}, l')$ with $(i_1...i_{k+n}, l') \in tr$ and[6] $n \geq 0$. We call *subtree of* tr a subtree linked to one of the nodes of tr. A subtree of tr of depth k is a subtree linked to a node of tr of depth k.

From Definition 2.1.2, we deduce that each subtree of a tree tr is also a tree.

Definition 2.1.4
A rational tree is a tree whose set of subtrees is a finite set.

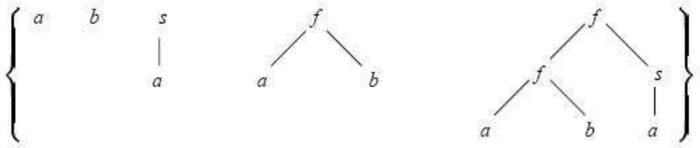
Note that an infinite tree can be rational. In fact, even if its set of nodes is infinite but n subtrees linked to n different nodes can be similar. Let us see this in the following example:

Example 2.1.5
Let us consider the three trees presented in the beginning of Section 2.1. Let us name them from left to righ by: tr_1, tr_2 and tr_3. The set of the subtrees of tr_1 is the following *finite* set:

$$\left\{\begin{array}{l} \{(\varepsilon, a)\}, \\ \{(\varepsilon, b)\}, \\ \{(\varepsilon, s), (1, a)\}, \\ \{(\varepsilon, f), (1, a), (2, b)\}, \\ \{(\varepsilon, f), (1, f), (2, s), (11, a), (12, b), (21, a)\} \end{array}\right\}$$

[5] Of course, for $k = 0$, $i_1...i_k$ is reduced to ε.
[6] Of course, for $n = 0$, $(i_{k+1}...i_{k+n}, l')$ is reduced to (ε, l').

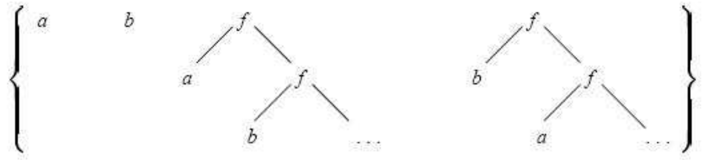
i.e.



The set of the subtrees of tr_2 is the following *finite* set:

$$\left\{\begin{array}{l}\{(\varepsilon,a)\},\\ \{(\varepsilon,b)\},\\ \{(\varepsilon,f),(1,a),(2,f),(21,b),(22,f),(221,a),...\},\\ \{(\varepsilon,f),(1,b),(2,f),(21,a),(22,f),(221,b),...\}\end{array}\right\}$$

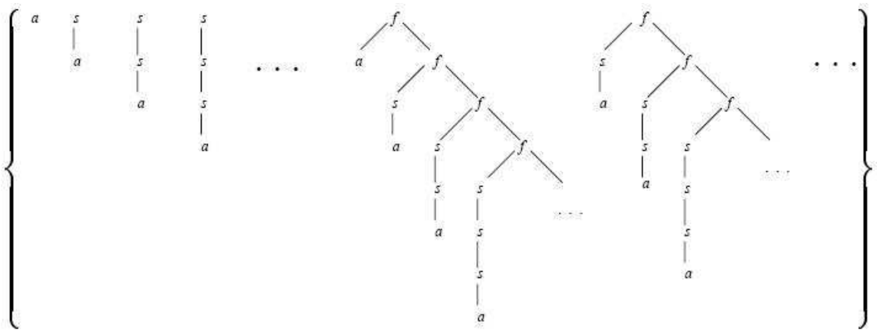
i.e.



The set of the subtrees of tr_3 is the following *infinite* set:

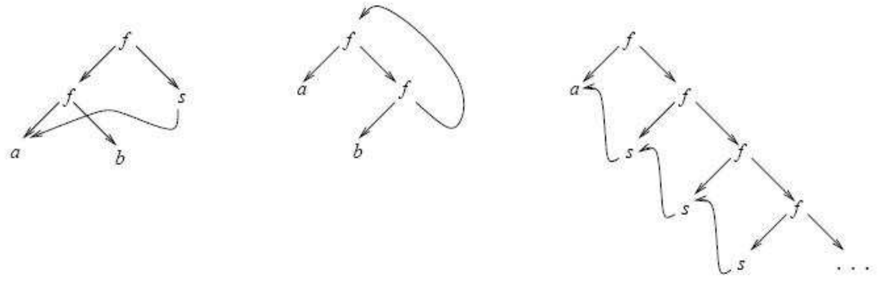
$$\left\{\begin{array}{l}\{(\varepsilon,a)\},\\ \{(\varepsilon,s),(1,a)\},\\ \{(\varepsilon,s),(1,s),(11,a)\},\\ \{(\varepsilon,s),(1,s),(11,s),(111,a)\},\\ ...\\ \{(\varepsilon,f),(1,a),(2,f),(21,s),(22,f),...\},\\ \{(\varepsilon,f),(1,s),(2,f),(11,a),(21,s),(22,f),...\}\\ ...\end{array}\right\}$$

i.e.

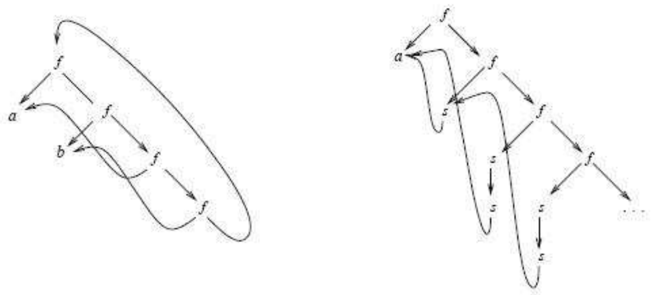


Note that the tree tr_1 has a finite set of nodes and a finite set of subtrees. Thus, it is a finite rational tree. The tree tr_2 has an infinite set of nodes but a finite set of subtrees. Thus, it is an infinite rational tree. The tree tr_3 has an infinite set of nodes and an infinite set of subtrees. Thus, it is an infinite non-rational tree.

Note also that a rational tree can always be represented by a *finite directed graph.* For that, it is enough to merge all the nodes whose linked subtrees are similar. A non-rational tree cannot be represented by a finite directed graph. In this case, only an *infinite directed graph* representation will be possible. For example, the trees tr_1, tr_2 and tr_3 can be represented as follows:

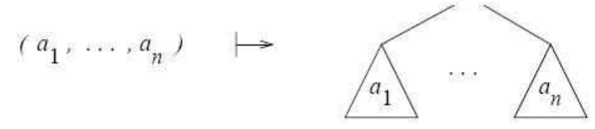


Of course, two different directed graphs can represent the same tree. For example the trees tr_2 and tr_3 can also be represented as follows:



2.2 Construction operations

We would like to provide the set Tr of finite or infinite trees with a set of *construction operations* ; one for each label l of L. These operations will be schematized as follows:



with n the arity of the label l. In order to formally define these construction operations, we need first to define them in the set D of sets of nodes[7] of N. Let i be a strictly positive integer. If $d = (j_1...j_k, l)$ is a node then we denote by $i.d$ the node $(ij_1...j_k, l)$. If a is a set of nodes (i.e. $a \in D$), then we denote by $i.a$ the set of nodes $\{i.d \mid d \in a\}$.

[7] In other words, each element of D is a set of nodes, i.e. a subset of N.

Definition 2.2.1
In the set D, the construction operation linked to the n-ary label l is the application $l^D : (a_1, ..., a_n) \mapsto \{(\varepsilon, l)\} \cup 1.a_1 \cup ... \cup n.a_n$ with $a_1 ... a_n$ elements of D.

Remark 2.2.2
Let a be an element of D. Let us denote by $\nu_k(a)$ the set of nodes of a of depth k. Many remarks must be stated concerning any elements a, a_i and b of D:

1. $a = b \leftrightarrow \bigwedge_{k=1}^{\infty} \nu_k(a) = \nu_k(b)$.
2. $\nu_0(l^D(a_1, ..., a_n)) = \{(\varepsilon, l)\}$.
3. For all $k \geq 0$, there exists a function φ_{k+1} which is independent from all the $\nu_{k+1}(a_i)$, with $i \in \{1, ..., n\}$, such that $\nu_{k+1}(l^D(a_1, ..., a_n)) = \varphi_{k+1}(\nu_k(a_1), ..., \nu_k(a_n))$.
4. The elements of $\nu_0(l^D(a_1, ..., a_n))$ are arborescent in $l^D(a_1, ..., a_n)$.
5. For all $k \geq 0$, the elements of $\nu_{k+1}(l^D(a_1, ..., a_n))$ are arborescent in $l^D(a_1, ..., a_n)$ if and only if, for each $i \in \{1, ..., n\}$, the elements of $\nu_k(a_i)$ are arborescent in a_i.
6. If for all $k \geq 0$ the elements of $\nu_k(l^D(a_1, ..., a_n))$ are arborescent in $l^D(a_1, ..., a_n)$ then each element of N is arborescent in $l^D(a_1, ..., a_n)$.

Let now F be an infinite set of function symbols. Let us denote by:

- N the set of the nodes labeled by F,
- D the set of sets of nodes of N,
- Tr the set of the elements of D which are trees,
- Ra the set of the elements of Tr which are rational,
- Fi the set of the elements of Tr which are finite.

If f is an n-ary function symbol taken from F then the operation of construction f^D associated to f is an application of the form $D^n \to D$. Let $tr_1, ..., tr_n$ be elements of Tr. From the fourth and fifth point of Remark 2.2.2 we deduce that $f^D(tr_1, ..., tr_n)$ is also a tree, i.e. an element of Tr. Thus, we can introduce the following application:

$$f^{Tr} : (tr_1, ..., tr_n) \mapsto f^D(tr_1, ..., tr_n) \text{ which is of type } Tr^n \to Tr.$$

On the other hand, the set of the subtrees of the tree $f^D(tr_1, ..., tr_n)$ is obtained by the union of the sets of the subtrees of all the tr_i plus the tree $f^D(tr_1, ..., tr_n)$. Thus, if all the tr_i's are rational trees then the tree $f^D(tr_1, ..., tr_n)$ is rational. As a consequence, we can introduce the following application:

$$f^{Ra} : (tr_1, ..., tr_n) \mapsto f^D(tr_1, ..., tr_n) \text{ which is of type } Ra^n \to Ra.$$

Finally, if all the tr_i's are finite trees, then the tree $f^D(tr_1, ..., tr_n)$ is finite. Thus, we can introduce the following application:

$$f^{Fi} : (tr_1, ..., tr_n) \mapsto f^D(tr_1, ..., tr_n) \text{ which is of type } Fi^n \to Fi.$$

The pairs $< D, (f^D)_{f \in F} >$, $< Tr, (f^{Tr})_{f \in F} >$, $< Fi, (f^{Fi})_{f \in F} >$ and $< Ra, (f^{Ra})_{f \in F} >$ are known as the *algebras* of sets of nodes, of finite or infinite trees, of finite trees and of rational trees.

3 The extended theory T of finite or infinite trees

3.1 Formal preliminaries

3.1.1 Formulas

We are given once and for all an infinite countable set V of *variables* and the set L of *logical* symbols:

$$=, \mathit{true}, \mathit{false}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists, (,).$$

We are also given once and for all a *signature* S, i.e. a set of symbols partitioned into two subsets: the set of *function* symbols and the set of *relation* symbols. To each element s of S is linked a non-negative integer called *arity* of s. An n-ary symbol is a symbol of arity n. A 0-ary function symbol is called *constant.*

As usual, an *expression* is a word on $L \cup S \cup V$ which is either a *term*, i.e. of one of the two forms:

$$x,\ f(t_1, \ldots, t_n), \tag{1}$$

or a *formula*, i.e. of one of the eleven forms:

$$\begin{gathered} s = t,\ r(t_1, \ldots, t_n),\ \mathit{true},\ \mathit{false}, \\ \neg\varphi,\ (\varphi \wedge \psi),\ (\varphi \vee \psi),\ (\varphi \rightarrow \psi),\ (\varphi \leftrightarrow \psi), \\ (\forall x\, \varphi),\ (\exists x\, \varphi). \end{gathered} \tag{2}$$

In (1), x is taken from V, f is an n-ary function symbol taken from S and the t_i's are shorter terms. In (2), s, t and the t_i's are terms, r is an n-ary relation symbol taken from S and φ and ψ are shorter formulas. The set of the expressions forms *a first-order language with equality.*

The formulas of the first line of (2) are known as *atomic*, and *flat* if they are of one of the following forms:

$$\mathit{true},\ \mathit{false},\ x_0 = x_1, x_0 = f(x_1, ..., x_n),\ r(x_1, ..., x_n),$$

where all the x_i's are (possibly non-distinct) variables taken from V, f is an n-ary function symbol taken from S and r is an n-ary relation symbol taken from S. An *equation* is a formula of the form $s = t$ with s and t terms.

An occurrence of a variable x in a formula is *bound* if it occurs in a sub-formula of the form $(\forall x\, \varphi)$ or $(\exists x\, \varphi)$. It is *free* in the contrary case. The *free variables of a formula* are those which have at least one free occurrence in this formula. A *proposition* or a *sentence* is a formula without free variables. If φ is a formula, then we denote by $\mathit{var}(\varphi)$ the set of the free variables of φ.

The syntax of the formulas being constraining, we allowed ourselves to use infix notations for the binary symbols and to add and remove brackets when there are no ambiguities. Moreover, we do not distinguish two formulas which can be made equal using the following transformations of sub-formulas:

$$\begin{gathered} \varphi \wedge \varphi \Longrightarrow \varphi,\ \ \varphi \wedge \psi \Longrightarrow \psi \wedge \varphi,\ \ (\varphi \wedge \psi) \wedge \phi \Longrightarrow \varphi \wedge (\psi \wedge \phi), \\ \varphi \wedge \mathit{true} \Longrightarrow \varphi,\ \ \varphi \vee \mathit{false} \Longrightarrow \varphi. \end{gathered}$$

If I is the set $\{i_1, ..., i_n\}$, we call *conjunction* of formulas and write $\bigwedge_{i \in I} \varphi_i$, each formula of the form $\varphi_{i_1} \wedge \varphi_{i_2} \wedge ... \wedge \varphi_{i_n} \wedge true$. In particular, for $I = \emptyset$, the conjunction $\bigwedge_{i \in I} \varphi_i$ is reduced to *true*.

3.1.2 Model

A *model* is a tuple $\mathcal{M} = < M, (f^M)_{f \in F}, (R^M)_{r \in R} >$, where:

- M, the *universe* or *domain* of $\mathcal{M}$, is a nonempty set disjoint from S, its elements are called *individuals* of $\mathcal{M}$;
- F and R are sets of n-ary functions and relations in the set M, subscripted by the elements of S and such that:
  - — for every n-ary function symbol f taken from S, f^M is an n-ary operation in M, i.e. an application from M^n in M. In particular, when f is a constant, f^M belongs to M;
  - — for every n-ary relation symbol r taken from S, r^M is an n-ary relation in M, i.e. a subset of M^n.

Let $\mathcal{M} = < M, F, R >$ be a model. An $\mathcal{M}$-*expression* φ is an expression built on the signature $S \cup M$ instead of S, by considering the elements of M as 0-ary function symbols. If for each free variable x of φ we replace each free occurrence of x by a same element m in M, we get an $\mathcal{M}$-expression φ' called *instantiation*[8] or *valuation* of φ by individuals of $\mathcal{M}$.

If φ is an $\mathcal{M}$-formula, we say that φ *is true in* $\mathcal{M}$ and we write

$$\mathcal{M} \models \varphi, \tag{3}$$

if for any instantiation φ' of φ by individuals of $\mathcal{M}$ the set M has the property expressed by $\varphi\prime$, when we interpret the function and relation symbols of $\varphi\prime$ by the corresponding functions and relations of $\mathcal{M}$ and when we give to the logical symbols their usual meaning.

Remark 3.1.3
For every $\mathcal{M}$-formula φ without free variables, one and only one of the following properties holds: $\mathcal{M} \models \varphi$, $\mathcal{M} \models \neg\varphi$.

Let us finish this sub-section by a convenient notation. Let $\bar{x} = x_1...x_n$ be a word on V and let $\bar{i} = i_1...i_n$ be a word on M or V of the same length as $\bar{x}$. If $\varphi(\bar{x})$ and ϕ are two $\mathcal{M}$-formulas, then we denote by $\varphi(\bar{i})$, respectively $\phi_{\bar{x} \leftarrow \bar{i}}$, the $\mathcal{M}$-formula obtained by replacing in $\varphi(\bar{x})$, respectively in ϕ, each free occurrence of x_j by i_j.

3.1.4 Theory

A *theory* is a (possibly infinite) set of propositions called *axioms*. We say that the model $\mathcal{M}$ is a *model of* T, if for each element φ of T, $\mathcal{M} \models \varphi$. If φ is a formula, we write

$$T \models \varphi,$$

[8] We also say that the variable x is instantiated by m in φ'.

if for each model $\mathcal{M}$ of T, $\mathcal{M} \models \varphi$. We say that the formulas φ and ψ are *equivalent in T* if $T \models \varphi \leftrightarrow \psi$.

Definition 3.1.5
A theory T is *complete* if for every proposition φ, one and only one of the following properties holds: $T \models \varphi$, $T \models \neg\varphi$.

Let ϕ be a formula and $\bar{x} = x_1 ... x_n$ be a word on V such that $var(\phi) = \bar{x}$. From the preceding definition we deduce that a decision procedure is sufficient in the case where we want just to show the completeness of a theory T, as it was done in (Djelloul 2006a) for decomposable theories. In fact, the completeness of T depends only on the truth values of the propositions in T. On the other hand, finding for each model $\mathcal{M}$ of T the instantiations $\bar{i}$ of $\bar{x}$ such that $\mathcal{M} \models \phi_{\bar{x} \leftarrow \bar{i}}$ can be obtained only using a first-order constraint solver in T. This kind of problem is generally known as *first-order constraint satisfaction problem.*

3.1.6 Vectorial quantifiers

Let $\mathcal{M}$ be a model and T a theory. Let $\bar{x} = x_1 \ldots x_n$ and $\bar{y} = y_1 \ldots y_n$ be two words on V of the same length. Let ϕ, φ and $\varphi(\bar{x})$ be $\mathcal{M}$-formulas. We write

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

The word $\bar{x}$, which can be the empty word ε, is called *vector of variables.* Note that the formulas $\exists ? \varepsilon \varphi$ and $\exists ! \varepsilon \varphi$ are respectively equivalent to *true* and to φ in any model $\mathcal{M}$.

Notation 3.1.7
Let Q be a quantifier taken from $\{\forall, \exists, \exists !, \exists ?\}$. Let $\bar{x}$ be vector of variables taken from V. We write:

$$Q\bar{x}\, \varphi \wedge \phi \;\; for \;\; Q\bar{x}\, (\varphi \wedge \phi).$$

Example 3.1.8
Let $I = \{1, ..., n\}$ be a finite set. Let φ and ϕ_i with $i \in I$ be formulas. Let $\bar{x}$ and $\bar{y}_i$ with $i \in I$ be vectors of variables. We write:

$$\begin{array}{lll} \exists \bar{x}\, \varphi \wedge \neg \phi_1 & \text{for} & \exists \bar{x}\, (\varphi \wedge \neg \phi_1), \\ \forall \bar{x}\, \varphi \wedge \phi_1 & \text{for} & \forall \bar{x}\, (\varphi \wedge \phi_1), \\ \exists ! \bar{x}\, \varphi \wedge \bigwedge_{i \in I} (\exists \bar{y}_i \phi_i) & \text{for} & \exists ! \bar{x}\, (\varphi \wedge (\exists \bar{y}_1 \phi_1) \wedge ... \wedge (\exists \bar{y}_n \phi_n) \wedge true), \\ \exists ? \bar{x}\, \varphi \wedge \bigwedge_{i \in I} \neg (\exists \bar{y}_i \phi_i) & \text{for} & \exists ? \bar{x}\, (\varphi \wedge (\neg (\exists \bar{y}_1 \phi_1)) \wedge ... \wedge (\neg (\exists \bar{y}_n \phi_n)) \wedge true). \end{array}$$

Notation 3.1.9
If $\bar{x}$ is a vector of variables then we denote by X the set of the variables of $\bar{x}$.

Let I be a (possible empty) finite set. The two following properties hold for any theory T:

Property 3.1.10
If $T \models \exists ? \bar{x}\, \varphi$ then

$$T \models (\exists \bar{x}\, \varphi \wedge \bigwedge_{i \in I} \neg \phi_i) \leftrightarrow ((\exists \bar{x} \varphi) \wedge \bigwedge_{i \in I} \neg (\exists \bar{x}\, \varphi \wedge \phi_i)).$$

Property 3.1.11
If $T \models \exists ! \bar{x}\, \varphi$ then

$$T \models (\exists \bar{x}\, \varphi \wedge \bigwedge_{i \in I} \neg \phi_i) \leftrightarrow \bigwedge_{i \in I} \neg (\exists \bar{x}\, \varphi \wedge \phi_i).$$

Full proofs of these two properties can be found in detail in (Djelloul 2006a).

