Equational Formulae with Membership Constraints*

Hubert Comon† and Catherine Delor

Laboratoire de Recherche en Informatique et CNRS
Bat. 490, Université de Paris Sud
91405 ORSAY cedex, France.
E-mail {comon,delor}@lri.lri.fr.

Abstract

We propose a set of transformation rules for first order formulae whose atoms are either equations between terms or "membership constraints" $t \in \zeta$. ζ can be interpreted as a regular tree language (ζ is called a *sort* in the algebraic specification community) or as a tree language in any class of languages which satisfies some adequate closure and decidability properties. This set of rules is proved to be correct, terminating and complete. This provides a quantifier elimination procedure: for every regular tree language $\mathcal{L}$, the first order theory of some structure defining $\mathcal{L}$ is decidable. This extends the results of Mal'cev (1971), Maher (1988), Comon and Lescanne (1989). We also show how to apply our results to automatic inductive proofs in equational theories.

Introduction

To *unify* two terms s and t means to turn the equation $s = t$ into an equivalent *solved form* which is either $\perp$ (this means that $s = t$ has no solution, or, in other words, that s and t are not unifiable) or else a formula $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$ which denotes a most general unifier of s and t. These concepts have been generalized to arbitrary first order formulae whose atoms are equations (they are called *equational formulae*). More precisely, every equational formula can be reduced, using a fixed set of rules, to a solved form which is either $\perp$ or has at least one solution in the Herbrand Universe $T(F)$ (Comon 1988, Comon and Lescanne 1989, Maher 1988). This property (solvability) is the main requirement for solved forms in this framework. On the other hand, it is not possible to expect a finite set of most general unifiers as in the unification case since such a finite set does not exist in general.

From the logical point of view, reduction of equational formulae to solved forms shows that the first order theory of finite trees over a finite alphabet is decidable. The reduction rules provide a complete (recursive) axiomatization of the algebra of finite trees. Mal'cev already gave long time ago all complete axiomatizations of classes of locally-free algebras, which includes a complete axiomatization of the algebra of finite trees (Mal'cev 1971). Actually, whether the alphabet has a finite or an infinite set of function symbols plays a crux role. In the latter case, Clark's theory of equality is a complete axiomatization (Kunen 1987). The former case is more difficult, because we have to consider the *domain closure axiom*. With this additional formula,

*This is a revised version of a draft paper whose abstract appeared in proc. ICALP 90 under the title "Equational formulas in Order-Sorted Algebras".

†This research was partly supported by the *Greco de Programmation du CNRS* and by the ESPRIT working group COMPASS.

the theory becomes complete again (Maher 1988). The transformations of equational formulae are interesting in many respects : applications in functional and logic programming, automated deduction and in other areas as well, are described by Comon (1991).

On the other hand, order-sorted signatures and order-sorted algebras have been introduced by J. Goguen in 1978 and the logic was subsequently studied in many papers (for example Goguen, Jouannaud, and Meseguer 1985, Kirchner, Kirchner, and Meseguer 1988, Aristide 1989, Goguen and Meseguer 1987a, Smolka, Nutt, Goguen, et al. 1989). Order-sorted equational logic is useful for specifying some kind of partial functions (Goguen and Meseguer 1987b) but also in automatic proofs by induction in equational theories (Comon 1989).

The aim of this paper is to bring together the order-sorted framework and the simplification of equational formulae. Because of the sort discipline, we introduce "membership constraints" in equational formulae. Therefore, what we call "equational formulae" in this paper are first order formulae whose atoms are either equations or expressions $t \in \zeta$. This point of view ("sorts as constraints") is now emerging as the right concept. It differs from the classical points of view as developed by Goguen and Meseguer (1987a), Kirchner, Kirchner, and Meseguer (1988), Smolka, Nutt, Goguen, et al. (1989). Indeed, instead of restricting the syntax to "well-typed" terms, we express the well-typedness of t by a predicate $t \in \zeta$ (as in (Smolka 1986, Smolka 1989)). This point of view is simpler and avoids inadequate properties such as "regularity", "co-regularity", "downward completeness"... Indeed, such properties become necessary when one wants to ensure special features of solved forms (such as to represent a most general unifier without adding new sort symbols) but are actually useless for deciding the solvability. On the other hand, solvability is the only (operational) requirement when the formulae are used as constraints in some constraint logic programming language. The counterpart of our choice is that some additional difficulties come up with the predicate symbol $\in$. Nevertheless, we propose in this paper a set of transformation rules which are correct, terminating and complete w.r.t. a set of solved forms which has the solvability property.