3.2 The axioms of T

Let F be a set of function symbols containing infinitely many non-constant function symbols and at least one constant. Let *finite* be an 1-ary relation symbol. The theory T of finite or infinite trees built on the signature $S = F \cup \{\mathit{finite}\}$ has as axioms the infinite set of propositions of one of the five following forms:

$\forall \bar{x} \forall \bar{y}$	$\neg (f(\bar{x}) = g(\bar{y}))$	[1]
$\forall \bar{x} \forall \bar{y}$	$f(\bar{x}) = f(\bar{y}) \rightarrow \bigwedge_i x_i = y_i$	[2]
$\forall \bar{x} \exists ! \bar{z}$	$\bigwedge_i z_i = t_i[\bar{x}\bar{z}]$	[3]
$\forall \bar{x} \forall u$	$\neg (u = t[u, \bar{x}] \wedge \mathit{finite}(u))$	[4]
$\forall \bar{x} \forall u$	$(u = f(\bar{x}) \wedge \mathit{finite}(u)) \leftrightarrow (u = f(\bar{x}) \wedge \bigwedge_i \mathit{finite}(x_i))$	[5]

where f and g are distinct function symbols taken from F, $\bar{x}$ is a vector of (possibly non-distinct) variables x_i, $\bar{y}$ is a vector of (possibly non-distinct) variables y_i, $\bar{z}$ is a vector of distinct variables z_i, $t_i[\bar{x}\bar{z}]$ is a term which begins with an element of F followed by variables taken from $\bar{x}$ or $\bar{z}$, and $t[u, \bar{x}]$ is a term containing at least one occurrence of an element of F and the variable u and possibly other variables taken from $\bar{x}$. For example, we have $T \models \forall x_1 x_2 \forall u\, \neg (u = f_1(x_1, f_2(u, x_2)) \wedge \mathit{finite}(u))$ and $T \models \forall u \neg (u = f_1(f_2(u, f_0), f_0) \wedge \mathit{finite}(u))$ where f_1 and f_2 are 2-ary function symbols and f_0 a constant of F.

The forms [1],..., [5] are also called *schemas of axioms* of the theory T. Proposition [1] called *conflict of symbols* shows that two distinct operations produce two distinct individuals. Proposition [2] called *explosion* shows that the same operation on two distinct individuals produces two distinct individuals. Proposition [3] called *unique solution* shows that a certain form a conjunction of equations has a unique set of solutions in T. In particular, the formula $\exists z\, z = f(z)$ has a unique solution which is the infinite tree $f(f(f(...)))$. Proposition [4] means that a finite tree cannot be a strict subtree of itself. We emphasize strongly that $t[u, \bar{x}]$ should contain at least one occurrence of an element of F and the variable u. In Axiom [5], if $\bar{x}$ is the empty vector and f is a constant then we get $\forall u\, u = f \wedge \mathit{finite}(u) \leftrightarrow u = f$, which means that the property $\mathit{finite}(f)$ is true for each constant f of F.

This theory is an extension of the basic theory of finite or infinite trees given by M. Maher in (Maher 1988) and built on a signature containing an infinite set of function symbols. Maher's theory is composed of the three first axioms of T and its completeness was shown using a decision procedure which transforms each proposition into a Boolean combination of existentially quantified conjunctions of atomic formulas. Note also that both Maher's theory and the theory T do not accept full elimination of quantifiers, i.e. there exist some quantified formulas whose quantifiers cannot be eliminated. For example, the formula $\exists x\, y = f(x)$ is neither true nor false in T. It accepts in each model of T a set of solutions and another set of non-solutions. As a consequence, we cannot simplify it any further. This non-full elimination of quantifiers makes the completeness of T not evident.

3.3 The models of T

Let us extend the algebras given at the end of section 2.2 by the relation *finite*. More precisely, if u_1, u_2, u_3 and u_4 are respectively elements of D, Tr, Fi and Ra then the operations $\mathit{finite}^D(u_1)$, $\mathit{finite}^{Tr}(u_2)$, $\mathit{finite}^{Fi}(u_3)$ and $\mathit{finite}^{Ra}(u_4)$ are true respectively in D, Tr, Fi and Ra, if and only if u_1, u_2, u_3 and u_4 have a finite set of nodes.

Let us now denote by:

- $\mathcal{D} =< D, (f^D)_{f\in F}, \mathit{finite}^D >$, the model of sets of nodes,
- $\mathcal{T}r =< Tr, (f^{Tr})_{f\in F}, \mathit{finite}^{Tr} >$, the model of finite or infinite trees,
- $\mathcal{R}a =< Ra, (f^{Ra})_{f\in F}, \mathit{finite}^{Ra} >$, the model of rational trees,
- $\mathcal{F}i =< Fi, (f^{Fi})_{f\in F}, \mathit{finite}^{Fi} >$, the model of finite trees.

We have:

Theorem 3.3.1
The models $\mathcal{D}$, $\mathcal{T}r$ and $\mathcal{R}a$ are models of the theory T.

This theorem is one of the essential contributions given in this paper and shows that our theory T is in fact an axiomatization of the structures D, Tr and Ra together with an infinite set of construction operations and the 1-ary relation *finite*. It also shows that T has at least one model and thus $T \models \neg(\mathit{true} \leftrightarrow \mathit{false})$.

Proof, first part: Let us show first that the model $\mathcal{D}$ of sets of nodes is a model of T. In other words, we must show that the following properties hold:

$[1^D]$ $(\forall a_1, ..., a_m \in D)(\forall b_1, ..., b_n \in D)\, \neg(f^D(a_1, ..., a_m) = g^D(b_1, ..., b_n))$
$[2^D]$ $(\forall a_1, ..., a_n \in D)(\forall b_1, ..., b_n \in D)\, (f^D(a_1, ..., a_n) = f^D(b_1, ..., b_n) \rightarrow \bigwedge_{i=1}^n a_i = b_i)$
$[3^D]$ $(\forall a_1, ..., a_m \in D)(\exists! b_1, ..., b_n \in D)\, (\bigwedge_{i=1}^n b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m])$
$[4^D]$ $(\forall a_1, ..., a_m \in D)(\forall u \in D)\neg(u = t^D[u, a_1, ..., a_n] \wedge \mathit{finite}^D(u))$
$[5^D]$ $(\forall a_1, ..., a_n \in D)(\forall u \in D)(u = f^D(a_1, ..., a_n) \wedge \mathit{finite}^D(u)) \leftrightarrow$
$(u = f^D(a_1, ..., a_n) \wedge \bigwedge_{i=1}^n \mathit{finite}^D(a_i))$

where f and g are distinct function symbols taken from F, $t_i^D[b_1, ..., b_n, a_1, ..., a_m]$ is a term which begins with an element of F followed by variables taken from $\{a_1, ..., a_m, b_1, ..., b_n\}$, and $t^D[u, a_1, ..., a_n]$ is a term containing at least one occurrence of an element of F and the variable u and possibly other variables taken from $\{a_1, ..., a_n\}$. According to Definition 2.2.1 and the definition of the relation finite^D, the properties $[1^D]$, $[2^D]$, $[4^D]$ and $[5^D]$ hold. On the other hand, property $[3^D]$ is much less obvious and deserves to be proved.

Let $a_1, ..., a_m$ and $b_1, ..., b_n$ be elements of D. According to the first point of Remark 2.2.2, the $\mathcal{D}$-formula

$$\bigwedge_{i=1}^{n} b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m], \tag{4}$$

is equivalent in $\mathcal{D}$ to

$$\bigwedge_{k=0}^{\infty} \bigwedge_{i=1}^{n} \nu_k(b_i) = \nu_k(t_i^D[b_1, ..., b_n, a_1, ..., a_m]). \tag{5}$$

Let $i \in \{1, ..., n\}$. Let us denote by f_i respectively $[b_1, ..., b_n, a_1, ..., a_m]_i$ the function symbol respectively the set of the variables which occur in the term $t_i^D[b_1, ..., b_n, a_1, ..., a_m]$. According to the second and third point of Remark 2.2.2 we have:

- For each $i \in \{1, ..., n\}$ there exists one node $\varphi_0^i = (\varepsilon, f_i)$, such that
$$\nu_0(t_i^D[b_1, ..., b_n, a_1, ..., a_m]) = \{\varphi_0^i\}.$$
- For each $i \in \{1, ..., n\}$ and each $k \geq 0$ there exists a function φ_{k+1}^i, which is independent from all the $\nu_{k+1}(x)$, with $x \in [b_1, ..., b_n, a_1, ..., a_m]_i$, such that
$$\nu_{k+1}(t_i^D[b_1, ..., b_n, a_1, ..., a_m]) = \varphi_{k+1}^i([\nu_k(b_1), ..., \nu_k(b_n), \nu_k(a_1), ..., \nu_k(a_m)]_i),$$
where $[\nu_k(b_1), ..., \nu_k(b_n), \nu_k(a_1), ..., \nu_k(a_m)]_i$ is a tuple of elements of the form $\nu_k(x)$ for all $x \in [b_1, ..., b_n, a_1, ..., a_m]_i$.

Thus, the $\mathcal{D}$-formula (5) is equivalent in $\mathcal{D}$ to

$$(\bigwedge_{i=1}^{n} \nu_0(b_i) = \{\varphi_0^i\}) \wedge (\bigwedge_{k=0}^{\infty} \bigwedge_{i=1}^{n} \nu_{k+1}(b_i) = \varphi_{k+1}^i([\nu_k(b_1), ..., \nu_k(b_n), \nu_k(a_1), ..., \nu_k(a_m)]_i)),$$

from which we deduce that:

- (i) For all $i \in \{1, ..., n\}$, $\nu_0(b_i)$ has a constant value, which is equal to (ε, f_i).
- (ii) Each $\nu_{k+1}(b_i)$ depends in the worst case on $\nu_k(b_1), ..., \nu_k(b_n), \nu_k(a_1), ..., \nu_k(a_m)$, i.e. on $\nu_k(b_1), ..., \nu_k(b_n)$ and $a_1, ..., a_m$.

Thus, by recurrence[9] on k, we deduce that (iii) each $\nu_{k+1}(b_i)$ with $k \geq 0$ and $i \in \{1, ..., n\}$, depends only on $a_1, ..., a_m$. From (i) and (iii) we deduce that all the b_i's depend only on $a_1, ..., a_m$ and thus property $[3^D]$ holds. In other words, for each instantiation of $a_1, ..., a_m$ by elements of D we can deduce the values of $\nu_k(b_i)$ for all $i \in \{1, ..., n\}$ and $k \geq 0$.

We have shown that the model $\mathcal{D}$ satisfies the five axioms of T and thus it is a model of T.

[9] If $k = 0$ then according to (ii) each $\nu_1(b_i)$ depends in the worst case on $\nu_0(b_1), ..., \nu_0(b_n)$ and a_1,...,a_m. According to (i) all the $\nu_0(b_1), ..., \nu_0(b_n)$ have constant values and thus each $\nu_1(b_i)$ depends only on $a_1, ..., a_m$. Let us now assume that each $\nu_k(b_i)$ depends only on $a_1, ..., a_m$ and let us show that this hypothesis is true for $\nu_{k+1}(b_i)$. According to (ii), each $\nu_{k+1}(b_i)$ depends in the worst case on $\nu_k(b_1), ..., \nu_k(b_n)$ and $a_1, ..., a_m$, which according to our hypothesis depend only on $a_1, ..., a_m$. Thus, the recurrence is true for all $k \geq 0$.

Proof, second part: Let us now show that the model $\mathcal{T}r$ of finite or infinite trees is a model of T. For that, it is enough to show the validity of the following properties

$[1^{Tr}]$ $(\forall a_1, ..., a_m \in Tr)(\forall b_1, ..., b_n \in Tr)\, \neg(f^{Tr}(a_1, ..., a_m) = g^{Tr}(b_1, ..., b_n))$
$[2^{Tr}]$ $(\forall a_1, ..., a_n \in Tr)(\forall b_1, ..., b_n \in Tr)\, (f^{Tr}(a_1, ..., a_n) = f^{Tr}(b_1, ..., b_n) \rightarrow \bigwedge_{i=1}^{n} a_i = b_i)$
$[3^{Tr}]$ $(\forall a_1, ..., a_m \in Tr)(\exists! b_1, ..., b_n \in Tr)\, (\bigwedge_{i=1}^{n} b_i = t_i^{Tr}[b_1, ..., b_n, a_1, ..., a_m])$
$[4^{Tr}]$ $(\forall a_1, ..., a_m \in Tr)(\forall u \in Tr)\neg(u = t^{Tr}[u, a_1, ..., a_n] \wedge \mathit{finite}^{Tr}(u))$
$[5^{Tr}]$ $(\forall a_1, ..., a_n \in Tr)(\forall u \in Tr)(u = f^{Tr}(a_1, ..., a_n) \wedge \mathit{finite}^{Tr}(u)) \leftrightarrow$
$(u = f^{Tr}(a_1, ..., a_n) \wedge \bigwedge_{i=1}^{n} \mathit{finite}^{Tr}(a_i))$

where f and g are distinct function symbols taken from F, $t_i^{Tr}[b_1, ..., b_n, a_1, ..., a_m]$ is a term which begins with an element of F followed by variables taken from $\{a_1, ..., a_m, b_1, ..., b_n\}$, and $t^{Tr}[u, a_1, ..., a_n]$ is a term containing at least one occurrence of an element of F and the variable u and possibly other variables taken from $\{a_1, ..., a_n\}$. Since Tr is a subset of D, then according to the definition of $f^{Tr}, f^D, \mathit{finite}^{Tr}$ and finite^D, the properties $[1^D]$, $[2^D]$, $[4^D]$ and $[5^D]$ imply $[1^{Tr}]$, $[2^{Tr}]$, $[4^{Tr}]$ and $[5^{Tr}]$. On the other hand, to show property $[3^{Tr}]$, it is enough to show the following implication:

$$(\forall a_1, ..., a_m, b_1, ...b_n \in D)(((\bigwedge_{i=1}^{n} b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m]) \wedge (\bigwedge_{i=1}^{m} a_i \in Tr)) \rightarrow (\bigwedge_{i=1}^{n} b_i \in Tr)) \tag{6}$$

Let a, b, a_1,...,a_m, b_1,...,b_n be elements of D. Let us consider the following notation:

$$Arb(a, b) \leftrightarrow \text{each element of } a \text{ is arborescent in } b.$$

According to Definition 2.1.2, the $\mathcal{T}r$-formula

$$(\bigwedge_{i=1}^{n} b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m]) \wedge (\bigwedge_{i=1}^{m} a_i \in Tr),$$

is equivalent in $\mathcal{T}r$ to

$$(\bigwedge_{i=1}^{n} b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m]) \wedge (\bigwedge_{i=1}^{m} Arb(N, a_i)),$$

which is equivalent to

$$(\bigwedge_{i=1}^{n} b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m]) \wedge (\bigwedge_{k=0}^{\infty} \bigwedge_{i=1}^{m} Arb(\nu_k(N), a_i)), \tag{7}$$

which for each $j \geq 0$ is equivalent in $\mathcal{T}r$ to

$$(\bigwedge_{i=1}^{n} b_i = t_i^D[b_1, ..., b_n, a_1, ..., a_m]) \wedge (\bigwedge_{k=0}^{\infty} \bigwedge_{i=1}^{m} Arb(\nu_k(N), a_i)) \wedge (\bigwedge_{i=1}^{n} Arb(\nu_j(b_i), b_i)). \tag{8}$$

The equivalence $(7 \leftrightarrow 8)$ holds for $j = 0$ according to the fourth point of Remark 2.2.2, and if we assume that this equivalence holds for an integer j with $j \geq 0$ then according to the fifth point of Remark 2.2.2, we deduce that it holds also for $j + 1$. Thus, since the equivalence $(7 \leftrightarrow 8)$ holds for any $j \geq 0$ then according to the sixth point of Remark 2.2.2 and Definition 2.1.2 we deduce that (8) implies

$$\bigwedge_{i=1}^{n} Arb(N, b_i),$$

which, according to Definition 2.1.2, implies

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

Thus, the implication (6) holds and Tr is a model of T.

Proof, third part: Finally, let us show that the model $\mathcal{R}a$ is a model of T. For that, it is enough to show the validity of the following properties:

$[1^{Ra}]$ $(\forall a_1, ..., a_m \in Ra)(\forall b_1, ..., b_n \in Ra)\, \neg(f^{Ra}(a_1, ..., a_m) = g^{Ra}(b_1, ..., b_n))$
$[2^{Ra}]$ $(\forall a_1, ..., a_n \in Ra)(\forall b_1, ..., b_n \in Ra)\, (f^{Ra}(a_1, ..., a_n) = f^{Ra}(b_1, ..., b_n) \rightarrow \bigwedge_{i=1}^{n} a_i = b_i)$
$[3^{Ra}]$ $(\forall a_1, ..., a_m \in Ra)(\exists! b_1, ..., b_n \in Ra)\, (\bigwedge_{i=1}^{n} b_i = t_i^{Ra}[b_1, ..., b_n, a_1, ..., a_m])$
$[4^{Ra}]$ $(\forall a_1, ..., a_m \in Ra)(\forall u \in Ra)\neg(u = t^{Ra}[u, a_1, ..., a_n] \wedge \mathit{finite}^{Ra}(u))$
$[5^{Ra}]$ $(\forall a_1, ..., a_n \in Ra)(\forall u \in Ra)(u = f^{Ra}(a_1, ..., a_n) \wedge \mathit{finite}^{Ra}(u)) \leftrightarrow$
$(u = f^{Ra}(a_1, ..., a_n) \wedge \bigwedge_{i=1}^{n} \mathit{finite}^{Ra}(a_i))$

where f and g are distinct function symbols taken from F, $t_i^{Ra}[b_1, ..., b_n, a_1, ..., a_m]$ is a term which begins with an element of F followed by variables taken from $\{a_1, ..., a_m, b_1, ..., b_n\}$, and $t^{Ra}[u, a_1, ..., a_n]$ is a term containing at least one occurrence of an element of F and the variable u and possibly other variables taken from $\{a_1, ..., a_n\}$. Since Ra is a subset of Tr and according to the definitions of f^{Tr}, f^{Ra}, finite^{Tr} and finite^{Ra} then the properties $[1^{Tr}]$, $[2^{Tr}]$, $[4^{Tr}]$ and $[5^{Tr}]$ imply $[1^{Ra}]$, $[2^{Ra}]$, $[4^{Ra}]$ and $[5^{Ra}]$. On the other hand, in property $[3^{Tr}]$, (in the preceding proof), a subtree of depth k of any b_i is either one of the trees b_1,...,b_n or a subtree of one of the a_j's with $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$. This is true for $k = 0$ and if we assume that it is true for k then we deduce that it is true for $k+1$. Thus, if the a_j's are rational then the b_i's in $[3^{Tr}]$ are also rational and thus we get $[3^{Ra}]$.

We have shown that the models $\mathcal{D}$, $\mathcal{T}r$ and $\mathcal{R}a$ are models of T. What about the model $\mathcal{F}i$ of finite trees? Since F contains at least one function symbol f which is not a constant then according to Axiom [3] of T we have

$$T \models \exists! x\, x = f(x, ..., x).$$

It is obvious that this property cannot be true in $\mathcal{F}i$, i.e. there exists no $x \in Fi$ such that $x = f^{Fi}(x, ..., x)$. Thus, the model $\mathcal{F}i$ of finite trees is not a model of T.

Let us end this section by a property concerning the cardinality of any model of T:

Property 3.3.2
Let $\mathcal{M} =< M, (f^M)_{f\in F}, \mathit{finite}^M >$ be a model of T. The model $\mathcal{M}$ has an infinity of individuals i such that $\mathcal{M} \models \mathit{finite}^M(i)$.

Proof
Since the set F contains at least one function symbol f which is a constant then according to Axiom [5], with $\bar{x} = \varepsilon$, we have

$$\mathcal{M} \models \mathit{finite}^M(f^M). \tag{9}$$

On the other hand, according to the definition of the signature of T, the set F

contains an infinity of distinct function symbols which are not constants. Let f_1 one of these symbols. According to (9) and Axiom [5] we have

$$\mathcal{M} \models \mathit{finite}^M(f_1^M(f^M, ..., f^M)),$$

thus the individual $f_1^M(f^M, ..., f^M)$ is finite in $\mathcal{M}$. Since the set F contains an infinity of distinct function symbols $f_1, f_2, f_3, ...$ which are not constants then we can create by following the same preceding steps an infinity of finite individuals $f_1^M(f^M, ..., f^M), f_2^M(f^M, ..., f^M), f_3^M(f^M, ..., f^M), ...$ which start by distinct function symbols. According to Axiom [1], all these individuals are distinct. According to (9) and Axiom [5] all these individuals are finite in $\mathcal{M}$. $\square$

Corollary 3.3.3
Each model of T has an infinite domain, i.e. an infinite set of individuals.

4 Solving first-order constraints in T

4.1 Discipline of the formulas in T

Let us assume that the infinite set V is ordered by a strict linear dense order relation without endpoints denoted by $\succ$. Starting from this section, we impose the following discipline to every formula φ in T: the quantified variables of φ are renamed so that:

- (i) The quantified variables of φ have distinct names and different from those of the free variables.
- (ii) For all variables x, y and all sub-formulas[10] φ_i of φ, if y has a free occurrence in φ_i and x has a bound occurrence in φ_i then $x \succ y$.

Example 4.1.1
Let x, y, z, v be variables of V such that $x \succ y \succ z \succ v$. Let φ be the formula

$$\exists x\, x = fy \wedge \left[\begin{array}{l} \neg(\exists z\, z = x)\wedge \\ \neg(\exists z\, z = v) \end{array} \right]. \tag{10}$$

The quantified variables of φ have no distinct names. Since the order $\succ$ is dense and without endpoints, there exists a variable w in V such that $x \succ y \succ z \succ v \succ w$, and thus φ is equivalent in T to

$$\exists x\, x = fy \wedge \left[\begin{array}{l} \neg(\exists z\, z = x)\wedge \\ \neg(\exists w\, w = v) \end{array} \right].$$

In the preceding formula, the variables z and w have bound occurrences while the variables y and v have free occurrences. Since $x \succ y \succ z \succ v \succ w$ then z and w must be renamed. On the other hand, since the order $\succ$ is dense and without endpoints,

[10] By considering that each formula is also a sub-formula of itself.

there exist two variables u and d in V such that $x \succ u \succ d \succ y \succ z \succ v \succ w$. Thus, the preceding formula is equivalent in T to

$$\exists x\, x = fy \wedge \left[\begin{array}{l} \neg(\exists u\, u = x) \wedge \\ \neg(\exists d\, d = v) \end{array} \right].$$

In the sub-formula $(\exists u\, u = x)$ the variable x has a free occurrence while the variable u has a bound occurrence. Since $x \succ u$ then u must be renamed. On the other hand, since the order $\succ$ is dense and without endpoints, there exists a variable n in V such that $n \succ x \succ u \succ d \succ y \succ z \succ v \succ w$. Thus, the preceding formula is equivalent in T to

$$\exists x\, x = fy \wedge \left[\begin{array}{l} \neg(\exists n\, n = x) \wedge \\ \neg(\exists d\, d = v) \end{array} \right]. \tag{11}$$

This formula satisfies our conditions. Of course, the equivalence between (11) and (10) holds because in each step we renamed only the quantified variables. It is obvious that we can always transform any formula φ into an equivalent formula ϕ, which respects the discipline of the formulas in T, only by renaming the quantified variables of φ. It is enough for that to rename the quantified variables by distinct names and different from those of the free variables and then check each sub-formula and rename the quantified variables if the condition (ii) does not hold.

We emphasize strongly that all the formulas which will be used starting from now satisfy the discipline of the formulas in T.

4.2 Basic formula

In this sub-section we introduce particular conjunctions of atomic formulas that we call *basic formulas* and show some of their properties. All of them will be used to show the correctness of our rewriting rules given in section 4.6.

Definition 4.2.1
Let $v_1, ..., v_n, u_1, ..., u_m$ be variables. A *basic formula* is a formula of the form

$$(\bigwedge_{i=1}^{n} v_i = t_i) \wedge (\bigwedge_{i=1}^{m} \mathit{finite}(u_i)) \tag{12}$$

in which all the equations $v_i = t_i$ are flat. Note that if $n = m = 0$ then (12) is reduced to *true*. The basic formula (12) is called *solved* if all the variables $v_1, ..., v_n, u_1, ..., u_m$ are distinct and for each equation of the form $x = y$ we have $x \succ y$. If α is a basic formula then we denote by

- $Lhs(\alpha)$ the set of the variables which occur in the left hand sides of the equations of α.
- $FINI(\alpha)$ the set of the variables which occur in a sub-formula of α of the form *finite*(x).

Note that if α is a solved basic formula then for all variables x of α we have $x \in Lhs(\alpha) \rightarrow x \notin FINI(\alpha)$.

Example 4.2.2

The basic formula $x = x \wedge \mathit{finite}(y)$ is not solved because $x \not\succ x$. The basic formula $x = f(y) \wedge z = f(y) \wedge \mathit{finite}(x)$ is also not solved because x is a left hand side of an equation and occurs also in $\mathit{finite}(x)$. The basic formulas *true* (empty conjunction) and $x = f(y) \wedge z = f(y) \wedge \mathit{finite}(y)$ are solved.

According to the axiom [3] of T we deduce the following property:

Property 4.2.3

Let α be a solved basic formula containing only equations. Let $\bar{x}$ be the vector of the variables of $Lhs(\alpha)$. We have: $T \models \exists! \bar{x}\, \alpha$.

Property 4.2.4

Let α and β be two solved basic formulas containing only equations. If $Lhs(\alpha) = Lhs(\beta)$ and $T \models \alpha \rightarrow \beta$ then $T \models \alpha \leftrightarrow \beta$.

Proof

Let α and β be two solved basic formulas containing only equations such that $Lhs(\alpha) = Lhs(\beta)$ and $T \models \alpha \rightarrow \beta$. Let us show that we have also $T \models \beta \rightarrow \alpha$. Let $\bar{x}$ be the vector of the variables of $Lhs(\alpha)$ and let $\bar{y}$ be the vector of the variables which occur in $\alpha \rightarrow \beta$ and do not occur in $\bar{x}$. Since α and β are two solved basic formulas such that $Lhs(\alpha) = Lhs(\beta)$ then (i) $\bar{x}$ is also the vector of the left hand sides of the equations of β. Moreover, the following equivalences are true in T:

$$\begin{array}{lll}
 & \alpha \rightarrow \beta & \\
\leftrightarrow & \forall \bar{x} \forall \bar{y}\, \alpha \rightarrow \beta & \\
\leftrightarrow & \forall \bar{y} \forall \bar{x}\, \neg\alpha \vee \beta & \\
\leftrightarrow & \forall \bar{y} (\neg(\exists \bar{x}\, \alpha \wedge \neg\beta)) & \\
\leftrightarrow & \forall \bar{y} (\neg(\neg(\exists \bar{x}\, \alpha \wedge \beta))) & \text{according to the properties 4.2.3 and 3.1.11} \\
\leftrightarrow & \forall \bar{y} (\neg(\neg(\exists \bar{x}\, \beta \wedge \alpha))) & \\
\leftrightarrow & \forall \bar{y} (\neg(\exists \bar{x}\, \beta \wedge \neg\alpha)) & \text{according to: (i) and Property 4.2.3 and using the other} \\
 & & \text{sense (right to left) of the equivalence of Property 3.1.11} \\
\leftrightarrow & \forall \bar{y} \forall \bar{x}\, \neg\beta \vee \alpha & \\
\leftrightarrow & \forall \bar{y} \forall \bar{x}\, \beta \rightarrow \alpha & \\
\leftrightarrow & \beta \rightarrow \alpha &
\end{array}$$

□

Property 4.2.5

Let α be a basic formula containing only equations and β and δ two conjunctions of constraints of the form $\mathit{finite}(x)$ such that $\alpha \wedge \beta$ and $\alpha \wedge \delta$ are solved basic formulas. We have $T \models (\alpha \wedge \beta) \leftrightarrow (\alpha \wedge \delta)$ if and only if β and δ have exactly the same contraints.

Proof
If β and δ have the same constraints then it is evident that we have $T \models (\alpha \wedge \beta) \leftrightarrow (\alpha \wedge \delta)$. Let us now show that if we have $T \models (\alpha \wedge \beta) \leftrightarrow (\alpha \wedge \delta)$ then β and δ have the same constraints. Suppose that we have (*) $T \models (\alpha \wedge \beta) \leftrightarrow (\alpha \wedge \delta)$ and let us show that if $\mathit{finite}(u)$ occurs in β then it occurs also in δ and vice versa. If $\mathit{finite}(u)$ occurs in β then $T \models (\alpha \wedge \beta) \rightarrow \mathit{finite}(u)$, thus from (*) we have (i) $T \models (\alpha \wedge \delta) \rightarrow \mathit{finite}(u)$. Since $\alpha \wedge \beta$ is solved then u is not the left hand side of an equation of α. Thus, (ii) the conjunction $\alpha \wedge \delta$ does not contain sub-formulas of the form $u = t[\bar{x}] \wedge \bigwedge_i \mathit{finite}(x_i)$. Since $\alpha \wedge \delta$ is solved then δ does not contain formulas of the form $\mathit{finite}(v)$ where v is the left hand side of an equation of α. Thus, (iii) the conjunction $\alpha \wedge \delta$ does not contain also sub-formulas of the form $v = t[\bar{x}, u] \wedge \mathit{finite}(v)$. From (i), (ii) and (iii), $\mathit{finite}(u)$ should occur in δ. By the same reasoning (we replace β by δ and vice versa), we show that if $\mathit{finite}(u)$ occurs in δ then it occurs in β. $\square$

Let us now introduce the notion of *reachable variable*:

Definition 4.2.6
Let α be a basic formula and $\bar{x}$ a vector of variables. The reachable variables and equations of α from the variable x_0 are those which occur in a sub-formula of α of the form:

$$x_0 = t_0(x_1) \wedge x_1 = t_1(x_2) \wedge ... \wedge x_{n-1} = t_{n-1}(x_n),$$

where x_{i+1} occurs in the term $t_i(x_{i+1})$. The reachable variables and equations of $\exists \bar{x}\, \alpha$ are those which are reachable in α from the free variables of $\exists \bar{x}\, \alpha$. A sub-formula of α of the form $\mathit{finite}(u)$ is called reachable in $\exists \bar{x}\, \alpha$ if $u \notin \bar{x}$ or u is a reachable variable of $\exists \bar{x}\, \alpha$.

Example 4.2.7
In the formula: $\exists uvw\, z = f(u,v) \wedge v = g(v,u) \wedge w = f(u,v) \wedge \mathit{finite}(u) \wedge \mathit{finite}(x)$, the equations $z = f(u,v)$ and $v = g(v,u)$, the variables z, u and v and the formulas $\mathit{finite}(u)$ and $\mathit{finite}(x)$ are reachable. On the other hand the equation $w = f(u,v)$ and the variable w are not reachable.

Remark 4.2.8
Let α be a solved basic formula. Let $\bar{x}$ be a vector of variables. We have:

- If all the variables of $\bar{x}$ are reachable in $\exists \bar{x}\, \alpha$ then all the equations and relations of α are reachable in $\exists \bar{x}\, \alpha$.
- If $v = t[y]$ is a reachable equation in $\exists \bar{x}\, \alpha$, then α contains a sub-formula of the form
$$\bigwedge_{j=1}^{k} v_j = t_j[v_{j+1}] \tag{13}$$
with $k \geq 1$ and (i) $v_1 \notin X$, (ii) for all $j \in \{1, ..., k\}$ the variable v_{j+1} occurs in the term $t_j[v_{j+1}]$, (iii) v_k is the variable v, (iv) v_{k+1} is the variable y and $t_k[v_{k+1}]$ is the term $t[y]$.

According to the first point of Remark 4.2.8 and Definition 4.2.6 we have the following property:

Property 4.2.9
Let α be a solved basic formula. If the formula $\exists \bar{x}\, \alpha$ has no free variables and if all the variables of $\bar{x}$ are reachable in $\exists \bar{x}\, \alpha$ then $\bar{x}$ is the empty vector ε and α is the formula *true*.

According to the axioms [1] and [2] of T we have the following property:

Property 4.2.10
Let α be a basic formula. If all the variables of $\bar{x}$ are reachable in $\exists \bar{x}\, \alpha$ then

$$T \models \exists ? \bar{x}\, \alpha.$$

Property 4.2.11
Let $\bar{x}$ be a vector of variables and α a solved basic formula. We have:

$$T \models (\exists \bar{x}\, \alpha) \leftrightarrow (\exists \bar{x}'\, \alpha'),$$

where:

- $\bar{x}'$ is the vector of the variable of $\bar{x}$ which are reachable in $\exists \bar{x}\, \alpha$,
- α' is the conjunction of the equations and the formulas of the form *finite*(x) which are reachable in $\exists \bar{x}\, \alpha$.

Proof
Let us decompose $\bar{x}$ into three vectors $\bar{x}', \bar{x}''$ and $\bar{x}'''$ such that:

- $\bar{x}'$ is the vector of the variables of $\bar{x}$ which are reachable in $\exists \bar{x}\, \alpha$.
- $\bar{x}''$ is the vector of the variables of $\bar{x}$ which are non-reachable in $\exists \bar{x}\, \alpha$ and do not occur in the left hand sides of the equations of α.
- $\bar{x}'''$ is the vector of the variables of $\bar{x}$ which are non-reachable in $\exists \bar{x}\, \alpha$ and occur in a left hand side of an equation of α.

Let us now decompose α into three formulas α', α'' and α''' such that:

- α' is the conjunction of the equations and the formulas of the form *finite*(x) which are reachable in $\exists \bar{x}\, \alpha$.
- α'' is the conjunction of the formulas of the form *finite*(x) which are non-reachable in $\exists \bar{x}\, \alpha$.
- α''' is the conjunction of the equations which are non-reachable in $\exists \bar{x}\, \alpha$.

According to Definition 4.2.6, all the variables of $\bar{x}''$ and $\bar{x}'''$ do not occur in α' (otherwise they will be reachable) and since α is solved then $\bar{x}'''$ is the vector of the left hand sides of the equations of α''' and its variables do not occur in α''. Thus the formula $\exists \bar{x}\, \alpha$ is equivalent in T to

$$(\exists \bar{x}'\, \alpha' \wedge (\exists \bar{x}''\, \alpha'' \wedge (\exists \bar{x}'''\, \alpha'''))).$$

According to Property 4.2.3 we have $T \models \exists ! \bar{x}'''\, \alpha'''$. According to Corollary 3.3.3 we have $T \models \exists \bar{x}''\, \alpha''$. Thus, the preceding formula is equivalent in T to $(\exists \bar{x}'\, \alpha')$. $\square$

Example 4.2.12
The formula $\exists xyzw\, v = f(x,x) \wedge w = g(y,z,x) \wedge \mathit{finite}(x) \wedge \mathit{finite}(y)$ is equivalent in T to

$$\exists x\, v = f(x,x) \wedge \mathit{finite}(x) \wedge (\exists yz\, \mathit{finite}(y) \wedge (\exists w\, w = g(y,z,x))),$$

which, since $T \models \exists! w\, w = g(y,z,x)$ and $T \models \exists yz\, \mathit{finite}(y)$, is equivalent in T to

$$\exists x\, v = f(x,x) \wedge \mathit{finite}(x).$$

Property 4.2.11 confirms the fact that the theory T does not accept full elimination of quantifiers and shows that we can eliminate only non-reachable quantified variables. On the other hand, reachable variables cannot be removed since their values depend on the instantiations of the free variables. In fact, the formula $\exists x\, v = f(x,x) \wedge \mathit{finite}(x)$ is neither true nor false in T since for each model $\mathcal{M}$ of T there exist instantiations of the free variable v which make it false in $\mathcal{M}$ and others which make it true in $\mathcal{M}$, and thus the reachable quantified variable x cannot be eliminated and the formula $\exists x\, v = f(x,x) \wedge \mathit{finite}(x)$ cannot be simplified anymore. On the other hand, the formula $\exists w\, w = g(y,z,x)$ is true in any model of T and for any instantiation of z. The quantified non-reachable variable w can then be eliminated and the formula is replaced by *true*. As we will see in section 4.6, reachability, has a crucial role while solving first-order constraints in T. It shows which quantifications can be eliminated and enables to simplify complex quantified basic formulas.

According to the axioms [1] and [2] and since the set F is infinite we have the following property:

Property 4.2.13
Let $I = \{1, ..., n\}$ be a finite (possibly empty) set and $\bar{x}$ and $\bar{x}'$ two disjoint vectors of variables. Let $\bar{y}_1$,...,$\bar{y}_n$ be vectors of variables and α_1,...,α_n solved basic formulas such that for all $i \in I$ all the variables of $\bar{y}_i$ are reachable in $\exists \bar{y}_i\, \alpha_i$. If each conjunction α_i contains at least (1) one sub-formula of the form $\mathit{finite}(x)$ with $x \in X$, **or** (2) one equation which contains at least one occurrence of a variable $x \in X \cup X'$, then:

$$T \models \exists \bar{x}\bar{x}' (\bigwedge_{x\in X'} \mathit{finite}(x)) \wedge (\bigwedge_{i\in I} \neg(\exists \bar{y}_i\, \alpha_i)). \tag{14}$$

Proof
Let $\mathcal{M} =< M, (f^M)_{f\in F}, \mathit{finite}^M >$ be a model of T. To show the validity of (14) it is enough to show that:

$$\mathcal{M} \models \exists \bar{x}\bar{x}' (\bigwedge_{x\in X'} \mathit{finite}^M(x)) \wedge (\bigwedge_{i\in I} \neg(\exists \bar{y}_i\, \alpha_i)). \tag{15}$$

Since the basic formulas α_i are solved, they do not contain equations of the form $x = x$. Suppose now that one of the α_i contains one equation of the form $x = v$ with $x \in X \cup X'$ and $v \in Y_i$. Since α_i is solved then $x \succ v$ but according to the discipline of the formulas in T we have $v \succ x$[11]. Since the order $\succ$ is strict then $x = v$ cannot be a sub-formula of α_i. Thus, according to the conditions of Property 4.2.13, each conjunction α_i contains at least (1) one sub-formula of the form $\mathit{finite}(x)$ with $x \in X$, **or** (2) one equation of the one of the following forms:

- (*) $x = f(v_1, ...v_n)$ with $x \in X \cup X'$,
- (**) $x = v$ with x and v two distinct variables such that $x \in X \cup X'$ and $v \notin Y$,
- (***) $v = t[x]$ where x is a variable of $X \cup X'$ which occurs in the term $t[x]$. According to the first point of Remark 4.2.8 and since for all $i \in \{1, ..., n\}$ the variables of $\bar{y}_i$ are reachable in $\exists \bar{y}_i\, \alpha_i$, then the equation $v = t[x]$ is reachable in $\exists \bar{y}_i\, \alpha_i$ and thus according to the second point of Remark 4.2.8 the conjunction α_i contains a sub-formula of the form $(\bigwedge_{j=1}^{k} v_j = t_j[v_{j+1}])$ with $v_1 \notin Y_i$, for all $j \in \{1, ..., k\}$ the variable v_{j+1} occurs in the term $t_j[v_{j+1}]$ and v_{k+1} is the variable x. But, since the case $v_1 \in X \cup X'$ is already treated in (*) and (**), then we can restrict ourself without loosing generality to the case where $v_1 \notin Y_i \cup X \cup X'$, i.e. v_1 is free in (15).

Let

$$\exists \bar{x}\bar{x}' (\bigwedge_{x \in X'} \mathit{finite}^M(x)) \wedge (\bigwedge_{i \in I} \neg(\exists \bar{y}_i\, \alpha_i^*)) \tag{16}$$

be an any instantiation of $\exists \bar{x}\bar{x}' (\bigwedge_{x \in X'} \mathit{finite}^M(x)) \wedge (\bigwedge_{i \in I} \neg(\exists \bar{y}_i\, \alpha_i))$ by individuals of $\mathcal{M}$. Let us show that there exists an instantiation for the variables of X and X' which satisfies the preceding formula. For that, let us chose an instantiation which respects the following conditions:

- (i) For each $x \in X'$, the instantiation x^* of x satisfies $\mathcal{M} \models \mathit{finite}^M(x^*)$.
- (ii) If a conjunction α_i^* contains a sub-formula of the form $\mathit{finite}^M(x)$ with $x \in X$ then the instantiation x^* of x satisfies $\mathcal{M} \models x^* = f^M(x^*, ..., x^*)$ with f an n-ary function symbol of strictly positive arity which does not occur in any α_i with $i \in I$.
- (iii) If a conjunction α_i^* contains a sub-formula of the form $x = f^M(v_1, ...v_n)$ with $x \in X \cup X'$, then the instantiation of x starts with a different function symbol than f.
- (iv) If a conjunction α_i^* contains a sub-formula of the form $x = v$ with x and v two distinct variables such that $x \in X \cup X'$ and $v \notin Y$, then the instantiation of x is different from those of v.
- (v) If a conjunction α_i^* contains a sub-formula of the form $(\bigwedge_{j=1}^{k} v_j = t_j[v_{j+1}])$ with $v_1 \notin (X \cup X' \cup Y)$, for all $j \in \{1, ..., k\}$ the variable v_{j+1} occurs in the term $t_j[v_{j+1}]$, and $v_{k+1} \in X \cup X'$, then the instantiation of v_{k+1} is different from v^*, where v^* is the instantiation of v_{k+1} obtained from those of v_1 in[12] (16) so that $\mathcal{M} \models \bigwedge_{j=1}^{k} v_j = t_j[v_{j+1}]$.

[11] In fact, the variable x has a free occurrence in $\exists \bar{y}_i\, \alpha_i$ and the variable v has a bound occurrence in $\exists \bar{y}_i\, \alpha_i$ (because v is a quantified reachable variable in $\exists \bar{y}_i\, \alpha_i$) and thus according to the discipline of our formulas we have $v \succ x$.

[12] Recall that $v_1 \notin (X \cup X' \cup Y)$ and thus v_1 is a free variable in (15). As a consequence, it is already instantiated in (16).

A such instantiation of the variables of X and X' is always possible since : (1) there exists an infinity of function symbols in F which are not constants (2) the set of the individuals i of $\mathcal{M}$ such that $\mathcal{M} \models \mathit{finite}^M(i)$ is infinite (see Property 3.3.2). As a consequence, according to axioms [1] and [4], this instantiation implies a conflict inside each sub-instantiated-formula $\exists \bar{y}_i\, \alpha_i^*$, with $i \in \{1, ..., n\}$ and thus

$$\mathcal{M} \models \exists \bar{x}\bar{x}' (\bigwedge_{i \in I} \neg(\exists \bar{y}_i\, \alpha_i^*)).$$

Since this instantiation satisfies the first condition (i) of the preceding list of conditions then (16) holds and thus (15) holds. $\square$

We emphasize strongly that this property holds only if the formula (14) satisfies the discipline of the formulas in T. This property is vital for solving first-order constraint over finite or infinite trees. In fact, since the variables of each $\bar{y}_i$ with $i \in \{1, ..., n\}$ are reachable in $\exists \bar{y}_i\, \alpha_i$ then we cannot eliminate or remove the quantification $\exists \bar{y}_i$ form $\exists \bar{y}_i\, \alpha_i$, and thus solving a constraint containing such formulas is not evident. Property 4.2.13 enables us to surmount this problem by reducing to *true* particular formulas containing sub-formulas which does not accept full elimination of quantifiers.