Let us show a very simple example of formulae and simplification of formulae.

Example 1
The signature consists in a constant 0 and two unary function symbols $succ$ and $succ'$. There are four sorts : $nat, even, odd, nat'$. The model is defined by the rules

$$0: \begin{array}{l} \to even \\ \to nat' \end{array} \qquad succ: \begin{array}{l} odd \to even \\ even \to odd \end{array} \qquad succ' : nat' \to nat'$$

and $even < nat; odd < nat$. Intuitively, we have two copies of natural numbers, sharing 0. The formula

$$\forall y : y \notin nat \vee (\exists x : x \in even \wedge succ(x) = y) \vee (\exists x : x \in odd \wedge succ(x) = y) \vee y = 0$$

will reduce to $\top$ with our system. The formula

$$\forall y : x \in nat \wedge (y \notin nat' \vee x \neq y) \wedge (y = succ(succ(x)) \supset y \in even)$$

represents all the terms that are even and that are not in nat'. A solved form is

$$x \neq 0 \wedge x \in even$$

Actually, an order-sorted signature is a finite bottom-up tree automaton. Surprisingly, this has never been noticed in the literature (except by the first author in (Comon 1989)). This point of view is useful since we have to perform some basic operations such as deciding finiteness, ¿co-finiteness, emptiness or computing the set of terms that are of sort ζ and not of sort ζ': deduction on sorts reduces to well-known computations on finite tree automata. This also leads to another consequence of our results : the first order theory of a regular tree language is decidable. For any regular tree language $\mathcal{L}$, it is actually possible to provide a complete axiomatization of some structure describing $\mathcal{L}$.

A set of transformation rules for equational formulae consists of an axiomatic part and a control. The control ensures termination whereas the axiomatic part ensures correctness. An original feature of our transformation rules lies in the weakness of the control. With a weak control, the termination proof becomes more difficult but this leaves more freedom for various implementations. In particular, experiments in the one-sorted case, carried on with the implementations of M. Mehl (1988) and C. Delor show that reducing the formula to some normal form (e.g. clausal form) at each transformation step can be very inefficient. In this paper we drop this condition which was required in previous papers.

We first give the syntax (section 1) and the semantics (section 2) of our equational formulae. Then, we give in section 3 the definition of solved forms and prove their solvability. In section 4 we give a set of transformation rules (or rewrite rules in the algebra of formulae) that are proved to be correct. The main results are termination and completeness of our set of rules. Section 5 is devoted to the termination of quantifier elimination and section 6 is devoted to completeness results. In section 7 we sum up the sequence of transformations and their properties. We also give a consequence on the axiomatization of a structure describing a regular tree language. We sketch in section 8 an extension to non-regular languages and we conclude in section 9 with an application to automatic proofs by induction in equational theories. Indeed, we show how deciding inductive reducibility may reduce to solving an equational formula.

1 Syntax

1.1 Terms

Given a finite ranked alphabet F of *function symbols* and an (infinite) set X of *variable symbols*, the sets $T(F)$ of ground terms and $T(F,X)$ of terms are defined as usually (they are confused with finite trees) If $t \in T(F,X)$, $\mathcal{V}ar(t)$ denotes the set of variables occurring in t. For all operations on terms, we use the notations of Dershowitz and Jouannaud (1990). For example, if p is a position of $t \in T(F,X)$, $t|_p$ and $t[u]_p$ are respectively the subterm of t at position p and the term obtained by replacing $t|_p$ with u in t. Λ is the empty position. An expression $t[u]$ indicates that u is a subterm of t. The *path* to a position p in a term t is the set of prefixes of p. It is also used for the sequence of symbols that labels the prefixes of p. $|t|$ is the *size* of the term t: it is the number of elements in the *set of positions* $Pos(t)$.

1.2 Order-Sorted Signatures and Finite Tree Automata

In the following, S denotes a finite (non empty) set of *sort symbols*, F a finite set of *function symbols*, each $f \in F$ being associated with a non-negative integer $a(f)$ called *arity* and X an infinite set of *variable symbols*. Each of these three sets is assumed to be disjoint from the two others. In addition, $\leq_S$ is an ordering on S and τ is a mapping that associates to each $f \in F$ a finite set of strings in $S^{a(f)+1}$. As usual, when $\underline{s}_1 \ldots \underline{s}_n \underline{s} \in \tau(f)$, we also write $f : \underline{s}_1 \times \ldots \times \underline{s}_n \rightarrow \underline{s}$.

Such a tuple $(S, F, \leq_S, \tau)$ is usually called an *order-sorted signature*. It can also be viewed as a finite bottom-up tree automaton (Q, Q_f, F, P) where S itself is the set of states Q, F is the terminal alphabet (labels of the trees). Q_f, the set of final states, is Q itself and the transition rules P are of the following two kinds :

- For each $\underline{s} \leq_S \underline{s}'$ P contains an ϵ-transition rule : $\underline{s}(t) \rightarrow \underline{s}'(t)$
- For each $f : \underline{s}_1 \times \ldots \times \underline{s}_n \rightarrow \underline{s}$, P contains the rule

$$f(\underline{s}_1(t_1), \ldots, \underline{s}_n(t_n)) \rightarrow \underline{s}(f(t_1, \ldots, t_n))$$

The transition rules are also abbreviated by omitting irrelevant terms, as for instance in $f(\underline{s}_1, \ldots, \underline{s}_n) \rightarrow \underline{s}$ for the latter set of rules.[1]

Assume that all symbols in Q are unary function symbols, then a *move* from $t \in T(F \cup Q)$ to $t' \in T(F \cup Q)$ of the automaton $\mathcal{A}$ is a rewrite step $t \xrightarrow{P} t'$ where P is the set of productions of $\mathcal{A}$ (we use here the notations of Dershowitz and Jouannaud (1990)).

It turns out that what is usually called *well formed terms* in the literature (Goguen and Meseguer 1987a, Smolka, Nutt, Goguen, et al. 1989) is the set of trees recognized by the above finite tree automaton. In the same way, if $\underline{s} \in S$, what is usually called *a well-formed term of sort* $\underline{s}$ is a tree that is recognized by the above automaton, replacing Q_f by $\{\underline{s}\}$.

In the following, we will make no difference between the two points of view, using either tree languages vocabulary or the order-sorted framework.

1.3 Sort Expressions

A *sort expression* is either $\top_S$, $\perp_S$, an element of S or one of the following expressions

- $\zeta \wedge \zeta'$
- $\zeta \vee \zeta'$
- $\neg\zeta$
- $f(\zeta_1, \ldots, \zeta_n)$ where $f \in F$ has arity n

where $\zeta, \zeta', \zeta_1, \ldots, \zeta_n$ are sort expressions. The set of sort expressions is denoted $\mathcal{E}(S)$. If we except the construction $f(\zeta_1, \ldots, \zeta_n)$, sort expressions are propositional formulae over the propositional variables S. The construction $f(\zeta_1, \ldots, \zeta_n)$ is similar to the domain construction in unified algebras (Mosses 1989). But although (ground) terms are sort expressions, we still have two levels; we do not allow sort variables, and sort expressions will always denote a regular language. (Our language is less expressive than the language of Mosses (1989), this is the price to pay for the decidability of our logic).

1.4 Equational Formulae

An *equation* is a set of two terms $s, t \in T(F, X)$ denoted $s = t$. A *membership constraint* is an expression $t \in \zeta$ where $t \in T(F, X)$ and $\zeta \in \mathcal{E}(S)$. An *equational formula* (over S, F, X) is a first order formula whose atoms are equations, membership constraints or the special symbol $\top$. We use the notation $\exists \vec{x}\ \phi$ as an abbreviation for $\exists x_1, \ldots, x_n\ \phi$. $\vec{x}$ is assumed to be a set. This means that we make no difference between for instance $\exists x_1, x_2, x_1, x_3$ and $\exists x_3, x_2, x_1$. Let us go further in the details of the syntax of equational formulae. (This seems to be irrelevant details, but this is compulsory for performing proofs in the algebra of equational formulae).