Example 4.2.14
Let x, y, z and v be variables such that $y \succ x \succ z \succ w$. Let us consider the following formula φ:

$$\exists x \begin{bmatrix} \neg(\exists y\, z = f(y) \wedge y = g(x)) \wedge \\ \neg(\exists \varepsilon\, x = w) \wedge \\ \neg(\exists \varepsilon\, x = g(x)) \end{bmatrix}. \tag{17}$$

This formula satisfies the discipline of the formulas in T. Let $\mathcal{M} =< M, (f^M)_{f \in F}, \mathit{finite}^M >$ be a model of T. Note that we cannot eliminate the quantifier $\exists y$ in the sub-formula $\exists y\, z = f(y) \wedge y = g(x)$. In fact, this sub-formula is neither true nor false in T because there exist instantiations of the free variable z in $\mathcal{M}$ which satisfy this sub-formula in $\mathcal{M}$ and others which do not satisfy it. On the other hand, Property 4.2.13, states that formula (17) is true in T for all instantiations of z even if the sub-formula $\exists y\, z = f(y) \wedge y = g(x)$ is neither true nor false in T. Let us check this strange result. For that, let us show that for each instantiation of the free variables z and w by two individuals z^* and w^* of $\mathcal{M}$, there exists an instantiation x^* of x which makes false the three $\mathcal{M}$-formulas $(\exists y\, z^* = f^M(y) \wedge y = g(x^*))$, $(\exists \varepsilon\, x^* = w^*)$ and $(\exists \varepsilon\, x^* = g(x^*))$. We have:

- In the formula $(\exists y\, z = f(y) \wedge y = g(x))$, the variable x is reachable. Thus, its value is determined by the value of z (because $z = f(g(x))$). Two cases arise:
  - — If z^* is of the form $f(g(i))$ with $i \in M$ then it is enough to instantiate x by an individual $x^* \in M$ which is different from[13] i, in order to make false $(\exists y\, z^* = f^M(y) \wedge y = g^M(x^*))$ in $\mathcal{M}$.

[13] For example, we can take $x^* = f^M(i)$.

— if z^* is not of the form $f(g(i))$ with $i \in M$ then the $\mathcal{M}$-formula $(\exists y\, z^* = f^M(y) \wedge y = g^M(x))$ is false in $\mathcal{M}$ for all the instantiations of x.

- In the $\mathcal{M}$-formula $(\exists\varepsilon\, x = w^*)$, it is enough to instantiate x by an element x^* of $\mathcal{M}$ which is different from w^* in order to make false the $\mathcal{M}$-formula $(\exists\varepsilon\, x^* = w^*)$.
- In the $\mathcal{M}$-formula $(\exists\varepsilon\, x = g^M(x))$, it is enough to instantiate x by an individual which starts by a distinct function symbol than g in order to make false $(\exists\varepsilon\, x = g^M(x))$ in $\mathcal{M}$.

Since the set of the functions symbols which are not constants is infinite then there exists an infinity of instantiations of x which satisfy the three preceding conditions. Each of these instantiations x^* makes false the three $\mathcal{M}$-formulas $(\exists y\, z^* = f^M(y) \wedge y = g^M(x^*))$, $(\exists\varepsilon\, x^* = w^*)$ and $(\exists\varepsilon\, x^* = g^M(x^*))$ and thus (17) holds.

4.3 Normalized formula

Definition 4.3.1
A normalized formula φ of depth $d \geq 1$ is a formula of the form

$$\neg(\exists\bar{x}\, \alpha \wedge \bigwedge_{i\in I} \varphi_i), \tag{18}$$

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

Example 4.3.2
Let f and g be two 1-ary function symbols which belong to F. The formula

$$\neg\left[\exists\varepsilon\, \mathit{finite}(u) \wedge \left[\begin{array}{l} \neg(\exists x\, y = f(x) \wedge x = g(y) \wedge \neg(\exists\varepsilon\, y = g(x) \wedge \mathit{finite}(x)))\wedge \\ \neg(\exists\varepsilon\, x = f(z) \wedge \mathit{finite}(z)) \end{array}\right]\right]$$

is a normalized formula of depth equals to three. The formula $\neg(\exists\varepsilon\, \mathit{true})$ is a normalized formula of depth 1. The smallest value of a depth of a normalized formula is 1. Normalized formulas of depth 0 are not defined and do not exist.

We will use now the abbreviation wnfv for *"without new free variables"*. A formula φ is equivalent to a wnfv formula ψ in T means that $T \models \varphi \leftrightarrow \psi$ and ψ does not contain other free variables than those of φ.

Property 4.3.3
Every formula φ is equivalent in T to a wnfv normalized formula of depth $d \geq 1$.

Proof
It is easy to transform any formula into a normalized formula, it is enough for example to follow the followings steps:

1. Introduce a supplement of equations and existentially quantified variables to transform the conjunctions of atomic formulas into conjunctions of flat formulas.

2. Replace each sub-formula of the form *false* by $\neg true$ then express all the quantifiers and logical connectors using only the logical symbols $\neg$, $\wedge$ and $\exists$. This can be done using the following transformations[14] of sub-formulas:

$$\begin{array}{lll}
(\varphi \vee \phi) & \Longrightarrow & \neg(\neg\varphi \wedge \neg\phi), \\
(\varphi \rightarrow \phi) & \Longrightarrow & \neg(\varphi \wedge \neg\phi), \\
(\varphi \leftrightarrow \phi) & \Longrightarrow & (\neg(\varphi \wedge \neg\phi) \wedge \neg(\phi \wedge \neg\varphi)), \\
(\forall x\, \varphi) & \Longrightarrow & \neg(\exists x\, \neg\varphi).
\end{array}$$

3. If the formula φ obtained does not start with the logical symbol $\neg$, then replace it by $\neg(\exists\varepsilon\, true \wedge \neg\varphi)$.
4. Rename the quantified variables so that the obtained formula satisfies the imposed discipline in T (see Section 4.1).
5. Lift the quantifier before the conjunction, i.e. $\varphi \wedge (\exists\bar{x}\,\psi)$ or $(\exists\bar{x}\,\psi) \wedge \varphi$, becomes $\exists\bar{x}\,\varphi \wedge \psi$ because the free variables of φ are distinct from those of $\bar{x}$.
6. Group the quantified variables into a vectorial quantifier, i.e. $\exists\bar{x}(\exists\bar{y}\,\varphi)$ or $\exists\bar{x}\exists\bar{y}\,\varphi$ becomes $\exists\overline{xy}\,\varphi$.
7. Insert empty vectors and formulas of the form *true* to get the normalized form using the following transformations of sub-formulas:

$$\neg(\bigwedge_{i\in I} \neg\varphi_i) \Longrightarrow \neg(\exists\varepsilon\, true \wedge \bigwedge_{i\in I} \neg\varphi_i), \tag{19}$$

$$\neg(\alpha \wedge \bigwedge_{i\in I} \neg\varphi_i) \Longrightarrow \neg(\exists\varepsilon\, \alpha \wedge \bigwedge_{i\in I} \neg\varphi_i), \tag{20}$$

$$\neg(\exists\bar{x} \bigwedge_{j\in J} \neg\varphi_j) \Longrightarrow \neg(\exists\bar{x}\, true \wedge \bigwedge_{j\in J} \neg\varphi_j). \tag{21}$$

with α a conjunction of elementary equations, I a finite (possibly empty) set and J a finite non-empty set.
8. Rename the quantified variables so that the obtained normalized formula satisfies the discipline of the formulas in T.

If the starting formula does not contain the logical symbol $\leftrightarrow$ then this transformation will be linear, i.e. there exists a constant k such that $n_2 \leq kn_1$, where n_1 is the size of the starting formula and n_2 the size of the normalized formula. We show easily by contradiction that the final formula obtained after application of these steps is normalized. $\square$

Example 4.3.4
Let x, v, w, u be variables such that $x \succ v \succ w \succ u$. Let f be a 2-ary function symbol which belongs to F. Let us apply the preceding steps to transform the following formula into a normalized formula:

$$(f(u,v) = f(w,u) \wedge (\exists x\, u = x)) \vee (\exists u \forall w\, u = f(v,w)).$$

Note that the formula does not start with $\neg$ and the variables u and w are free in

[14] These equivalences are true in the empty theory and thus in any theory T.

$f(u,v) = f(w,u) \wedge (\exists x\, u = x)$ and bound in $\exists u \forall w\, u = f(v,w)$. Note also that this formula does not respect the discipline of the formulas in T.
Step 1: Let us first transform the equations into flat equations. The preceding formula is equivalent in T to

$$(\exists u_1\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge (\exists x\, u = x)) \vee (\exists u \forall w\, u = f(v,w)), \tag{22}$$

where u_1 is a variable of V such that $u_1 \succ x \succ v \succ w \succ u$.
Step 2: Let us now express the quantifier $\forall$ using $\neg$, $\wedge$ and $\exists$. Thus, the formula (22) is equivalent in T to

$$(\exists u_1\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge (\exists x\, u = x)) \vee (\exists u\, \neg(\exists w\, \neg(u = f(v,w)))).$$

Let us also express the logical symbol $\vee$ using $\neg$, $\wedge$ and $\exists$. Thus, the preceding formula is equivalent in T to

$$\neg(\neg(\exists u_1\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge (\exists x\, u = x)) \wedge \neg(\exists u\, \neg(\exists w\, \neg(u = f(v,w))))). \tag{23}$$

Step 3: As the formula starts with $\neg$, we move to Step 4.
Step 4: The occurrences of the quantified variables u and w in $(\exists u\, \neg(\exists w\, \neg(u = f(v,w))))$ must be renamed. Thus, the formula (23) is equivalent in T to

$$\neg(\neg(\exists u_1\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge (\exists x\, u = x)) \wedge \neg(\exists u_2\, \neg(\exists w_1\, \neg(u_2 = f(v,w_1))))),$$

where u_2 and w_1 are variables of V such that $w_1 \succ u_2 \succ u_1 \succ x \succ v \succ w \succ u$.
Step 5: By lifting the existential quantifier $\exists x$, the preceding formula is equivalent in T to

$$\neg(\neg(\exists u_1\, \exists x\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge u = x) \wedge \neg(\exists u_2\, \neg(\exists w_1\, \neg(u_2 = f(v,w_1))))).$$

Step 6: Let us group the two quantified variables x and u_1 into a vectorial quantifier. Thus, the preceding formula is equivalent in T to

$$\neg(\neg(\exists u_1 x\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge u = x) \wedge \neg(\exists u_2\, \neg(\exists w_1\, \neg(u_2 = f(v,w_1))))).$$

Step 7: Let us introduce empty vectors of variables and formulas of the form *true* to get the normalized formula. According to the rule (19), the preceding formula is equivalent in T to

$$\neg \left[\exists \varepsilon\, \mathit{true} \wedge \left[\begin{array}{l} \neg(\exists u_1 x\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge u = x) \wedge \\ \neg(\exists u_2\, \neg(\exists w_1\, \neg(u_2 = f(v,w_1)))) \end{array} \right] \right],$$

which using the rule (20) with $I = \emptyset$ is equivalent in T to

$$\neg \left[\exists \varepsilon\, \mathit{true} \wedge \left[\begin{array}{l} \neg(\exists u_1 x\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge u = x) \wedge \\ \neg(\exists u_2\, \neg(\exists w_1\, \neg(\exists \varepsilon\, u_2 = f(v,w_1)))) \end{array} \right] \right],$$

which using the rule (21) is equivalent in T to

$$\neg \left[\exists \varepsilon\, \mathit{true} \wedge \left[\begin{array}{l} \neg(\exists u_1 x\, u_1 = f(u,v) \wedge u_1 = f(w,u) \wedge u = x) \wedge \\ \neg(\exists u_2\, \mathit{true} \wedge \neg(\exists w_1\, \mathit{true} \wedge \neg(\exists \varepsilon\, u_2 = f(v,w_1)))) \end{array} \right] \right].$$

Step 8: This is a normalized formula of depth 4 which respects the discipline of the formulas in T since $w_1 \succ u_2 \succ u_1 \succ x \succ v \succ w \succ u$.

4.4 General solved formula

Definition 4.4.1

A *general solved* formula is a normalized formula of the form

$$\neg(\exists \bar{x}\, \alpha \wedge \bigwedge_{i=1}^{n} \neg(\exists \bar{y}_i\, \beta_i)),$$

with $n \geq 0$ and such that:

1. α and all the β_i, with $i \in \{1, ..., n\}$, are solved basic formulas.
2. If α' is the conjunction of the equations of α then all the conjunctions $\alpha' \wedge \beta_i$, with $i \in \{1, ..., n\}$, are solved basic formulas.
3. All the variables of $\bar{x}$ are reachable in $\exists \bar{x}\, \alpha$.
4. For all $i \in \{1, ..., n\}$, all the variables of $\bar{y}_i$ are reachable in $\exists \bar{y}_i\, \beta_i$.
5. If $\mathit{finite}(u)$ is a sub-formula of α then for all $i \in \{1, ..., n\}$, the formula β_i contains either $\mathit{finite}(u)$, or $\mathit{finite}(v)$ where v is a reachable variable from u in $\alpha \wedge \beta_i$ and does not occur in a left hand side of an equation of $\alpha \wedge \beta_i$.
6. For all $i \in \{1, ..., n\}$, the formula β_i contains at least one atomic formula which does not occur in α.

Example 4.4.2

Let w, v, u_1, u_2, u_3 be variables such that $w \succ v \succ u_1 \succ u_2 \succ u_3$. The following formula is not a general solved formula

$$\neg(\exists \varepsilon\, \mathit{finite}(w) \wedge \neg(\exists v\, w = v \wedge \mathit{finite}(v))). \tag{24}$$

This formula satisfies all the conditions of Definition 4.4.1 but it does not satisfy the discipline of the formulas in T. In fact, the variable v is bound in $(\exists v\, w = v \wedge \mathit{finite}(v))$ and the variable w is free in $(\exists v\, w = v \wedge \mathit{finite}(v))$ and thus we should have $v \succ w$ and not $w \succ v$. Let u_4 be a variable such that $u_4 \succ w \succ v \succ u_1 \succ u_2 \succ u_3$. The formula (24) is equivalent in T to

$$\neg(\exists \varepsilon\, \mathit{finite}(w) \wedge \neg(\exists u_4\, w = u_4 \wedge \mathit{finite}(v))).$$

This formula respects the discipline of the formulas of T but is not a general solved formula since it does not satisfy the first condition of Definition 4.4.1. In fact, $w = u_4 \wedge \mathit{finite}(v)$ is not a solved basic formula since we have $u_4 \succ w$.

The following formula is a general solved formula

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

Property 4.4.3

Let φ be a general solved formula. If φ has no free variables then φ is the formula $\neg(\exists \varepsilon\, \mathit{true})$ else neither $T \models \neg\varphi$ nor $T \models \varphi$.

Proof
Let φ be a general solved formula of the form

$$\neg(\exists \bar{x}\, \alpha \wedge \bigwedge_{i\in I} \neg(\exists \bar{y}_i\, \beta_i)), \tag{25}$$

two cases arise:

(1) If φ does not contain free variables, then according to the first and third condition of Definition 4.4.1 and using Property 4.2.9 we get $\bar{x} = \varepsilon$ and $\alpha = \mathit{true}$. As a consequence, the formula (25) is equivalent in T to

$$\neg(\exists \varepsilon\, \mathit{true} \wedge \bigwedge_{i\in I} \neg(\exists \bar{y}_i\, \beta_i)), \tag{26}$$

Since (26) has no free variables then each $\exists \bar{y}_i\, \beta_i$ has no free variables. According to the first and fourth condition of Definition 4.4.1, and using Property 4.2.9 we get: for all $i \in I$: $\bar{y}_i = \varepsilon$ and $\beta_i = \mathit{true}$. But according to the last condition of Definition 4.4.1 all the formulas β_i should be different from *true* (since we do not distinguish between α and $\alpha \wedge \mathit{true}$). Thus, the set I must be empty. As a consequence, φ is the formula $\neg(\exists \varepsilon\, \mathit{true})$.

(2) If φ contains free variables then it is enough to show that there exist two distinct instantiations φ' and φ'' of φ by individuals of $\mathcal{T}r$[15] such that

$$\mathcal{T}r \models \varphi' \textit{ and } \mathcal{T}r \models \neg\varphi''.$$

Note first that if $I \neq \emptyset$ then each $(\exists \bar{y}_i\, \beta_i)$, with $i \in I$, should contain at least one free variable. In fact, if $(\exists \bar{y}_i\, \beta_i)$, with $i \in I$, does not contain free variables then this formula is of the form $(\exists \varepsilon\, \mathit{true})$ according to the first and fourth point of Definition 4.4.1 and Property 4.2.9, which contradicts the last condition of Definition 4.4.1 (since we do not distinguish between α and $\alpha \wedge \mathit{true}$). Thus each $(\exists \bar{y}_i\, \beta_i)$, with $i \in I$, contains at least one free variable that can be instantiated. On the other hand:

Case 1: If $\exists \bar{x}\, \alpha$ contains free variables then we can easily find an instantiation of the free variables of $\exists \bar{x}\, \alpha$ which contradicts the constraints of α. In fact, let z be a free variable. Four cases arise:

- If $z = w$ is a sub-formula of α then according to Definition 4.4.1 α is a solved basic formula and thus $z \succ w$. As a consequence, w cannot be a quantified variable otherwise the formula φ does not respect the discipline of the formulas in T. Thus is enough to instantiate z and w by two distinct values.
- If $z = f(\bar{w})$ is a sub-formula of α then it is enough to instantiate z by a tree which starts by a function symbol which is different from f.
- If $w = z$ or $w = t[z]$ is a sub-formula of α then according to Definition 4.4.1 all the variables of $\bar{x}$ are reachable in $\exists \bar{x}\, \alpha$ and thus according to the first point of Remark 4.2.8 the equations $w = z$ and $w = t[z]$ are reachable. According to the second point of Remark 4.2.8 the value of z is linked to another free variable v which occurs in

[15] Recall that $\mathcal{T}r$ is the model of finite or infinite trees.

a left hand side of an equation of α. This case is already treated in two preceding cases.

- If *finite*(z) is a sub-formula of α then it is enough to instantiate z by an infinite tree.

As a consequence, the instantiated formula of $\exists \bar{x}\, \alpha$ will be false in $\mathcal{T}r$ and thus $\mathcal{T}r \models \varphi'$. On the other hand, by following the same preceding steps and since:
(i) the set F contains an infinity of function symbols which are not constants,
(ii) $\mathcal{T}r$ contains an infinity of individuals u of $\mathcal{T}r$ such that $\mathcal{T}r \models \mathit{finite}^{\mathcal{T}r}(u)$,
(iii) φ is a general solved formula,
then we show that there exists at least one instantiation which satisfies all the constraints of α and contradicts the constraints of each β_i, with $i \in I$. In fact, (iv) in order to contradicts each constraint β_i, it is enough to follow the preceding discussion (by replacing α by β_i) and use (i) and (ii). On the other hand, according to Definition 4.4.1 all the variables of $\bar{x}$ are reachable in $\exists \bar{x}\, \alpha$, thus according to the first point of remark 4.2.8 all the equations and relations of α are reachable in $\exists \bar{x}\, \alpha$. According to the second point of remark 4.2.8 the values of the free variables which occur in these formulas are mainly linked to those of free variables which occur in left hand side of equations of α. According to the two first conditions of Definition 4.4.1, the variables of $Lhs(\alpha)$ are distinct and do not occur in $FINI(\alpha)$, $Lhs(\beta_i)$ and $FINI(\beta_i)$ for all $i \in \{1, ..., n\}$. As a consequence, from (iv) and using (i), (ii) and (iii) there exists at least one instantiation which satisfies $\exists \bar{x}\, \alpha$ and contradicts each $\exists \bar{y}_i\, \beta_i$ in $\mathcal{T}r$, with $i \in I$ and thus $\mathcal{T}r \models \neg\varphi''$. Note that if $I = \emptyset$ then we have also $\mathcal{T}r \models \neg\varphi''$ and $\mathcal{T}r \models \varphi'$ using the preceding instantiations.

Case 2: If $\exists \bar{x}\, \alpha$ does not contain free variables then according to the first and third condition of Definition 4.4.1 and Property 4.2.9 we have $\bar{x} = \varepsilon$ and $\alpha = \mathit{true}$. Since φ contains at least one free variable then $I \neq \emptyset$. Let $k \in I$. Since:
(i) the set F contains an infinity of function symbols which are not constants,
(ii) $\mathcal{T}r$ contains an infinity of individuals u of $\mathcal{T}r$ such that $\mathcal{T}r \models \mathit{finite}^{\mathcal{T}r}(u)$,
(iii) φ is a general solved formula,
then we can easily find an instantiation of the free variables of $\exists \bar{y}_k\, \beta_k$ which satisfies the constraints of β_k (similar to the second part of Case 1 by replacing α by β_k). Such an instantiation makes false the instantiated formula $\neg(\exists \bar{y}_k\, \beta_k)$ in $\mathcal{T}r$ and thus $\mathcal{T}r \models \varphi'$. On the other hand, according to (i), (ii) and (iii), we show that there exists at least one instantiation which contradicts the constraints of each β_i, with $i \in I$ (similar to the second part of Case 1 with $\alpha = \mathit{true}$ and $\bar{x} = \varepsilon$). As a consequence, this instantiation satisfies all the $\neg(\exists \bar{y}_i\, \beta_i)$ in $\mathcal{T}r$, with $i \in I$ and thus $\mathcal{T}r \models \neg\varphi''$.