We introduce some additional connectives : $\perp$ stands for $\neg\top$ and $s \neq t$ for $\neg(s = t)$. Actually, the set $\mathcal{EF}$ of equational formulae is a quotient algebra since, for example, $\wedge$ and $\vee$

[1] These definitions correspond to definitions of Gécseg and Steinby (1984), except that we droped initial states which are given by the rules $a \rightarrow \underline{s}$ where a is a constant.

are associative and commutative. All the properties we want to incorporate in the syntax are summarized in figure 1. They consist of associativity and commutativity axioms and three sets of rewrite rules. The rules P_1 and P_2 are rather rule schemas; they denote the (infinite) set of the rules that fulfill the corresponding condition. These two rules assume that no variable of $\vec{y_2}$ occurs free in $P[\top]_u$, so as to avoid captures. This property is always assumed in the paper.

The set $\mathcal{S}_1 \cup \mathcal{S}_2$ of rules[2] is convergent modulo the identities (missing definitions can be found in Dershowitz and Jouannaud 1990). This means in particular that every formula ϕ has a unique (up to identities) normal form $\phi\downarrow_{1,2}$ w.r.t. $\mathcal{S}_1 \cup \mathcal{S}_2$. Such a normal form does not involve the symbol $\neg$, except in sort expressions (this is easy to verify). Similarly, the set of rewrite rules denoted by $\mathcal{S}_3$ is convergent modulo the identities and does not introduce any redex for $\mathcal{S}_1 \cup \mathcal{S}_2$. Therefore, any formula ϕ has a unique normal form (up to identities) $(\phi\downarrow_{1,2})\downarrow_3$ which is irreducible w.r.t. the rules of $\mathcal{S}_1, \mathcal{S}_2$.

The equality of equational formulae (denoted $\equiv$) is decidable using these rules. All formulae considered in the following are assumed to denote their normal form and we will make no difference between $\mathcal{EF}$ and its set of representatives. Moreover, we will omit irrelevant parentheses around $x+y$ in an expression $(x+y)+z$ involving an AC symbol $+$.

We did not incorporate the distributivity rules in the syntax (this makes a difference with, for example, with Comon and Lescanne 1989). Indeed, choosing a boolean normal form (disjunctive or conjunctive) at once has been proved to be very expensive in practice; we want to allow more general controls, using the most convenient boolean form for the problem at hand.

2 Semantics

2.1 Semantics of Sort Expressions

Sorts will be interpreted as subsets of $T(F)$. Let $[\![\cdot]\!]$ be such a mapping from S to $2^{T(F)}$. $[\![\cdot]\!]$ is extended to a Boolean lattice homomorphism from sort expressions into sets in a straightforward way:

- $[\![\top_S]\!] = T(F)$ and $[\![\bot_S]\!] = \emptyset$
- $[\![\zeta \wedge \zeta']\!] = [\![\zeta]\!] \cap [\![\zeta']\!]$
- $[\![\zeta \vee \zeta']\!] = [\![\zeta]\!] \cup [\![\zeta']\!]$
- $[\![\neg\zeta]\!]$ is the complement of $[\![\zeta]\!]$ into $T(F)$
- $[\![f(\zeta_1, \ldots, \zeta_n)]\!] = \{f(t_1, \ldots, t_n) \mid t_1 \in [\![\zeta_1]\!], \ldots, t_n \in [\![\zeta_n]\!]\}$

Since we aim at solving arbitrary first order formulae, we need of course some strong properties of this semantic mapping $[\![\cdot]\!]$. First, finiteness and emptiness of $[\![\zeta]\!]$ should be (effectively) decidable. But we need moreover a *decomposability property* which can be stated as follows:

Definition 1 *A semantic mapping $[\![\cdot]\!]$ from $\mathcal{E}(S)$ to $2^{T(F)}$ is* decomposable *if there is a (effectively computable) mapping which associates to each function symbol $f \in F$, of arity $n \geq 1$ and each sort expression ζ a finite set $f^{-1}(\zeta)$ of n-uples of sort expressions in such a way that*

$$f(t_1, \ldots, t_n) \in [\![\zeta]\!] \quad \textit{iff} \quad \exists\,(\zeta_1, \ldots \zeta_n) \in f^{-1}(\zeta),\ t_1 \in [\![\zeta_1]\!], \ldots, t_n \in [\![\zeta_n]\!]$$

[2] The distinction between $\mathcal{S}_1$ and $\mathcal{S}_2$ is not relevant yet. The need for this distinction will become apparent in section 5.