From Case 1 and Case 2, we have $\mathcal{T}r \models \varphi'$ and $\mathcal{T}r \models \neg\varphi''$, and thus neither $T \models \varphi$ nor $T \models \neg\varphi$.
□

Example 4.4.4
Let v_1, v_2, v, u and w be variables such that $v_1 \succ v_2 \succ v \succ u \succ w$. Let φ be the

following general solved formula

$$\neg(\exists v\, u = g(v,w) \wedge \neg(\exists v_1\, v = g(v,v_1) \wedge v_1 = f(v)) \wedge \neg(\exists v_2\, w = g(w,v_2) \wedge v_2 = f(w)) \tag{27}$$

Let us consider for example the model $\mathcal{T}r$ of finite or infinite trees. If we instantiate the free variable u by the finite tree 1 where 1 is a constant in F which is distinct from g then according to axiom [1] of conflict of symbols, the instantiated formula of (27) is true in $\mathcal{T}r$. On the other hand, if u is instantiated by a tree of the form $g(v^*, w^*)$ with $v^* \neq g(v^*, f(v^*))$ (for example $v^* = 1$) and $w^* \neq g(w^*, f(w^*))$ (for example $w^* = 1$) then the instantiated formula of (27) is false in $\mathcal{T}r$. As a consequence (27) is neither true nor false in the theory T. The reader should not think that the fact that we have neither $T \models \neg\varphi$ nor $T \models \varphi$ means that φ is unsatisfiable in T. This is of course false. In fact, since neither $T \models \neg\varphi$ nor $T \models \varphi$ then φ has in each model $\mathcal{M}$ of T a set of solutions which make it true in $\mathcal{M}$ and another set of non-solutions which make it false in $\mathcal{M}$. We also remind the reader that all the properties given after Section 4.1 hold only for formulas that respect the discipline of the formulas of T.

A similar property has been shown for the finite trees of J. Lassez (Lassez and Marriott 1987) and the rational trees of M. Maher (Maher and Stuckey 1995). M. Maher in (Maher and Stuckey 1995) has also shown that if the set F is finite and contains at least one n-ary function symbol with $n \geq 2$, then the problem of deciding if a formula containing equations and the logical symbols $\wedge$, $\vee$, $\neg$ is equivalent to a disjunction of conjunctions of equations is a co-NP-complete problem, and the problem of deciding if an expression represents a nonempty set of rational trees is NP-complete. Note also that in all our proofs we have not used the famous independence of inequations (Colmerauer 1984; Lassez et al. 1986; Comon 1988; Lassez and McAloon 1986) but only the condition that the signature of T is infinite and contains an infinity of function symbols which are not constants and at least one symbol which is a constant, which implies in this case the independence of the inequations.

Property 4.4.5
Every general solved formula of the form $\neg(\exists\bar{x}\,\alpha \wedge \bigwedge_{i=1}^{n} \neg(\exists\bar{y}_i\,\beta_i))$ is equivalent in T to the following Boolean combination of existentially quantified basic formulas:

$$(\neg(\exists\bar{x}\,\alpha)) \vee \bigvee_{i=1}^{n} (\exists\bar{x}\bar{y}_i\,\alpha \wedge \beta_i).$$

Proof
Let

$$\neg(\exists\bar{x}\,\alpha \wedge \bigwedge_{i=1}^{n} \neg(\exists\bar{y}_i\,\beta_i)), \tag{28}$$

be a general solved formula. According to the third point of Definition 4.4.1, all the variables of $\bar{x}$ are reachable in $\exists\bar{x}\,\alpha$. Thus, according to Property 4.2.10, we have

$T \models \exists ? \bar{x}\, \alpha$. According to Property 3.1.10, the formula (28) is equivalent in T to

$$\neg((\exists \bar{x}\, \alpha) \wedge \bigwedge_{i=1}^{n} \neg(\exists \bar{x}\, \alpha \wedge (\exists \bar{y}_i\, \beta_i))),$$

i.e. to

$$(\neg(\exists \bar{x}\, \alpha)) \vee \bigvee_{i=1}^{n} (\exists \bar{x}\, \alpha \wedge (\exists \bar{y}_i\, \beta_i)),$$

which, since the quantified variables have distinct names and different from those of the free variables, is equivalent in T to

$$(\neg(\exists \bar{x}\, \alpha)) \vee \bigvee_{i=1}^{n} (\exists \bar{x} \bar{y}\, \alpha \wedge \beta_i),$$

which is a Boolean combination of existentially quantified basic formulas. $\square$

Definition 4.4.6
Let φ be a formula of the form

$$\exists \bar{x}\, \alpha \wedge \bigwedge_{i=1}^{n} \neg(\exists \bar{y}_i\, \beta_i), \tag{29}$$

with $\bar{x}$ and $\bar{y}$ two vectors of variables, $n \geq 0$ and α and the β_i, with $i \in \{1, ..., n\}$, basic formulas. We say that φ is written in an *explicit solved form* if and only if the formula $\neg\varphi$, i.e.

$$\neg(\exists \bar{x}\, \alpha \wedge \bigwedge_{i=1}^{n} \neg(\exists \bar{y}_i\, \beta_i)), \tag{30}$$

is a general solved formula.

This definition shows how to easily extract from a general solved formula, a simple formula φ which has only one level of negation and where the solutions of the free variables are given in clear and explicit way, i.e. for each model $\mathcal{M}$ of T, it is easy to find all the possible instantiations of the free variables of φ which make it true in $\mathcal{M}$. In fact, according to Definition 4.4.1, we warrant among other things that the left hand sides of the equations of α are distinct and do not occur in those of each β_i, the left hand sides of the equations of each β_i are distinct and we cannot eliminate any quantification since all the variables are reachable.

Example 4.4.7
Let w, v, u_1, u_2, u_3 be variables such that $w \succ v \succ u_1 \succ u_2 \succ u_3$. Let φ be the following general solved formula

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

According to Definition 4.4.6, the following formula ϕ is written in an explicit solved form:

$$\exists v\, u_1 = f(v) \wedge v = u_2 \wedge \mathit{finite}(u_2) \wedge \neg(\exists w\, u_2 = f(w) \wedge \mathit{finite}(w) \wedge \mathit{finite}(u_3)). \tag{31}$$

Let us chose the model $\mathcal{T}r$ of finite or infinite trees and let us give all the possible instantiations u_1^*, u_2^*, u_3^* of the free variables u_1, u_2, u_3 so that the instantiated formula of ϕ is true in the model $\mathcal{T}r$. From (31) it is clear that we have two possibilities:

- Solution 1:
  - u_3^* is any infinite tree.
  - u_2^* is any finite tree.
  - u_1^* is the tree $f(u_2^*)$.
- Solution 2:
  - u_3^* is any finite tree.
  - u_2^* is any finite tree which starts by a function symbol which is different from f.
  - u_1^* is the tree $f(u_2^*)$.

4.5 Working formula

Definition 4.5.1
A working formula is a normalized formula in which all the occurrences of $\neg$ are replaced by $\neg^k$ with $k \in \{0, ..., 5\}$ and such that each occurrence of a sub-formula of the form

$$p = \neg^k(\exists \bar{x}\, \alpha \wedge q), \quad \text{with} \;\; k > 0, \tag{32}$$

satisfies the k first conditions of the condition list bellow. In (32) α is a basic formula, q is a conjunction of working formulas of the form $\bigwedge_{i=1}^{n} \neg^{k_i}(\exists \bar{y}_i\, \beta_i \wedge q_i)$, with $n \geq 0$, β_i a basic formula, q_i a conjunction of working formulas, and in the below condition list α' is the basic formula of the immediate top-working formula[16] p' of p if it exists.

1. If p' exists then $T \models \alpha \rightarrow \alpha'$ and $T \models \alpha_{eq} \rightarrow \alpha'_{eq}$ where α_{eq} and α'_{eq} are the conjunctions of the equations of α respectively α'. Moreover, the set of the variables of $Lhs(\alpha') \cup FINI(\alpha')$ is included in those of $Lhs(\alpha) \cup FINI(\alpha)$.
2. The left hand sides of the equations of α are distinct and for all equations of the form $u = v$ we have $u \succ v$.
3. α is a basic solved formula.
4. If p' exists then the set of the equations of α' is included in those of α.
5. The variables of $\bar{x}$, the equations of α and the constraints of the form *finite*(x) of α are reachable in $\exists \bar{x}\, \alpha$. Moreover, if $n > 0$ then for all $i \in \{1, ..., n\}$ the conjunction β_i contains at least one atomic formula which does not occur in α.

The intuitions behind these working formulas come from an aim to have a full control on the execution of our rewriting rules by adding semantic informations on

[16] In other words, p' is of the form $\neg^{k'}(\exists \bar{x}'\, \alpha' \wedge p^* \wedge p)$ where p^* is a conjunction of working formulas and p is the formula (32).

a syntactic form of formulas. We emphasize strongly that $\neg^k$ does not mean that the normalized formula satisfies only the k^{th} condition but all the conditions i with $1 \leq i \leq k$.

Example 4.5.2
Let w_1, w_2, w_3, v_1, u be variables such that $w_1 \succ w_2 \succ w_3 \succ v_1 \succ u$. This is a working formula of depth 2:

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

Definition 4.5.3
An *initial* working formula is a working formula which begins with $\neg^4$ and such that $k = 0$ for all the other occurrences of $\neg^k$. A *final* working formula is a working formula of depth less or equal to 2 with $k = 5$ for all the occurrences of $\neg^k$.

The relation between the final working formulas and the general solved formulas is expressed in the following property:

Property 4.5.4
Let p be the following final working formula

$$\neg^5(\exists \bar{x}\, \alpha \land \bigwedge_{i=1}^{n} \neg^5(\exists \bar{y}_i\, \beta_i)).$$

The formula

$$\neg(\exists \bar{x}\, \alpha \land \bigwedge_{i=1}^{n} \neg(\exists \bar{y}_i\, \beta_i^*)),$$

is a general solved formula equivalent to p in T where β_i^* is the basic formula β_i from which we have removed all the equations which occur also in α.

Example 4.5.5
Let w_2, v, u and u_1 be variables such that $w_2 \succ v \succ u \succ u_1$. Let φ be the following final working formula

$$\neg^5 \left[\begin{array}{l} \exists \varepsilon\, v = u \land \mathit{finite}(u)\land \\ \neg^5(\exists \varepsilon\, v = u \land u = u_1 \land \mathit{finite}(u_1))\land \\ \neg^5(\exists w_2\, v = u \land u = s(w_2) \land \mathit{finite}(w_2)) \end{array} \right].$$

The formula

$$\neg \left[\begin{array}{l} \exists \varepsilon\, v = u \land \mathit{finite}(u)\land \\ \neg(\exists \varepsilon\, u = u_1 \land \mathit{finite}(u_1))\land \\ \neg(\exists w_2\, u = s(w_2) \land \mathit{finite}(w_2)) \end{array} \right].$$

is a general solved formula equivalent to φ in T.

4.6 Rewriting rules

We now present the rewriting rules which transform an initial working formula of any depth d into an equivalent conjunction of final working formulas. 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 the following, the letters u, v and w represent variables, the letters $\bar{x}$, $\bar{y}$ and $\bar{z}$ represent vectors of variables, the letters a, b and c represent basic formulas, the letter q represents a conjunction of working formulas, the letter r represents a conjunction of flat equations, formulas of the form *finite*(x) and working formulas. All these letters can be subscripted or have primes.

$$
\begin{array}{llcc}
(1) & \neg^1(\exists \bar{x}\, u = u \wedge r) & \Longrightarrow & \neg^1(\exists \bar{x}\, r) \\
(2) & \neg^1(\exists \bar{x}\, v = u \wedge r) & \Longrightarrow & \neg^1(\exists \bar{x}\, u = v \wedge r) \\
(3) & \neg^1(\exists \bar{x}\, u = v \wedge u = t \wedge r) & \Longrightarrow & \neg^1(\exists \bar{x}\, u = v \wedge v = t \wedge r) \\
(4) & \neg^1(\exists \bar{x}\, u = fv_1...v_n \wedge u = gw_1...w_m \wedge r) & \Longrightarrow & \mathit{true} \\
(5) & \neg^1(\exists \bar{x}\, u = fv_1...v_n \wedge u = fw_1...w_n \wedge r) & \Longrightarrow & \neg^1(\exists \bar{x}\, u = fv_1...v_n \wedge \bigwedge_{i=1}^n v_i = w_i \wedge r) \\
(6) & \neg^1(\exists \bar{x}\, a \wedge q) & \Longrightarrow & \neg^2(\exists \bar{x}\, a \wedge q) \\
(7) & \neg^2(\exists \bar{x}\, \mathit{finite}(u) \wedge \mathit{finite}(u) \wedge r) & \Longrightarrow & \neg^2(\exists \bar{x}\, \mathit{finite}(u) \wedge r) \\
(8) & \neg^2(\exists \bar{x}\, u = v \wedge \mathit{finite}(u) \wedge r) & \Longrightarrow & \neg^2(\exists \bar{x}\, u = v \wedge \mathit{finite}(v) \wedge r) \\
(9) & \neg^2(\exists \bar{x}\, \mathit{finite}(u) \wedge a \wedge q) & \Longrightarrow & \mathit{true} \\
(10) & \neg^2(\exists \bar{x}\, u = f(v_1, ..., v_n) \wedge \mathit{finite}(u) \wedge r) & \Longrightarrow & \neg^2(\exists \bar{x}\, u = f(v_1, ..., v_n) \wedge \bigwedge_{i=1}^n \mathit{finite}(v_i) \wedge r) \\
(11) & \neg^2(\exists \bar{x}\, a \wedge q) & \Longrightarrow & \neg^3(\exists \bar{x}\, a \wedge q) \\
(12) & \neg^4(\exists \bar{x}\, a \wedge q \wedge \neg^0(\exists \bar{y}\, r)) & \Longrightarrow & \neg^4(\exists \bar{x}\, a \wedge q \wedge \neg^1(\exists \bar{y}\, a \wedge r)) \\
(13) & \neg^4(\exists \bar{x}\, a \wedge a' \wedge q \wedge \neg^3(\exists \bar{y}\, a'' \wedge r)) & \Longrightarrow & \neg^4(\exists \bar{x}\, a \wedge a' \wedge q \wedge \neg^4(\exists \bar{y}\, a \wedge r)) \\
(14) & \neg^4(\exists \bar{x}\, a \wedge q \wedge \neg^5(\exists \bar{y}\, a)) & \Longrightarrow & \mathit{true} \\
(15) & \neg^4(\exists \bar{x}\, a \wedge \bigwedge_{i=1}^n \neg^5(\exists \bar{y}_i\, b_i)) & \Longrightarrow & \neg^5(\exists \bar{x}'\, a' \wedge \bigwedge_{i \in K} \neg^5(\exists \bar{y}'_i\, b'_i)^*) \\
(16) & \neg^4 \begin{bmatrix} \exists \bar{x}\, a \wedge q \wedge \\ \neg^5 \begin{bmatrix} \exists \bar{y}\, b \wedge \\ \bigwedge_{i=1}^n \neg^5(\exists \bar{z}_i\, c_i) \end{bmatrix} \end{bmatrix} & \Longrightarrow & \begin{bmatrix} \neg^4(\exists \bar{x}\, a \wedge q \wedge \neg^5(\exists \bar{y}\, b)) \wedge \\ \bigwedge_{i=1}^n \neg^4(\exists \bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)^* \end{bmatrix}
\end{array}
$$

with $u \succ v$, f and g two distinct function symbols taken from F. In rule (3), t is a flat term, i.e. either a variable or a term of the form $f(x_1, ..., x_n)$ with f an n-ary function symbol taken from F. In rule (6), the equations of a have distinct left hand sides and for each equation of the form $u = v$ we have $u \succ v$. In rule (9), the variable u is reachable from u in a. In rule (10), the variable u is non-reachable from u in a. Moreover, if f is a constant then $n = 0$. In rule (11), a is a solved basic formula. In rule (13), a and a'' are conjunctions of equations having the same left hand sides and a' is a conjunction of formulas of the form *finite*(u). In rule (15), $n \geq 0$ and for all $i \in \{1, ..., n\}$ the formula b_i is different from the formula a. The pairs $(\bar{x}', a')$ and $(\bar{y}'_i, b'_i)$ are obtained by a decomposition of $\bar{x}$ and a into $\bar{x}'\bar{x}''\bar{x}'''$ and $a' \wedge a'' \wedge a'''$ as follows:

- a' is the conjunction of the equations and the formulas of the form $\mathit{finite}(x)$ which are reachable in $\exists\bar{x}\, a$.
- $\bar{x}'$ is the vector the variables of $\bar{x}$ which are reachable in $\exists\bar{x}\, a$.
- a'' is the conjunction of the formulas of the form $\mathit{finite}(x)$ which are non-reachable in $\exists\bar{x}\, a$.
- $\bar{x}''$ is the vector the variables of $\bar{x}$ which are non-reachable in $\exists\bar{x}\, a$ and do not occur in the left hand sides of the equations of a.
- a''' is the conjunction of the equations which are non-reachable in $\exists\bar{x}\, a$.
- $\bar{x}'''$ is the vector the variables of $\bar{x}$ which are non-reachable in $\exists\bar{x}\, a$ and occur in the left hand sides of the equations of a.
- b_i^* is the formula obtained by removing from b_i the formulas of the form $\mathit{finite}(u)$ which occur also in a''
- $\bar{y}_i'$ is the vector of the variables of $\bar{y}_i\bar{x}'''$ which are reachable in $\exists\bar{y}_i\bar{x}'''\, b_i^*$.
- b_i' is the conjunction of the equations and the formulas of the form $\mathit{finite}(x)$ which are reachable in $\exists\bar{y}_i\bar{x}'''\, b_i^*$.
- $K \subseteq \{1, ..., n\}$ is the set of the indices i such that $i \in K$ if and only if no variable of $\bar{x}''$ occurs in b_i'.
- The formula $\bigwedge_{i\in K} \neg^5(\exists\bar{y}_i'\, b_i')^*$ is the formula $\bigwedge_{i\in K} \neg^5(\exists\bar{y}_i'\, b_i')$ in which we have renamed the quantified variables so that they satisfy the discipline of the formulas in T.

In rule (16), $n > 0$ and q_0 is the formula q in which all the occurrences of $\neg^k$ have been replaced by $\neg^0$. The formula $\bigwedge_{i=1}^n \neg^4(\exists\bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)^*$ is the formula $\bigwedge_{i=1}^n \neg^4(\exists\bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)$ in which we have renamed the quantified variables so that they satisfy the discipline of the formulas of T.

The use of indices on the negations of the working formulas enables us to force the application of the rules to follow a clear strategy until reaching a conjunction of final working formulas. In fact, the algorithm follows two main steps while solving any first-order constraint in T:

- (i) A top-down propagation of basic formulas following the tree structure of the working formulas and using the rules (1),...,(13). In this step, basic formulas are solved and copied in all sub-working formulas. Finiteness is also check and inconsistent basic formulas are removed by the rules (4) and (9).
- (ii) A bottom-up elimination of quantifiers and depth reducing of the working formulas using the rules (14),...,(16). Inconsistent working formulas are also removed in this step.

More precisely, starting from an initial working formula φ of the form $\neg^4(\exists\bar{x}\, a \wedge \bigwedge_{i\in I} q_i)$, where all the q_i are working formulas whose negations are of the form $\neg^0$, rule (12) propagates the atomic formulas of a into a sub-formula q_i, with $i \in I$, and changes the first negation of q_i into $\neg^1$. The rules (1),...,(5) can now be applied until the equations of a have distinct left hand sides and for each equation of the form $u = v$ we have $u \succ v$. Rule (6) is then applied and changes the first negation of q_i into $\neg^2$. The algorithm starts now a new phase which consists in solving the basic formulas using the rules (7),...,(10). In particular finiteness is checked by rule (9). When a solved basic formula is obtained, rule (11) is applied and changes the negation into $\neg^3$. Note that if a working formula starts by $\neg^3$ then its top working formula starts by $\neg^4$. Rule (13) is then applied. It restores some equations and changes the first negation into $\neg^4$. Rule (12) can now be applied again since all the nested negations are of the form $\neg^0$ and so on. This is the first step of our algorithm. Once the sub-working formulas of depth 1 are of the form $\neg^4(\exists \bar{y}_i\, b_i)$, the second step starts using rule (15) with $n = 0$ on all these sub-working-formulas of depth 1 and transforms their negations into $\neg^5$. Inconsistent working formulas of the form $\neg^4(\exists \bar{x}\, \alpha \wedge \neg^5(\exists \bar{y}\, \alpha) \wedge q)$ are then removed by rule (14). When all the inconsistent working formulas have been removed, rule (15) with $n \neq 0$ can be applied on the sub-working-formulas of depth 2 of the form $\neg^4(\exists \bar{x}\, a \wedge \bigwedge_{i \in I} \neg^5(\exists \bar{y}_i\, b_i))$ and produces working formulas of the form $\neg^5(\exists \bar{x}\, a \wedge \bigwedge_{i \in I} \neg^5(\exists \bar{y}_i\, b_i))$. Rule (16) can now be applied on the working formulas of depth $d > 2$ of the form $\neg^4(\exists \bar{x}\, a \wedge q \wedge \neg^5(\exists \bar{y}\, b \wedge \bigwedge_{i=1}^n \neg^5(\exists \bar{z}_i\, c_i)))$. After each application of this rule, new working formulas containing negations of the form $\neg^0$ are created which implies the execution of the rules of the first step of our algorithm, starting by rule (12) and so on. After several applications of our rules, we get a conjunction of working formulas whose depth is less or equal to 2. The rules are then applied again until all the negations of these working formulas are of the form $\neg^5$. It is a conjunction of final working formulas.