Identities

$$s = t \sim t = s$$

$$\begin{array}{rclrcl} (\phi_1 \wedge \phi_2) \wedge \phi_3 & \sim & \phi_1 \wedge (\phi_2 \wedge \phi_3) & \phi_1 \wedge \phi_2 & \sim & \phi_2 \wedge \phi_1 \\ (\phi_1 \vee \phi_2) \vee \phi_3 & \sim & \phi_1 \vee (\phi_2 \vee \phi_3) & \phi_1 \vee \phi_2 & \sim & \phi_2 \vee \phi_1 \end{array}$$

Rules (the set $\mathcal{S}_1$)

$$\begin{array}{rclrcl} s = s & \to & \top & s \neq s & \to & \bot \\ \neg \bot & \to & \top & \neg\top & \to & \bot \\ \neg(s \neq t) & \to & s = t & \neg(s = t) & \to & s \neq t \\ \top \wedge \phi & \to & \phi & \top \vee \phi & \to & \top \\ \bot \wedge \phi & \to & \bot & \bot \vee \phi & \to & \phi \\ \phi \wedge \phi & \to & \phi & \phi \vee \phi & \to & \phi \\ \exists \emptyset\ \phi & \to & \phi & \exists \vec{x} (\exists \vec{y}\ \phi) & \to & \exists (\vec{x} \cup \vec{y})\ \phi \\ \forall \emptyset\ \phi & \to & \phi & \forall \vec{x} \forall \vec{y}\ \phi & \to & \forall (\vec{x} \cup \vec{y})\ \phi \\ \neg(s \in \zeta) & \to & s \in \neg\zeta & & & \end{array}$$

Rules (the set $\mathcal{S}_2$)

$$\begin{array}{rclrcl} \neg(\phi \wedge \psi) & \to & \neg\phi \vee \neg\psi & \neg(\phi \vee \psi) & \to & \neg\phi \wedge \neg\psi \\ \neg(\exists \vec{x}\ \phi) & \to & \forall \vec{x}\ \neg\phi & \neg(\forall \vec{x}\ \phi) & \to & \exists \vec{x}\ \neg\phi \\ \neg\neg\phi & \to & \phi & & & \end{array}$$

Schemes of Rules (the set $\mathcal{S}_3$)

$$(P_1)\ \ \forall \vec{y_1} : \phi[\forall \vec{y_2} : \psi]_u \to \forall \vec{y_1}, \vec{y_2} : \phi[\psi]_u$$

if $\vec{y_1} \neq \emptyset$ and there is no symbol $\exists$ and no symbol $\neg$ along any path to u.

$$(P_2)\ \ \exists \vec{y_1} : \phi[\exists \vec{y_2} : \psi]_u \to \exists \vec{y_1}, \vec{y_2} : \phi[\psi]_u$$

if $\vec{y_1} \neq \emptyset$ and there is no symbol $\forall$ and no symbol $\neg$ along the path to u.

For these two last rules (schemes) it is assumed that no variable of $\vec{y_2}$ occurs free in $\phi[\top]_u$; if this is the case, we assume some renaming of the variables in $\vec{y_2}$.

Figure 1: Normalization of equational formulae

$[\![\cdot]\!]$ *is* admissible *if it is decomposable and if there is a terminating algorithm which decides, for every sort expression ζ, whether $[\![\zeta]\!]$ is finite or not and, in case of finiteness, effectively computes $[\![\zeta]\!]$.*

Admissibility of $[\![\cdot]\!]$ is all we need for the quantifier elimination procedures. However, it is worthwhile to give some examples of interpretations which fulfill these conditions. We give below the example of regular tree languages, which correspond to order-sorted algebras. Actually, this is the only example of admissible interpretation (S. Tison, private communication). We also give in section 8 another interpretation which corresponds to some order-sorted signatures with term declarations. This interpretation is borrowed from Bogaert and Tison (1992): the sort symbols are interpreted as members of the $\mathrm{REC}_=$ class which is closed under the Boolean operations. However, the corresponding mapping $[\![\cdot]\!]$ is not decomposable; we need a slight extension of decomposability. But the results extend to this interpretation in a straightforward way.