Example 4.6.1
Let f and g be two function symbols taken from F of respective arities $2, 1$. Let w_1, w_2, v_1, u_1, u_2, u_3 be variables such that $w_1 \succ w_2 \succ v_1 \succ u_1 \succ u_2 \succ u_3$. Let us run our rules on the following initial working formula

$$\neg^4 \begin{bmatrix} \exists v_1\, v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^0(\exists w_1\, v_1 = g(w_1)) \wedge \\ \neg^0(\exists w_2\, u_2 = g(w_2) \wedge w_2 = g(u_3) \wedge \mathit{finite}(w_2)) \end{bmatrix}. \tag{33}$$

According to rule (12), the preceding formula is equivalent in T to

$$\neg^4 \begin{bmatrix} \exists v_1\, v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^1(\exists w_1\, v_1 = g(w_1) \wedge v_1 = f(u_1, u_2) \wedge u_2 = g(u_1)) \wedge \\ \neg^0(\exists w_2\, u_2 = g(w_2) \wedge w_2 = g(u_3) \wedge \mathit{finite}(w_2)) \end{bmatrix}.$$

The application of rule (4) on the sub formula $\neg^1(\exists w_1\, v_1 = g(w_1) \wedge v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge \mathit{finite}(w_2))$ simplifies this sub formula into the formula *true*. Thus, the preceding formula is equivalent in T to

$$\neg^4 \begin{bmatrix} \exists v_1\, v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^0(\exists w_2\, u_2 = g(w_2) \wedge w_2 = g(u_3) \wedge \mathit{finite}(w_2)) \end{bmatrix},$$

which according to rule (12) is equivalent in T to

$$\neg^4 \begin{bmatrix} \exists v_1\, v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^0(\exists w_2\, v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge u_2 = g(w_2) \wedge w_2 = g(u_3) \wedge \mathit{finite}(w_2)) \end{bmatrix}.$$

Rule (5) can now be applied. Thus, the preceding formula is equivalent in T to

$$\neg^4 \begin{bmatrix} \exists v_1\, v_1 = f(u_1, u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^1(\exists w_2\, v_1 = f(u_1, u_2) \wedge u_2 = g(w_2) \wedge w_2 = u_1 \wedge w_2 = g(u_3) \wedge \mathit{finite}(w_2)) \end{bmatrix},$$

which according to rule (3) is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^1(\exists w_2\, v_1 = f(u_1,u_2) \wedge u_2 = g(w_2) \wedge w_2 = u_1 \wedge u_1 = g(u_3) \wedge \mathit{finite}(w_2)) \end{array} \right].$$

Since the conjunction of equations of the sub-formula which starts by $\neg^1$ has distinct left hand sides and $w_2 \succ u_1$, then rule (6) can be applied. Thus, the preceding formula is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^2(\exists w_2\, v_1 = f(u_1,u_2) \wedge u_2 = g(w_2) \wedge w_2 = u_1 \wedge u_1 = g(u_3) \wedge \mathit{finite}(w_2)) \end{array} \right],$$

which according to rule (8) is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^2(\exists w_2\, v_1 = f(u_1,u_2) \wedge u_2 = g(w_2) \wedge w_2 = u_1 \wedge u_1 = g(u_3) \wedge \mathit{finite}(u_1)) \end{array} \right],$$

which according to rule (10) is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^2(\exists w_2\, v_1 = f(u_1,u_2) \wedge u_2 = g(w_2) \wedge w_2 = u_1 \wedge u_1 = g(u_3) \wedge \mathit{finite}(u_3)) \end{array} \right].$$

Since the basic formulas are solved then rule (11) can be applied. Thus, the preceding formula is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^3(\exists w_2\, v_1 = f(u_1,u_2) \wedge u_2 = g(w_2) \wedge w_2 = u_1 \wedge u_1 = g(u_3) \wedge \mathit{finite}(u_3)) \end{array} \right],$$

which according to rule (13) is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^4(\exists w_2\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge w_2 = u_1 \wedge u_1 = g(u_3) \wedge \mathit{finite}(u_3)) \end{array} \right].$$

Rule (15) can now be applied with $n = 0$. Thus, the preceding formula is equivalent in T to

$$\neg^4 \left[\begin{array}{l} \exists v_1\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge \\ \neg^5(\exists \varepsilon\, v_1 = f(u_1,u_2) \wedge u_2 = g(u_1) \wedge u_1 = g(u_3) \wedge \mathit{finite}(u_3)) \end{array} \right].$$

Once again rule (15) can be applied, with $n \neq 0$ and we get the following final working formula

$$\neg^5 \left[\begin{array}{l} \exists \varepsilon\, u_2 = g(u_1) \wedge \\ \neg^5(\exists \varepsilon\, u_2 = g(u_1) \wedge u_1 = g(u_3) \wedge \mathit{finite}(u_3)) \end{array} \right],$$

which according to Property 4.5.4 is equivalent in T to the following general solved formula

$$\neg \left[\begin{array}{l} u_2 = g(u_1) \wedge \\ \neg(u_1 = g(u_3) \wedge \mathit{finite}(u_3)) \end{array} \right].$$

We have seen in the preceding example how the rules (1),...,(15) can be applied. Let us now see how rule (16) is applied.

Example 4.6.2
Let s and 0 be two function symbols taken from F of respective arities $1, 0$. Let w_1, w_2, u, v be variables such that $w_1 \succ w_2 \succ v \succ u$. Let us apply our rules on the following working formula of depth 3:

$$\neg^4 \left[\exists\varepsilon\, \mathit{true} \wedge \left[\begin{array}{l} \neg^5(\exists\varepsilon\, u = s(v)) \wedge \\ \neg^5(\exists w_1\, u = s(w_1) \wedge w_1 = s(v)) \wedge \\ \neg^5(\exists\varepsilon\, v = u \wedge \neg^5(\exists\varepsilon\, v = u \wedge u = 0) \wedge \neg^5(\exists w_2\, v = u \wedge u = s(w_2))) \end{array} \right] \right].$$

By considering that

- $(\exists\bar{x}\, a) = (\exists\varepsilon \mathit{true})$
- $q = \left[\begin{array}{l} \neg^5(\exists\varepsilon\, u = s(v)) \wedge \\ \neg^5(\exists w_1\, u = s(w_1) \wedge w_1 = s(v)) \end{array} \right]$
- $(\exists\bar{y}\, b) = (\exists\varepsilon\, v = u)$
- $\bigwedge_{i=1}^{n} \neg^5(\exists\bar{z}_i\, c_i) = \left[\begin{array}{l} \neg^5(\exists\varepsilon\, v = u \wedge u = 0) \wedge \\ \neg^5(\exists w_2\, v = u \wedge u = s(w_2)) \end{array} \right]$

rule (16) can be applied and produces the following formula

$$\left[\begin{array}{l} \neg^4(\exists\varepsilon\, \mathit{true} \wedge \neg^5(\exists\varepsilon\, u = s(v)) \wedge \neg^5(\exists w_1\, u = s(w_1) \wedge w_1 = s(v)) \wedge \neg^5(\exists\varepsilon\, v = u)) \wedge \\ \neg^4(\exists\varepsilon\, v = u \wedge u = 0 \wedge \neg^0(u = s(v)) \wedge \neg^0(\exists w_{11}\, u = s(w_{11}) \wedge w_{11} = s(v))) \wedge \\ \neg^4(\exists w_2\, v = u \wedge u = s(w_2) \wedge \neg^0(\exists\varepsilon\, u = s(v)) \wedge \neg^0(\exists w_{12}\, u = s(w_{12}) \wedge w_{12} = s(v))) \end{array} \right],$$

where w_{11} and w_{12} are variables such that $w_{11} \succ w_{12} \succ w_1 \succ w_2 \succ v \succ u$. Now, only the rules (1),...,(15) will be applied until all the negations are of the form $\neg^5$. Rule (16) will not be applied anymore since there exists no working formulas of depth greater or equal to 3 and the rules (1),...,(15) never increase the depth of the working formulas.

Property 4.6.3
Every repeated application of the preceding rewriting rules on an initial working formula p is terminating and producing a wnfv conjunction of final working formulas equivalent to p in T.

Proof

Proof, first part: The application of the rewriting rules terminates. Let us introduce the function $\alpha : q \rightarrow n$, where q is a conjunction of working formulas, n an integer and such that

- $\alpha(\mathit{true}) = 0$,
- $\alpha(\neg(\exists\bar{x}\, a \wedge \varphi)) = 2^{\alpha(\varphi)}$,
- $\alpha(\bigwedge_{i \in I} \varphi_i) = \sum_{i \in I} \alpha(\varphi_i)$,

with a a basic formula, φ a conjunction of working formulas and the φ_i's working formulas. Note that if $\alpha(p_2) < \alpha(p_1)$ then $\alpha(p[p_2]) < \alpha(p)$ where $p[p_2]$ is the formula obtained from p when we replace the occurrence of the formula p_1 in p by p_2. This function has been introduced in (Vorobyov 1996) and (Colmerauer and Dao 2003) to show the non-elementary complexity of all algorithms solving propositions in the theory of finite or infinite trees. It has also the property to decrease if the depth of the working formula decreases after application of distributions as it is done in our rule (16).

Let us introduce also the function $\lambda : (u, a) \rightarrow n$, where u is a variable, a a basic formula, n an integer and such that

$$\lambda(u,a) = \begin{bmatrix} 0, & \text{if the conjunction of the equations of } a \text{ has} \\ & \text{not distinct left hand sides or contains a} \\ & \text{sub-formula of the form } x = y \text{ with } y \succ x, \textbf{else} \\ 1, & \text{if } u \text{ does not occur in a left hand side of an equation} \\ & \text{of } a, \text{ or } u \text{ is reachable from } u \text{ in } a, \textbf{else} \\ 1 + \lambda(v,a), & \text{if the equation } u = v \text{ is in } a, \textbf{else} \\ 2 + \sum_{i=1}^{n} \lambda(v_i, a), & \text{if the equation } u = f(v_1, ..., v_n) \text{ is in } a. \end{bmatrix}$$

Since the variables which occur in our formulas are ordered by the order relation " $\succ$ ", we can number them by positive integers such that

$$x \succ y \leftrightarrow no(x) > no(y),$$

where $no(x)$ is the number associated to the variable x. Let us consider the 10-tuple $(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10})$ where the n_i's are the following positive integers:

- $n_1 = \alpha(p)$,
- n_2 is the number of $\neg^0$,
- n_3 is the number of $\neg^1$,
- n_4 is the number of occurrences of function symbols in sub-formulas of the form $\neg^1(...)$. For example, if we have $\neg^1(\exists x\, x = f(y) \wedge y = f(x) \wedge x = g(x,w) \wedge y = f(y))$ then $n_4 = 4$.
- n_5 is the sum of all the $no(x)$ for each occurrence of a variable x in a basic formula of a sub-formula of the form $\neg^1(...)$. For example, if we have $\neg^1(\exists w\, x = f(x,z) \wedge y = x \wedge \mathit{finite}(z) \wedge ...)$ then $n_5 = no(x) + no(x) + no(z) + no(y) + no(x) + no(z) + ...$
- n_6 is the number of formulas of the form $v = u$ with $u \succ v$ in sub-formulas of the form $\neg^1(...)$,
- n_7 is the number of $\neg^2$,
- n_8 is the sum of all the $\lambda(u,a)$ for each occurrence of a sub-formula $\mathit{finite}(u)$ in a basic-formula a of a working formula of the form $\neg^2(\exists \bar{x}\, a \wedge q)$. For example, if we have $\neg^2(\exists z\, x = f(x,z) \wedge z = f(y,y) \wedge \mathit{finite}(x) \wedge \mathit{finite}(x) \wedge \mathit{finite}(z))$ then $n_8 = \lambda(x,a) + \lambda(x,a) + \lambda(z,a) = 1 + 1 + (2 + 1 + 1)$ where a is the basic formula $x = f(x,z) \wedge z = f(y,y) \wedge \mathit{finite}(x) \wedge \mathit{finite}(x) \wedge \mathit{finite}(z)$.
- n_9 is the number of $\neg^3$
- n_{10} is the number of $\neg^4$.

For each rule, there exists a positive integer i such that the application of this rule decreases or does not change the values of the n_j's, with $1 \leq j < i$, and decreases the value of n_i. These i are equal to: 1 for the rules (4), (9), (14) and (16), 2 for rule (12), 3 for rule (6), 4 for rule (5), 5 for the rules (1), (3), (7) and (8) , 6 for rule (2), 7 for rule (11), 8 for rule (10), 9 for rule (13), and 10 for rule (15). To each sequence of formulas obtained by a finite application of the preceding rewriting rules, we can associate a series of 10-tuples $(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10})$ which is strictly decreasing in the lexicographic order. Since the n_i's are positive integers, they cannot be negative, thus, this series of 10-tuples is a finite series and the application of the rewriting rules terminates.

Proof, second part: Let us now show that for each rule of the form $p \Longrightarrow p'$ we have $T \models p \leftrightarrow p'$ and the formula p' remains a conjunction of working formula.

Correctness of the rules (1),...,(14)

The rules (1),...(5) are correct according to the axioms [1] and [2] of T. Rules (6) and (11) are evident. The rules (7) and (8) are true in the empty theory and thus true in T. In rule (9), the variable u is reachable from itself in a, i.e. the basic formula a contains a sub-formula of the form

$$u = t_1 \wedge u_2 = t_2 \wedge ... \wedge u_n = t_n \tag{34}$$

where u_i occurs in the term t_{i-1} for all $i \in \{2, ..., n\}$ and u occurs in t_n. According to Definition 4.5.1, since our working formula starts with $\neg^2$ then all the equations of a have distinct lef hand sides and for all equations of the form $x = y$ we have $x \succ y$. Thus, there exists at least one equation in (34) which contains a function symbol which is not a constant, otherwise (34) is of the form $u = u_2 \wedge u_2 = u_3 \wedge ... \wedge u_n = u$ which implies $u \succ u_2 \succ ... \succ u$, i.e. $u \succ u$ which is false since the order $\succ$ is strict. Thus, according to the fourth axiom of T we have $T \models a \rightarrow \neg \mathit{finite}(u)$. As a consequence, rule (9) is correct in T. Rule (10) is correct according to the last axiom of T. Rule (13) is correct according to Property 4.2.4 and Definition 4.5.1. The rules (12) and (14) are true in the empty theory and thus true in T. Note that according to Property 4.2.5, two solved basic formulas having the same equations are equivalent if and only if they have the same relations $\mathit{finite}(x)$. This is why in Definition 4.5.1 of the working formulas (more precisely in condition 4) we force only the equations to be included in the sub-forworking formulas and use the elementary rule (14) to remove inconsistent working formulas of depth 2.

Correctness of rule (15)

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

with $n \geq 0$, and for all $i \in \{1, ..., n\}$ the formula b_i is different from the formula a. The pairs $(\bar{x}', a')$ and $(\bar{y}'_i, b'_i)$ are obtained by a decomposition of $\bar{x}$ and a into $\bar{x}'\bar{x}''\bar{x}'''$ and $a' \wedge a'' \wedge a'''$ as follows:

- a' is the conjunction of the equations and the formulas of the form $\mathit{finite}(x)$ which are reachable in $\exists \bar{x}\, a$.

- $\bar{x}'$ is the vector the variables of $\bar{x}$ which are reachable in $\exists \bar{x}\, a$.
- a'' is the conjunction of the formulas of the form *finite*(x) which are non-reachable in $\exists \bar{x}\, a$.
- $\bar{x}''$ is the vector the variables of $\bar{x}$ which are non-reachable in $\exists \bar{x}\, a$ and do not occur in the left hand sides of the equations of a.
- a''' is the conjunction of the equations which are non-reachable in $\exists \bar{x}\, a$.
- $\bar{x}'''$ is the vector the variables of $\bar{x}$ which are non-reachable in $\exists \bar{x}\, a$ and occur in the left hand sides of the equations of a.
- b_i^* is the formula obtained by removing from b_i the formulas of the form *finite*(u) which occur also in a''
- $\bar{y}_i'$ is the vector of the variables of $\bar{y}_i\bar{x}'''$ which are reachable in $\exists \bar{y}_i\bar{x}'''\, b_i^*$.
- b_i' is the conjunction of the equations and the formulas of the form *finite*(x) which are reachable in $\exists \bar{y}_i\bar{x}'''\, b_i^*$.
- $K \subseteq \{1, ..., n\}$ is the set of the indices i such that $i \in K$ if and only if no variable of $\bar{x}''$ occurs in b_i'.
- The formula $\bigwedge_{i\in K} \neg^5(\exists \bar{y}_i'\, b_i')^*$ is the formula $\bigwedge_{i\in K} \neg^5(\exists \bar{y}_i'\, b_i')$ in which we have renamed the quantified variables so that they satisfy the discipline of the formulas in T.

Let $\bar{x}', \bar{x}'', \bar{x}''', \bar{y}'$ and $a', a'', a''', b_i^*, b_i'$ be the vector of variables and the basic formulas defined above. According to Definition 4.2.6, (i) all the variables of $\bar{x}''$ and $\bar{x}'''$ do not occur in a', otherwise they are reachable in $\exists \bar{x}\, a$. On the other hand, since the first negation in the left hand side of rule (15) is of the form $\neg^4$ then according to Definition 4.5.1 (ii) a is a solved basic formula and thus $\bar{x}'''$ is the vector of the left hand sides of the equations of a''' and its variables do not occur in a''. Thus, according to (i) and (ii) the left hand side of rule (15) is equivalent in T to

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

Since a is a solved basic formula then a''' is a solved basic formula which contains only equations and thus according to Property 4.2.3 we have $T \models \exists! \bar{x}'''\, a'''$. Thus, according to Property 3.1.11 the preceding formula is equivalent in T to

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

which, according to the discipline of the formulas in T (the quantified variables have distinct names and different from those of the free variables), is equivalent in T to

$$\neg(\exists \bar{x}'\, a' \wedge (\exists \bar{x}''\, a'' \wedge \bigwedge_{i=1}^{n} \neg(\exists \bar{x}'''\bar{y}_i\, a''' \wedge b_i))). \tag{35}$$

Since all the nested negations in the left hand side of rule (15) are of the form $\neg^5$ then according to Definition 4.5.1, for all $i \in \{1, ..., n\}$, the set of the equations of a

is included in those of b_i. As a consequence, the formula (35) is equivalent in T to

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

i.e. to

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

Since all the nested negations in the left hand side of rule (15) are of the form $\neg^5$, then according to Definition 4.5.1, for all $i \in \{1, ..., n\}$, b_i^* is a solved basic formula. Thus, according to Property 4.2.11, the preceding formula is equivalent in T to

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

which is equivalent in T to

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

where $K \subseteq \{1, ..., n\}$ is the set of the indices i such that $i \in K$ if and only if no variable of $\bar{x}''$ occurs in b_i'. Since all the nested negations in the left hand side of rule (15) are of the form $\neg^5$ then according to Definition 4.5.1, for all $i \in \{1, ..., n\} - K$, the variables of $\bar{y}_i'$ are reachable in $\exists \bar{y}_i'\, b_i'$ and the formula b_i' is a solved basic formula. Moreover, since each b_i' does not contain sub-formulas of the form $\mathit{finite}(x)$ which occur also in a'' (see the construction of b_i^*), then the formula $\exists \bar{x}''\, a'' \wedge \bigwedge_{i\in\{1,...,n\}-K} \neg(\exists \bar{y}'\, b_i')$ satisfies the conditions of Property 4.2.13. As a consequence, according to Property 4.2.13 the preceding formula is equivalent in T to

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

i.e. to

$$\neg(\exists \bar{x}'\, a' \wedge \bigwedge_{i\in K} \neg(\exists \bar{y}_i'\, b_i')^*),$$

where $\bigwedge_{i\in K} \neg^5(\exists \bar{y}_i'\, b_i')^*$ is the formula $\bigwedge_{i\in K} \neg^5(\exists \bar{y}_i'\, b_i')$ in which we have renamed the quantified variables so that they satisfy the discipline of the formulas in T. According to the conditions of application of rule (15) and the form of the negations in the left hand side of this rule, we check easily that we can fix the negations of the preceding formula as follows

$$\neg^5(\exists \bar{x}'\, a' \wedge \bigwedge_{i\in K} \neg^5(\exists \bar{y}_i'\, b_i')^*).$$

Thus, rule (15) is correct in T.

Correctness of rule (16)

$$\neg^4 \left[\begin{array}{l} \exists \bar{x}\, a \wedge q \wedge \\ \neg^5 \left[\begin{array}{l} \exists \bar{y}\, b \wedge \\ \bigwedge_{i=1}^n \neg^5 (\exists \bar{z}_i\, c_i) \end{array} \right] \end{array} \right] \Longrightarrow \left[\begin{array}{l} \neg^4(\exists \bar{x}\, a \wedge q \wedge \neg^5(\exists \bar{y}\, b)) \wedge \\ \bigwedge_{i=1}^n \neg^4(\exists \bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)^* \end{array} \right]$$

with $n > 0$, and q_0 is the formula q in which all the occurrences of $\neg^k$ have been replaced by $\neg^0$. The formula $\bigwedge_{i=1}^n \neg^4(\exists \bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)^*$ is the formula $\bigwedge_{i=1}^n \neg^4(\exists \bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)$ in which we have renamed the quantified variables so that they satisfy the discipline of the formulas of T.

The left hand side of rule (16) is equivalent in T to

$$\neg(\exists \bar{x}\, a \wedge q \wedge \neg(\exists \bar{y}\, b \wedge \neg \bigvee_{i=1}^n (\exists \bar{z}_i\, c_i))).$$

Since the first negation of $\neg(\exists \bar{y}\, b...$ in the left hand side of rule (16) is of the form $\neg^5$ then according to Definition 4.5.1, all the variables of $\bar{y}$ are reachable in $\exists \bar{y}\, b$, and thus according to Property 4.2.10 we have $T \models \exists ? \bar{y}\, b$. According to Property 3.1.10, the precedent formula is equivalent in T to

$$\neg(\exists \bar{x}\, a \wedge q \wedge \neg((\exists \bar{y}\, b) \wedge \neg(\exists \bar{y}\, b \wedge \bigvee_{i=1}^n (\exists \bar{z}_i\, c_i)))).$$

By distributing the $\wedge$ on the $\vee$ and the $\exists$ on the $\vee$ and since the quantified variables have distinct names and different from those of the free variables then the preceding formula is equivalent in T to

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

i.e. to

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

i.e. to

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

which, according to the discipline of the formulas in T (the quantified variables have distinct names and different from those of the free variables), is equivalent in T to

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

i.e. to

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

i.e. to

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

Since we have $\neg^5(\exists\bar{y}\, b...$ in the left hand side of rule (16) then according to Definition 4.5.1, we have (i) $T \models b \rightarrow a$. But since we have also $\neg^5(\exists\bar{z}_i\, c_i)$ for all $i \in \{1, ..., n\}$, then according to Definition 4.5.1 we have (ii) $T \models c_i \rightarrow b$. From (i) and (ii) we have $T \models c_i \rightarrow (a \wedge b)$. Thus the preceding formula is equivalent in T to

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

i.e. to

$$\neg(\exists\bar{x}\, a \wedge q \wedge \neg(\exists\bar{y}\, b)) \wedge \bigwedge_{i=1}^{n} \neg(\exists\bar{x}\bar{y}\bar{z}_i\, c_i \wedge q)^*,$$

where $\bigwedge_{i=1}^{n} \neg^4(\exists\bar{x}\bar{y}\bar{z}_i\, c_i \wedge q)^*$ is the formula $\bigwedge_{i=1}^{n} \neg^4(\exists\bar{x}\bar{y}\bar{z}_i\, c_i \wedge q)$ in which we have renamed the quantified variables so that they satisfy the discipline of the formulas of T. According to the conditions of application of rule (16) and the form of the negations in the left hand side of this rule, we check easily that we can fix the negations of the preceding formula as follows

$$\neg^4(\exists\bar{x}\, a \wedge q \wedge \neg^5(\exists\bar{y}\, b)) \wedge \bigwedge_{i=1}^{n} \neg^4(\exists\bar{x}\bar{y}\bar{z}_i\, c_i \wedge q_0)^*,$$

where q_0 is the formula q in which all the occurrences of $\neg^k$ have been replaced by $\neg^0$. Thus rule (16) is correct in T.

Proof, third part: Every repeated application of the rewriting rules on an initial working formula produces a conjunction of final working formulas. According to what we have shown in the sub-section "*proof: first part*", every repeated application of our rules on an initial working formula terminates. Let us now show that the obtained formula is a conjunction of final working formulas.

Recall that we write $\bigwedge_{i\in I} \varphi_i$, and call *conjunction* each formula of the form $\varphi_{i_1} \wedge \varphi_{i_2} \wedge ... \wedge \varphi_{i_n} \wedge true$. In particular, for $I = \emptyset$, the conjunction $\bigwedge_{i\in I} \varphi_i$ is reduced to *true*. Moreover, we do not distinguish two formulas which can be made equal using the following transformations of sub-formulas:

$$\varphi \wedge \varphi \Longrightarrow \varphi,\ \ \varphi \wedge \psi \Longrightarrow \psi \wedge \varphi,\ \ (\varphi \wedge \psi) \wedge \phi \Longrightarrow \varphi \wedge (\psi \wedge \phi),$$
$$\varphi \wedge \mathit{true} \Longrightarrow \varphi,\ \ \varphi \vee \mathit{false} \Longrightarrow \varphi.$$

Let us show first that every substitution of a sub-working formula of a conjunction of working formulas by a conjunction of working formulas produces a conjunction of working formulas. Let $\bigwedge_{i\in I} \varphi_i$ be a conjunction of working formulas. Let φ_k with $k \in I$ be an element of this conjunction of depth d_k. Two cases arise:

1. We replace φ_k by a conjunction of working formulas. Thus, let $\bigwedge_{j\in J_k} \phi_j$ be a conjunction of working formulas which is equivalent to φ_k in T. The conjunction of

working formulas $\bigwedge_{i\in I}\varphi_i$ is equivalent in T to

$$(\bigwedge_{i\in I-\{k\}}\varphi_i)\wedge(\bigwedge_{j\in J_k}\phi_j)$$

which is clearly a conjunction of working formulas.

2. We replace a strict sub-working formula of φ_k by a conjunction of working formulas. Thus, let ϕ be a sub-working formula of φ_k of depth $d_\phi < d_k$ (thus ϕ is different from φ_k). Thus, φ_k has a sub-working formula[17] of the form

$$\neg(\exists\bar{x}\alpha\wedge(\bigwedge_{l\in L}\psi_l)\wedge\phi),$$

where L is a finite (possibly empty) set and all the ψ_l are working formulas. Let $\bigwedge_{j\in J}\phi_j$ be a conjunction of working formulas which is equivalent to ϕ in T. Thus the preceding sub-working formula of φ_k is equivalent in T to

$$\neg(\exists\bar{x}\alpha\wedge(\bigwedge_{l\in L}\psi_l)\wedge(\bigwedge_{j\in J}\phi_j)),$$

which is clearly a sub-working formula and thus φ_k is equivalent to a working formula and thus $\bigwedge_{i\in I}\varphi_i$ is equivalent to a conjunction of working formulas.

From 1 and 2 we deduce that (i) every substitution of a sub-working formula of a conjunction of working formulas by a conjunction of working formulas produces a conjunction of working formulas.

Since each rule transforms a working formula into a conjunction of working formulas, then according to the sub-section *"proof: first part"* and (i) we deduce that every repeated application of the rewriting rules on an initial working formula terminates and produces a conjunction of working formulas. Thus, since an initial working formula starts by $\neg^4$ and all its other negations are of the form $\neg^0$ then all long the application of our rules and by going down along the nested negations of any working formula φ obtained after any finite application of our rules, we can build many series of negations which represent the paths that we should follow from the top negation of φ to reach one of the sub-working formulas of φ of depth equal to one. Each of these series is of the one of the following forms:

- a series of $\neg^4$ followed by a possibly series of $\neg^0$,
- a series of $\neg^4$ followed by one $\neg^1$, followed by a possibly series of $\neg^0$,
- a series of $\neg^4$ followed by one $\neg^2$, followed by a possibly series of $\neg^0$,
- a series of $\neg^4$ followed by one $\neg^3$, followed by a possibly series of $\neg^0$,
- a series of $\neg^4$ followed by one or two $\neg^5$,
- one or two $\neg^5$.

While all the negations of these series are not of the form $\neg^5$ or their length is greater than 2 then one of the rules (1),...,(16) can still be applied. As a consequence, when no rule can be applied, we obtain a conjunctions of formulas of depth less or equal

[17] By considering that the set of the sub-formulas of any formula φ contains also the whole formula φ.

to 2 in which all the negations are of the form $\neg^5$. It is a conjunction of final working formulas. Since all the rules do not introduce new free variables then Property 4.6.3 holds. $\square$

4.7 The Solving Algorithm

Let p be a formula. Solving p in T proceeds as follows:

(1) Transform the formula $\neg p$ (the negation of p) into a wnfv normalized formula p_1 equivalent to $\neg p$ in T.

(2) Transform p_1 into the following initial working formula p_2

$$p_2 = \neg^4(\exists\varepsilon\, true \wedge \neg^0(\exists\varepsilon\, true \wedge p_1)),$$

where all the occurrences of $\neg$ in p_1 are replaced by $\neg^0$.

(3) Apply the preceding rewriting rules on p_2 as many time as possible. According to Property 4.6.3 we obtain at the end a wnfv conjunction p_3 of final working formulas of the form

$$\bigwedge_{i=1}^{n} \neg^5(\exists\bar{x}_i\, \alpha_i \wedge \bigwedge_{j=1}^{n_i} \neg^5(\exists\bar{y}_{ij}\, \beta_{ij})).$$

According to Property 4.5.4, the formula p_3 is equivalent in T to the following wnfv conjunction p_4 of general solved formulas

$$\bigwedge_{i=1}^{n} \neg(\exists\bar{x}_i\, \alpha_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists\bar{y}_{ij}\, \beta^*_{ij})),$$

where β^*_{ij} is the formula β_{ij} from which we have removed all the equations which occur also in α_i. Since p_4 is equivalent to $\neg p$ in T, then p is equivalent in T to

$$\neg \bigwedge_{i=1}^{n} \neg(\exists\bar{x}_i\, \alpha_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists\bar{y}_{ij}\, \beta^*_{ij})),$$

which is equivalent to the following disjunction p_5

$$\bigvee_{i=1}^{n} (\exists\bar{x}_i\, \alpha_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists\bar{y}_{ij}\, \beta^*_{ij})).$$

This is the final answer of our solver to the initial constraint p. Note that the negations which were at the beginning of each general solved formula of p_4 have been removed and the top conjunction of p_4 has been replaced by a disjunction. As a consequence, the set of the solutions of the free variables of p_5 is nothing other than the union of the solutions of each formula of the form $\exists\bar{x}_i\, \alpha_i \wedge \bigwedge_{j=1}^{n_i} \neg(\exists\bar{y}_{ij}\, \beta^*_{ij})$. According to Definition 4.4.6, each of these formulas is written in an explicit solved form which enables us to easily extract the solutions of its free variables. On the other hand, two cases arise:

- If p_4 does not contain free variables then according to Property 4.4.3 the formula p_4 is of the form $\bigwedge_{i=1}^{n} \neg(\exists\varepsilon\, true)$ and thus p_5 is of the form $\bigvee_{i=1}^{n} \exists\varepsilon\, true$. Two cases arise: if $n = 0$ then p_5 is the empty disjunction (i.e. the formula *false*). Else, if $n \neq 0$ then since we do not distinguish between $\varphi \wedge \varphi$ and φ, p_5 is the formula $\exists \varepsilon\, true$.

- If p_4 contains at least one free variable then according to Property 4.4.3 neither $T \models p_4$ nor $T \models \neg p_4$ and thus neither $T \models \neg p_5$ nor $T \models p_5$.

Since T has at least one model and since p_5 is equivalent to p in T and does not contain news free variables then we have the following theorem:

Theorem 4.7.1
Every formula is equivalent in T either to *true*, or to *false*, or to a wnfv formula which has at least one free variable, which is equivalent neither to *true* nor to *false*, and where the solutions of the free variables are expressed in a clear and explicit way.

The fact that T accepts at least one model is vital in this theorem. In fact, if T does not have models then the formula *true* can be equivalent to *false* in T. In other words, a formula can be equivalent to *true* in T using a finite application of our rules and equivalent to *false* using another different finite application of our rules. Theorem 3.3.1 prevents these kinds of conflicts and shows that T has at least three models $\mathcal{D}$, $\mathcal{T}r$ and $\mathcal{R}a$ and thus $T \models \neg(true \leftrightarrow false)$.

Corollary 4.7.2
T is a complete theory.

Note that using Theorem 4.7.1 and the properties 4.4.5 and 4.2.11, we get Maher's decision procedure (Maher 1988) for the basic theory of finite or infinite trees.

5 Implementation of our algorithm

We have implemented our algorithm in C++ and CHR (Constraint Handling Rules) (Fruehwirth 1998; Fruehwirth and Abdennadher 2003; Schrijvers and Fruehwirth 2006). The C++ implementation is a straightforward extension of those given in (Djelloul and Dao 2006b). It uses records and pointers and releases unused pointers after each rule application. The CHR implementation was done using Christian Holzbaur's CHR library of Sicstus Prolog 3.11.0. It consists of 18 CHR constraints and 73 CHR rules – most of them are needed for the complicated rules (15) and (16) of our algorithm. Even if our C++ implementation has given better performances, we think that it is interesting to show how can we translate our rules into CHR rules. We will be able to quickly prototype optimizations and variations of our algorithm and to parallelize it. For CHR, the implementation of this complex solver helps to understand what programming patterns and language features can be useful. The CHR code without comments and examples, but pretty-printed, is about 250 lines, which is one seventh of the size of our C++ implementation. Indeed for code size and degree of abstraction it seems only possible and interesting to describe the CHR implementation, and we do so in the following. The reader can find our full CHR implementation at `http://khalil.djelloul.free.fr/solver.txt` and can experiment with it online using webchr at `http://chr.informatik.uni-ulm.de/~webchr/`.

5.1 Constraint Handling Rules (CHR) Implementation

CHR manipulates conjunctions of constraints that reside in a constraint store. Let H, C and B denote conjunctions of constraints. A simplification rule $H \Leftrightarrow C \mid B$ replaces instances of the CHR constraints H by B provided the guard test C holds. A propagation rule $H \Rightarrow C \mid B$ instead just adds B to H without removing anything. The hybrid simpagation rules will come handy in the implementation: $H_1 \backslash H_2 \Leftrightarrow C \mid B$ removes matched constraints H_2 but keeps constraints H_1.

The constraints of the store comprise the state of an execution. Starting from an arbitrary initial store (called query), CHR rules are applied exhaustively until a fixpoint is reached. Trivial non-termination of a propagation rule application is avoided by applying it at most once to the same constraints.

Almost all CHR implementations execute queries from left to right and apply rules top-down in the textual order of the program (Duck et al. 2004). A CHR constraint in a query can be understood as a procedure that goes efficiently through the rules of the program. When it matches a head constraint of a rule, it will look for the other constraints of the head in the constraint store and check the guard. On success, it will apply the rule. The rule application cannot be undone. If the initial constraint has not been removed after trying all rules, it will be put into the constraint store. Constraints from the store will be reconsidered if newly added constraints constrain its variables.

5.1.1 CHR Constraints

The implementation consists of 18 constraints: two main constraints that encode the tree data structure of the working formulas (nf/4) and the atomic formulas (of/2), 9 auxiliary constraints that perform reachability analysis, variable renaming and copying of formulas, and 7 constraints that encode execution control information, mainly for rules (15) and (16).

In more detail, `nf(ParentId,Id,K,ExVars)` describes a **n**egated quantified basic **f**ormula with the identifier of its parent node, its own identifier Id, the level K from $\neg^k$ and the list of existentially quantified variables. `Var=FlatTerm of Id` denotes an equation between a variable and a flat term (a variable or a function symbol applied to variables) that belongs to the negated sub-formula with the identifier Id. `finite(U) of Id` denotes the relation $\mathit{finite}(U)$.

It is easy to represent any working formula φ using conjunctions of nf/4 and of/2 constraints. It is enough to create one nf/4 constraint for each quantified basic formula of φ and to use a conjunction of of/2 constraints to enumerate the atomic formulas linked to each quantified basic formula.

Example 5.1.2
Let φ be the following working formula

$$\neg^4 \left[\begin{array}{l} \exists u\, u = 1 \wedge \\ \left[\begin{array}{l} \neg^0(\exists \varepsilon\, u = s(v)) \wedge \\ \neg^0(\exists w_1\, u = s(w_1) \wedge w_1 = s(v)) \wedge \\ \neg^5(\exists \varepsilon\, v = s(u) \wedge u = 1 \wedge \left[\begin{array}{l} \neg^5(\exists \varepsilon\, v = s(u) \wedge u = 1 \wedge \mathit{finite}(w_1)) \wedge \\ \neg^5(\exists w_3\, v = s(u) \wedge u = 1 \wedge w_2 = s(w_3) \wedge \mathit{finite}(w_3)) \end{array} \right]) \end{array} \right] \end{array} \right].$$

φ can be expressed using the following conjunction of constraints:

$$\begin{array}{l}
\mathtt{nf}(\mathtt{Q},\mathtt{P1},4,[\mathtt{U}]),\mathtt{U}=1\,\mathrm{of}\,\mathtt{P1},\\
\mathtt{nf}(\mathtt{P1},\mathtt{P2},0,[\,]),\mathtt{U}=\mathtt{S}(\mathtt{V})\,\mathrm{of}\,\mathtt{P2},\\
\mathtt{nf}(\mathtt{P1},\mathtt{P3},0,[\mathtt{W1}]),\mathtt{U}=\mathtt{S}(\mathtt{W1})\,\mathrm{of}\,\mathtt{P3},\mathtt{W1}=\mathtt{S}(\mathtt{V})\,\mathrm{of}\,\mathtt{P3},\\
\mathtt{nf}(\mathtt{P1},\mathtt{P4},5,[\,]),\mathtt{V}=\mathtt{S}(\mathtt{U})\,\mathrm{of}\,\mathtt{P4},\mathtt{U}=1\,\mathrm{of}\,\mathtt{P4}\\
\mathtt{nf}(\mathtt{P4},\mathtt{P5},5,[\,]),\mathtt{V}=\mathtt{S}(\mathtt{U})\,\mathrm{of}\,\mathtt{P5},\mathtt{U}=1\,\mathrm{of}\,\mathtt{P5},\mathtt{finite}(\mathtt{W1})\,\mathrm{of}\,\mathtt{P5}\\
\mathtt{nf}(\mathtt{P4},\mathtt{P6},5,[\mathtt{W3}]),\mathtt{V}=\mathtt{S}(\mathtt{U})\,\mathrm{of}\,\mathtt{P6},\mathtt{U}=1\,\mathrm{of}\,\mathtt{P6},\mathtt{W2}=\mathtt{S}(\mathtt{W3})\,\mathrm{of}\,\mathtt{P6},\mathtt{finite}(\mathtt{W3})\,\mathrm{of}\,\mathtt{P6}
\end{array}$$

5.1.3 CHR Rules

The rules (1) to (14) have a rather direct translation into CHR rules. It seems hard to come up with a more concise implementation.

```
% 1 Locally simplify equations
(1) @ nf(Q,P,1,Xs) \ U=U of P <=> true.
(2) @ nf(Q,P,1,Xs) \ V=U of P <=> gt(U,V) | U=V of P.
(3) @ nf(Q,P,1,Xs), U=V of P \ U=G of P <=> gt(U,V) | V=G of P.
(4) @ nf(Q,P,1,Xs), U=F of P,  U=G of P <=> notsamefunctor(F,G) | true(P).
(5) @ nf(Q,P,1,Xs), U=F of P \ U=G of P <=>    samefunctor(F,G) |
                                                          same_args(F,G,P).
(6) @ nf(Q,P,1,Xs) <=> nf(Q,P,2,Xs).

% 2 finiteness check
(7) @ nf(P0,P,2,Xs), finite(U) of P \ finite(U) of P <=> true.
(8) @ nf(P0,P,2,Xs), U=V of P \ finite(U) of P <=> var(V) | finite(V) of P.
(9+10)@nf(P0,P,2,Xs),U=T of P \ finite(U) of P <=> nonvar(T) |
                                     reach_args(U,T,P), finite_args(U,T,P).
(11) @ nf(Q,P,2,Xs) <=> nf(Q,P,3,Xs).

% 4/0-4/1 copy down before solving
(12) @ nf(Q,P,4,Xs), A of P, nf(P,P1,0,Ys) ==> A of P1.
       nf(Q,P,4,Xs)        \ nf(P,P1,0,Ys) <=> nf(P,P1,1,Ys).

% 4/3-4/4 replace down after solving
(13) @ nf(Q,P,4,Xs),U=V of P, nf(P,P1,3,Ys)\ U=G of P1 <=> V\==G | U=V of P1.
       nf(Q,P,4,Xs)        \ nf(P,P1,3,Ys)             <=> nf(P,P1,4,Ys).

% 4/5-true trivial satisfaction - each A of P1 also occurs as A of P
(14) @ nf(Q,P,4,Xs), nf(P,P1,5,Ys) <=>
       \+(findconstraint(P1,(A of P1),_), \+findconstraint(P,(A of P),_)) |
                                                           true(P).
```

Note that rules (1) to (5) are similar to the classical CHR equation solver for flat rational trees (Fruehwirth and Abdennadher 2003; Meister and Fruehwirth 2006). By applying results of (Meister and Fruehwirth 2006), we can show that the worst-case time complexity of these rules of the algorithm is quadratic in the size of the equations.

In the rules (2) and (3), the predicate `gt(U,V)` checks if $\mathtt{U} \succ \mathtt{V}$. Note that the constraint `true(P)` used in rule (4) removes all constraints associated with `P` using an auxiliary rule not shown.

In rule (9+10) `reach_args(U,T,P)` checks reachability of `U` from itself in `P`. If so, `true(P)` will be executed and thus `P` will be removed, implementing rule (9). Otherwise, the subsequent `finite_args(U,T,P)` will propagate down the `finite` relation from `U` to its arguments, implementing rule (10).

In the rules (12) and (13) we handle equations one by one (due to the chosen granularity of the constraints), and thus we need auxiliary second CHR rules that perform the update of the level `K` afterwards.

For rule (14) the implementation is easy when nested negation-as-absence (Van Weert et al. 2006) is used to verify that there is no constraint in the sub-formula that is not in the main formula. Negation-as-absence can be directly encoded in CHR, but then it requires two additional rules per negation. Instead, we have chosen to use in the guard of the rule the CHR library built-in `findconstraint(Var,Pattern,Match)` that returns on backtracking all constraints `Match` that match `Pattern` and that are indexed on variable `Var` together with negation-as-failure provided by the Prolog built-in \+.