2.2 Interpretation of sorts as regular tree languages

Let $\mathcal{A} = (S, S_f, F, P)$ be a finite tree automaton (or, equivalently, an order-sorted signature). Let $[\![\underline{s}]\!]_{\mathcal{A}}$ be the language recognized by $\mathcal{A}$ in the state $\underline{s}$. $[\![\cdot]\!]_{\mathcal{A}}$ is lifted to sort expressions as above.

For simplicity, we may assume in this section that there are two particular sorts $\bot_S$ and $\top_S$ in S, which are interpreted as $\emptyset$ and $T(F)$ respectively (this interpretation allows the confusion with the sort expressions $\bot_S$ and $\top_S$ respectively). If this is not the case, then we only have to replace $\mathcal{A}$ with the automaton $\mathcal{A}' = (S', S_f, F, P')$ where $S' = S \cup \{\top_S, \bot_S\}$ and P' is the set of productions P augmented by the production rules

$$\begin{array}{rcll} f(\top_S, \ldots, \top_S) & \rightarrow & \top_S & \text{for every function symbol } f \in F \\ f(\top_S, \ldots, \bot_S, \ldots, \top_S) & \rightarrow & \bot_S & \text{for every function symbol } f \in F \\ \underline{s} & \rightarrow & \top_S & \text{for every } \underline{s} \in S \\ \bot_S & \rightarrow & \underline{s} & \text{for every } \underline{s} \in S \end{array}$$

Of course, every sort expression denotes a regular tree language; for every $\zeta \in \mathcal{E}(S)$, it is easy to construct an automaton which recognizes $[\![\zeta]\!]_{\mathcal{A}}$. Therefore, emptiness and finiteness of $[\![\zeta]\!]_{\mathcal{A}}$ are decidable in the usual way.

Let us show now that the decomposability property is also satisfied:

Lemma 1 $[\![\cdot]\!]_{\mathcal{A}}$ *is decomposable and hence admissible.*

Proof

We assume that f is a non-constant function symbol. It is easy to construct $f^{-1}(\zeta)$ when ζ does not involve a negation :

- $f^{-1}(\underline{s}) = \{(\underline{s}_1, \ldots, \underline{s}_n) \in S^n \mid f(\underline{s}_1, \ldots, \underline{s}_n) \rightarrow^+_{P'} \underline{s}\}$ ($\rightarrow^+$ denotes the transitive closure of $\rightarrow$. Therefore, $u \rightarrow^+_{P'} v$ means that u can be reduced in one or more moves of the automaton)
- $f^{-1}(g(\zeta_1, \ldots, \zeta_m)) = \emptyset$ if $f \neq g$
- $f^{-1}(f(\zeta_1, \ldots, \zeta_n)) = \{(\zeta_1, \ldots, \zeta_n)\}$
- $f^{-1}(\zeta \wedge \zeta') = \{(\zeta_1 \wedge \zeta'_1, \ldots, \zeta_n \wedge \zeta'_n) \mid (\zeta_1, \ldots, \zeta_n) \in f^{-1}(\zeta), (\zeta'_1, \ldots, \zeta'_n) \in f^{-1}(\zeta')\}$
- $f^{-1}(\zeta \vee \zeta') = f^{-1}(\zeta) \cup f^{-1}(\zeta')$