The translation of the complex rules (15) and (16) of the algorithm require 40 CHR rules, because several non-trivial new expressions have to be computed. Simpagation rules and auxiliary constraints collect the nested nf/4 constraints, compute the reachable variables and atomic formulas, rename the quantified variables and produce updated nf/4 and of/2 constraints. In order not to overburden the reader with technical details, we omit the description of those 40 rules.

5.2 Benchmarks: Two partner game

Let us consider the following two partner game: An ordered pair (i, j) is given, with i a non-negative (possibly null) integer and $j \in \{0, 1\}$. One after another, each player changes the values of i and j according to the following rules

- If $j = 0$ then the actual player should replace i by $i - 1$ in the pair (i, j).
- If $j = 1$ and i is odd then the actual player can either replace i by $i + 1$ or replace j by $j - 1$, in the pair (i, j).
- If $j = 1$ and i is even then the actual player can either replace i by $i + 1$ and j by $j - 1$ in the pair (i, j) or replace only i by $i + 1$ in the pair let (i, j)

The first player who cannot keep i non negative has lost. This game can be represented by the following directed infinite graph:

$$\begin{tikzcd}
(0,1) \arrow[r] \arrow[dr] & (1,1) \arrow[r] \arrow[d] & (2,1) \arrow[r] \arrow[dr] & (3,1) \arrow[r] \arrow[d] & (4,1) \arrow[r] \arrow[dr] & (5,1) \arrow[r] \arrow[d] & \dots \\
(0,0) & (1,0) \arrow[l] & (2,0) \arrow[l] & (3,0) \arrow[l] & (4,0) \arrow[l] & (5,0) \arrow[l] & \dots \arrow[l]
\end{tikzcd}$$

It is clear that the player which is at the position $(0, 0)$ and should play 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. It is easy to show that

$$winning_k(x) = \left[\begin{array}{l} \exists y\, move(x,y) \wedge \neg(\\ \exists x\, move(y,x) \wedge \neg(\\ ... \\ \exists y\, move(x,y) \wedge \neg(\\ \exists x\, move(y,x) \wedge \neg(\\ false \qquad \underbrace{)...)}_{2k} \end{array} \right]$$

where move(x, y) means: "starting from the position x we play one time and reach the position y". By moving down the negations, we get an embedding of 2k alternated quantifiers.

Suppose that F contains the function symbols 0, 1, f, g, c of respective arities 0, 0, 1, 1, 2. We code the vertices (i, j) of the game graph by the trees $c(\bar{i}, 0)$ and $c(\bar{i}, 1)$ with $\bar{i} = (fg)^{i/2}(0)$ if i is even, and $\bar{i} = g(\overline{i-1})$ if i is odd.[18] The relation $move(x, y)$ is then defined as follows:

$$move(x,y) \overset{\text{def}}{\leftrightarrow} transition(x,y) \vee (\neg(\exists uv\, x = c(u,v)) \wedge x = y)$$

with

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

$$pred(u_1,u_2) \overset{\text{def}}{\leftrightarrow} \left[\begin{array}{l} (\exists j\, u_1 = f(j) \wedge \left[\begin{array}{l} (\exists k\, j = g(k) \wedge u_2 = j) \vee \\ (\neg(\exists k\, j = g(k)) \wedge u_2 = u_1) \end{array} \right]) \vee \\ (\exists j\, u_1 = g(j) \wedge \left[\begin{array}{l} (\exists k\, j = g(k) \wedge u_2 = u_1) \vee \\ (\neg(\exists k\, j = g(k)) \wedge u_2 = j) \end{array} \right]) \vee \\ (\neg(\exists j\, u_1 = f(j)) \wedge \neg(\exists j\, u_1 = g(j)) \wedge \neg(u_1 = 0) \wedge u_2 = u_1) \end{array} \right]$$

If we take as input of our solver the formula $winning_k(x)$ then we will get as output a disjunction of simple formulas where the solutions of the free variable x represent all the k-winning positions.

For $winning_1(x)$ our algorithm gives the following formula:

$$\exists u_1 u_2\, x = c(u_1,u_2) \wedge u_1 = g(u_2) \wedge u_2 = 0,$$

[18] $(fg)^0(x) = x$ and $(fg)^{i+1}(x) = f(g((fg)^i(x)))$.

which corresponds to the solution $x = c(g(0), 0)$. For $winning_2(x)$ our algorithm gives the following disjunction of simple formulas

$$\left[\begin{array}{l} (\exists u_1 u_2\, x = c(u_1, u_2) \wedge u_1 = g(u_2) \wedge u_2 = 0) \\ \vee \\ (\exists u_3 u_4 u_5 u_6\, x = c(u_3, u_6) \wedge u_3 = g(u_4) \wedge u_4 = f(u_5) \wedge u_5 = g(u_6) \wedge u_6 = 0) \end{array}\right],$$

which corresponds to the solution $x = c(g(0), 0) \vee x = c(g(f(g(0))), 0)$. Note that x is the only free variable in the two preceding disjunctions and its solutions represent the positions which are k-winning.

The times of execution (CPU time in milliseconds) of the formulas $winning_k(x)$ are given in the following table as well as a comparison with those obtained using a decision procedure for decomposable theories (Djelloul 2006a) (even though the later does not produce comprehensible results, i.e. explicit solved forms). The benchmarks are performed on a 2.5Ghz Pentium IV processor, with 1024Mb of RAM.

k ($winning_k(x)$)	1	2	4	5	7	10	20	40
CHR (our 16 rules)	320	690	1750	2745	5390	–	–	–
C++ (Djelloul 2006a)	28	50	115	150	245	430	2115	–
C++ (our 16 rules)	25	40	90	115	175	315	1490	15910

This decision procedure takes from 10% to 40% more time, comparing with our C++ implementation to solve the $winning_k(x)$ formulas of our game and overflows the memory for $k > 20$, i.e. 40 nested alternated quantifiers. Our C++ implementation has better performance and is able to give all the $winning_k$ strategies in a clear and explicit way until $k = 40$, i.e. 80 nested alternated quantifiers.

The execution times of $winning_k(x)$ using our CHR implementation are 12-30 times slower than those obtained using our C++ implementation and the maximal depth of working formula that can be solved is 14 ($k = 7$). These results are in line with the experience that the overhead of using declarative CHR without optimisations induces an overhead of about an order of magnitude over implementations in procedural languages. As discussed in the conclusions, switching to a more recent optimizing CHR compiler may close the gap to a small constant factor.

The algorithm given in (Djelloul 2006a) is a decision procedure in the form of five rewriting rules which for every decomposable theory T transforms a first-order formula φ into a conjunction ϕ of final formulas easily transformable into a Boolean combination of existentially quantified conjunctions of atomic formulas. This decision procedure does not warrant that the solutions of the free variables are expressed in a clear and explicit way and can even produce formulas having free variables but being always true or false in T. In fact, for our two player game, we got conjunctions of final formulas where the solutions of the free variable x was incomprehensible, especially from $k = 5$.

We also tried to use Remark 4.4.2 of (Djelloul 2006a) which gives a way to get a disjunction of the form

$$\bigvee_{i\in I}(\exists \bar{x}'_i\, \alpha'_i \wedge \bigwedge_{j\in J_i} \neg(\exists \bar{y}'_{ij}\, \beta'_{ij})) \tag{36}$$

as output of the decision procedure. As the author of (Djelloul 2006a) wrote: *"it is more easy to understand the solutions of the free variables of this disjunction of solved formulas than those of a conjunction of solved formulas"*. That is of course true, but this does not mean that the solutions of the free variables of this formula are expressed in a clear and explicit way. In fact, we got a disjunction of the form (36) where many variables which occurred in left hand sides of equations of α'_i occurred also in left hand sides of equations of some β'_{ij}. Moreover, many formulas of the preceding disjunction contained occurrences of the free variable x but after a hard and complex manual checking we found them equivalent to *false*. As a consequence, the solutions of x was completely not evident to understand and we could not extract clear and understandable $winning_k(x)$ strategies for all $k \geq 5$. In order to simplify the formula (36) we finally used our solving algorithm on it and have got a disjunction of simple formulas equivalent to (36) in T in which: (1) all the formulas having free occurrences of x but being always false in T have been removed, (2) the solutions of the free variable x were expressed in a clear and explicit way.

We now discuss why our solver is faster than the decision procedure of K. Djelloul. The latter uses many times a particular distribution (rule (5) in (Djelloul 2006a)) which decreases the depth of the working formulas but increases exponentially the number of conjunctions of the working formulas until overflowing the memory. Our solving algorithm uses a similar distribution (rule (16)) but only after a necessary propagation step which copies the basic formulas into the sub-working formulas and checks if there exists no working formulas which contradict their top-working formula. This step enables us to remove the inconsistent working formulas and to not lose time with solving a huge working formulas (i.e. of big depth) which contradicts their top-working formulas. It also prevents us from making exponential distributions between huge inconsistent working formulas which finally are all equivalent to *false*. Unfortunately, we cannot add this propagation step to the decision procedure of (Djelloul 2006a) since it uses many properties which hold only for the theory of finite or infinite trees and not for any decomposable theory T.

The game introduced in this paper was inspired from those given in (Djelloul 2006a) but is different. Solving a $winning_k(x)$ formula in this game generates many huge working formulas which contradict their top-working formulas. Our algorithm removes directly these huge working formulas after the first propagation step (rules (1),...,(13)). The decision procedure cannot detect this inconsistency and is obliged to apply a costly rule (rule (5) in (Djelloul 2006a)) to decrease the size of these inconsistent working formulas until finding basic inconsistent formulas of the form $\neg(a \wedge \neg(\exists\varepsilon\, true))$ or $\neg(\exists\varepsilon\, false \wedge \varphi)$. At each application of this rule, the depth of the working formulas decreases but the number of conjunctions increase exponentially until overflowing the memory. This explains why for this game the decision procedure overflows the memory for $k > 20$ while our solver can compute the $winning_k(x)$ strategies until $k = 40$.

5.3 Benchmarks: Random normalized formulas

We have also tested our 16 rules on randomly generated normalized formulas such that in each sub-normalized formula of the form $\neg(\exists \bar{x}\, \alpha \wedge \bigwedge_{i=1}^{n} \varphi_i)$, with the φ_i's normalized formulas and $n \geq 0$, we have:

- n is a positive integer randomly chosen between 0 and 4.
- The number of the atomic formulas in the basic formula α is randomly chosen between 1 and 8. Moreover, the atomic formula *true* occurs at most once in α.
- The vector of variables and the atomic formulas of $\exists \bar{x}\, \alpha$ are randomly generated starting from a set containing 10 variables, the relation *finite* and 6 function symbols: $f_0, f_1, f_2, g_0, g_1, g_2$. Each function symbol f_j or g_j is of arity j with $0 \geq j \geq 2$.

The benchmarks were realized on a 2.5Ghz Pentium IV processor with 1024Mb of RAM as follows: For each integer $1 \geq d \geq 42$ we generated 10 random normalized formulas[19] of depth d, we solved them and computed the average execution time (CPU time in milliseconds). Once again, the performances (time and space) of our 16 rules are impressive comparing with those of the decision procedure for decomposable theories.

d	4	8	12	22	26	41
CHR (our 16 rules)	1526	4212	16104	–	–	–
C++ (Djelloul 2006a)	108	375	1486	18973	–	–
C++ (our 16 rules)	88	202	504	3552	11664	2142824

Note that for $d = 42$, all the normalized formulas could not be solved and overflowed the memory.

6 Discussion and conclusion

We gave in this paper a first-order axiomatization of an extended theory T of finite or infinite trees, built on a signature containing not only an infinite set of function symbols but also a relation *finite*(t) which enables to distinguish between finite or infinite trees. We showed that T has at least one model and proved its completeness by giving not only a decision procedure but a full first-order constraint solver which

[19] We of course renamed the quantified variables of each randomly generated normalized formula so that it respects the discipline of the formulas in T

transforms any first-order constraint φ into an equivalent disjunction ϕ of simple formulas such that ϕ is either the formula *true*, or the formula *false*, or a formula having at least one free variable, being equivalent neither to *true* nor to *false* and where the solutions of the free variables are expressed in a clear and explicit way. This algorithm detects easily formulas that have free variables but are always true or always false in T and is able to solve any first-order constraint satisfaction problem in T. Its correctness implies the completeness of T.

On the other hand S. Vorobyov (Vorobyov 1996) has shown that the problem of deciding if a proposition is true or not in the theory of finite or infinite trees is non-elementary, i.e. the complexity of all algorithms solving propositions is not bounded by a tower of powers of $2's$ (top down evaluation) with a fixed height. A. Colmerauer and T. Dao (Colmerauer and Dao 2003) have also given a proof of non-elementary complexity of solving constraints in this theory. As a consequence, our algorithm does not escape this huge complexity and the function $\alpha(\varphi)$ used to show the termination of our rules illustrates this result.

We implemented our algorithm in C++ and CHR and compared both performances with those obtained using a recent decision procedure for decomposable theories (Djelloul 2006a). This decision procedure is not able to present the solutions of the free variables in a clear and explicit way and overflows the memory while solving normalized formulas with depth $d > 40$. Our C++ implementation is faster than this decision procedure and can solve normalized formulas of depth $d = 80$. This is mainly due to the fact that our algorithm uses two steps: (1) a top-down propagation of constraints and (2) a bottom-up elimination of quantifiers and depth reduction of the working formulas. In particular, the first step enables to minimize the number of application of costly distributions and avoids to lose time with solving huge formulas which contradict their top-formulas.

Future implementation work will focus on our CHR implementation, since from previous experience we are confident that we can get the performance overhead down to a small constant factor while gaining the possibility to prototype variations of our algorithm in a very high level language. Switching to a more recent optimizing CHR compiler from K.U. Leuven would most likely improve performance. We also think that we can minimize the use of the debated negation-as-absence (Van Weert et al. 2006) by introducing reference counters for the two main constraints. This should also give us the possibility to obtain a parallel implementation that is derived from the existing one with little modification, similar to what has been done for parallelizing the union-find algorithm in CHR (Fruehwirth 2005).

Acknowledgments We thank Alain Colmerauer for our very long discussions about the theory of finite or infinite trees and its models. Many thanks also to the anonymous referees for their careful reading and suggestions which help us to improve this paper. Khalil Djelloul thanks the DFG research project GLOB-CON for funding and supporting his research. Thanks also to Marc Meister and Hariolf Betz for their kind review of this article.

References

Abdennadher, S. 1997. Operational Semantics and Confluence of Constraint Propagation Rules. In Proc of the third International Conference on Principles and Practice of Constraint Programming. LNCS 1330.

Baader, F. and Nipkow, T. 1998. Term rewriting and all that. Cambridge university press. ISBN 0-521-45520-0.

Benhamou, F., Colmerauer, A., Garetta, H., Pasero, R. and Van-caneghem, M. 1996. Le manuel de Prolog IV. PrologIA, Marseille, France.

Burckert, H. 1988. Solving disequations in equational theories. In Proceeding of the 9th Conference on Automated Deduction, LNCS 310, pp. 517–526, Springer-Verlag.

Clark, K.L. 1978. Negation as failure. In Logic and Data bases. Ed Gallaire, H. and Minker, J. Plenum Pub.

Colmerauer, A. 1982. Prolog and infinite trees. In K.L. Clark and S-A. Tarnlund, editors, Logic Programming. Academic Press. pp. 231–251.

Colmerauer, A., Kanoui, H. and Van-caneghem,M. 1983. Prolog, Theoretical Basis and Current Developments. TSI (Technology and Science of Informatics), 2(4):271–311.

Colmerauer, A. 1984. Equations and inequations on finite and infinite trees. Proceeding of the International conference on the fifth generation of computer systems, pp. 85–99.

Colmerauer, A. 1990. An introduction to Prolog III. Communication of the ACM, 33(7):68–90.

Colmerauer, A. and Dao, T. 2003. Expressiveness of full first-order formulas in the algebra of finite or infinite trees, Constraints, 8(3): 283–302.

Comon, H. 1988. Unification et disunification : Theorie et applications. PhD thesis, Institut National Polytechnique de Grenoble.

Comon, H. and Lescanne, P. 1989. Equational problems and disunification. Journal of Symbolic Computation, 7: 371–425.

Comon, H. 1991a. Disunification: a survey. In J.L. Lassez and G. Plotkin, editors, Computational Logic: Essays in Honor of Alan Robinson. MIT Press.

Comon, H. 1991b. Resolution de contraintes dans des algebres de termes. Rapport d'Habilitation, Universite de Paris Sud.

Courcelle, B. 1983. Fundamental Properties of Infinite Trees, Theoretical Computer Science, 25(2):95–169.

Courcelle, B. 1986. Equivalences and Transformations of Regular Systems applications to Program Schemes and Grammars, Theoretical Computer Science, 42: 100–122.

Dao, T. 2000. Resolution de contraintes du premier ordre dans la theorie des arbres finis ou infinis. These d'informatique, Universite de la mediterranee, France.

Djelloul, K. 2006a. Decomposable Theories. Journal of Theory and practice of Logic Programming. (to appear)

Djelloul, K. and Dao, T. 2006b. Solving First-Order formulas in the Theory of Finite or Infinite Trees : Introduction to the Decomposable Theories. Proceeding of the 21st ACM Symposium on Applied Computing (SAC'06). ACM press, pp. 7–14.

Duck, G., Stuckey, P., Banda, M. and Holzbaur, C. 2004. The Refined Operational Semantics of Constraint Handling Rules. In Proc of the 20th International Conference on Logic Programming. LNCS 3132, pp. 105-119.

Fruehwirth, T. 1998. Theory and Practice of Constraint Handling Rules. Special Issue on Constraint Logic Programming. Journal of Logic Programming. 37(1–3): 95-138.

Fruehwirth, T. and Abdennadher, S. 2003. Essentials of Constraint Programming. Springer.

Fruehwirth, T. 2005. Parallelizing Union-Find in Constraint Handling Rules Using Confluence. In proc of the 21st International Conference of Logic Programming. LNCS, Vol 3668. pp: 113-127.

Herbrand, J. 1930. Recherches sur la theorie de la demonstration. PhD thesis, Universite de Paris, France.

Huet, G. 1976. Resolution d'equations dans les langages d'ordre 1, 2,... ω. These d'Etat, Universite Paris 7. France.

Jaffar, J. 1984. Efficient unification over infinite terms. New Generation Computing, 2(3): 207–219.

John, E. and Ullman, D. 1979. Introduction to automata theory, languages and computation. Addison-Wesley publishing company.

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.

Kunen, K. 1987. Negation in logic programming. Journal of Logic Programming, 4: 289–308.

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.

Lassez, J. and Marriott, K. 1987. Explicit representation of terms defined by counter examples. Journal of automated reasonning. 3:301-317.

Lassez, J. and McAloon, K. 1989. Independence of negative constraints. In proceedings of TOPSOFT, LNCS 351, pp. 19-27.

Lyndon, R.C. 1964. Notes on logic. Van Nostrand Mathematical studies.

Maher, M. 1988. Complete axiomatization of the algebra of finite, rational and infinite trees. Technical report, IBM - T.J.Watson Research Center.

Maher, M. and Stuckey, P. 1995. On inductive inference of cyclic structures. Annals of mathematics and artificial intelligence, 15(2):167-208.

Malcev, A. 1971. Axiomatizable classes of locally free algebras of various types. In B.Wells III, editor, The Metamathematics of Algebraic Systems. Anatolii Ivanovic Malcev. Collected Papers: 1936-1967, volume 66, chapter 23, pp. 262–281.

Martelli, A. and Montanari, U. 1982. An efficient unification algorithm. ACM Trans. on Languages and Systems, 4(2): 258–282.

Meister, M. and Fruehwirth, T. 2006. Complexity of the CHR Rational Tree Equation Solver. In Proc of the third Workshop on Constraint Handling Rules.

Paterson, M. and Wegman, N. 1978. Linear unification. Journal of Computer and Systems Science, 16:158–167.

Podelski, A. and Van Roy, P. 1994. The beauty and beast algorithm : quasi-linear incremental tests of entailment and disentailment over trees. In proc of the 1994 International Symposium on Logic Programming. MIT press, pp. 359-374.

Ramachandran, V. and Van Hentenryck, P. 1993. Incremental algorithms for formula solving and entailment over rational trees. Proceeding of the 13th Conference Foundations of Software Technology and Theoretical Computer Science, LNCS volume 761, pp. 205–217.

Robinson, J.A. 1965. A machine-oriented logic based on the resolution principle. JACM, 12(1):23–41.

Rybina, T. and Voronkov, A. 2001. A decision procedure for term algebras with queues. ACM transaction on computational logic. 2(2): 155-181.

Schrijvers, T., Demoen, B., Duck, G., Stuckey, P. and Fruehwirth, T. 2006. Automatic implication checking for CHR constraints. In Proc of the 6th International Workshop on Rule-Based Programming. ENTC, vol 147, pp. 93-111.

Schrijvers, T. and Fruehwirth. CHR Website, `www.cs.kuleuven.ac.be/~dtai/projects/CHR/`

Smith, A. 1991. Constraint operations for CLP. In Logic Programming: Proceedings of the 8th International Conference. Paris. pp. 760–774.

Van Weert, P., Sneyers, J., Schrijvers, T. and Demoen, B. 2006. Constraint Handling Rules with Negations as Absence. In Proc of the third Workshop on Constraint Handling Rules.

Vorobyov, S. 1996. An Improved Lower Bound for the Elementary Theories of Trees, Proceeding of the 13th International Conference on Automated Deduction (CADE'96). Springer Lecture Notes in Artificial Intelligence, vol 1104, pp. 275– 287.