In order to construct $f^{-1}(\neg\zeta)$, we may assume that either $\zeta \in S$ or $\zeta = g(\zeta_1, \ldots, \zeta_m)$ (because $\neg(\zeta_1 \wedge \zeta_2) = \neg(\zeta_1) \vee \neg(\zeta_2), \ldots$). $f^{-1}(\neg g(\zeta_1, \ldots, \zeta_m)) = f^{-1}(\top_S)$ if $f \neq g$ and $f^{-1}(\neg f(\zeta_1, \ldots, \zeta_n))$ is the set with n elements $\{(\neg\zeta_1, \top_S, \ldots, \top_S), \ldots (\top_S, \ldots, \top_S, \neg\zeta_n)\}$. Therefore, it only remains to define $f^{-1}(\neg\underline{s})$ when $\underline{s} \in S$. Consider the usual way for constructing the automaton which recognizes the complement of a rational language: $\mathcal{A}_{d,\underline{s}} = (S_d, S_{fd}, F, P_d)$ is the deterministic (bottom-up) automaton whose states are finite subsets of S, whose final states are the sets which contain $\underline{s}$ and transitions are defined as usual. Then the automaton $(S_d, S_d - S_{fd}, F, P_d)$ recognizes $[\![\neg\underline{s}]\!]_{\mathcal{A}}$. Now, each finite set of states Q is identified with the sort expression

$$\bigwedge_{q \in Q} q \bigwedge_{q \notin Q} \neg q$$

and $f^{-1}(\underline{s})$ is the set of n-uples $(\zeta_1, \ldots, \zeta_n)$ such that $f(\zeta_1, \ldots, \zeta_n) \rightarrow^{+}_{P_d} q \in S_d - S_{fd}$.

Now, we prove by induction on the depth of the sort expression ζ that, for every $f \in F$, for every ground terms $t_1, \ldots, t_n$, for every sort expression ζ, $f(t_1, \ldots, t_n) \in [\![\zeta]\!]_{\mathcal{A}}$ iff there is a tuple $(\zeta_1, \ldots, \zeta_n) \in f^{-1}(\zeta)$ such that $t_1 \in [\![\zeta_1]\!]_{\mathcal{A}} \wedge \ldots \wedge t_n \in [\![\zeta_n]\!]_{\mathcal{A}}$.

We have to investigate a number of cases:

- $\zeta \in S$. $f(t_1, \ldots, t_n) \in [\![\zeta]\!]_{\mathcal{A}}$ iff $f(t_1, \ldots, t_n) \rightarrow^*_P \zeta$, by definition. In this reduction sequence, consider the last rule whose left hand side has a depth 1 (such a rule does exist because $f(t_1, \ldots, t_n)$ has a depth larger or equal to 1): $f(t_1, \ldots, t_n) \rightarrow^*_P f(\zeta_1, \ldots \zeta_n) \rightarrow_P \zeta' \rightarrow^*_P \zeta$. By definition, $(\zeta_1, \ldots, \zeta_n) \in f^{-1}(\zeta)$ and, for all i, $t_i \rightarrow^*_P \zeta_i$.

- $\zeta = g(\zeta_1, \ldots, \zeta_m)$ with $f \neq g$. For all $f \neq g$, $f(t_1, \ldots, t_n) \notin [\![g(\zeta_1, \ldots, \zeta_m)]\!]_{\mathcal{A}}$.

- $\zeta = f(\zeta_1, \ldots, \zeta_m)$. By definition, $f(t_1, \ldots, t_n) \in [\![f(\zeta_1, \ldots, \zeta_n)]\!]_{\mathcal{A}}$ iff for all i, $t_i \in [\![\zeta_i]\!]_{\mathcal{A}}$.

- $\zeta = \zeta_1 \wedge \zeta_2$.

$$\begin{array}{rcl} f(t_1, \ldots, t_n) \in [\![\zeta]\!]_{\mathcal{A}} & \Leftrightarrow & f(t_1, \ldots, t_n) \in [\![\zeta_1]\!]_{\mathcal{A}} \wedge f(t_1, \ldots, t_n) \in [\![\zeta_2]\!]_{\mathcal{A}} \\ & \Leftrightarrow & \exists(\zeta_{1,1}, \ldots, \zeta_{1,n}) \in f^{-1}(\zeta_1),\ \forall j, t_j \in [\![\zeta_{1,j}]\!]_{\mathcal{A}} \\ & & \wedge\exists(\zeta_{2,1}, \ldots, \zeta_{2,n}) \in f^{-1}(\zeta_2), \forall j, t_j \in [\![\zeta_{2,j}]\!]_{\mathcal{A}} \\ & & \qquad \text{by induction hypothesis} \\ & \Leftrightarrow & \exists(\zeta_{1,1}, \ldots, \zeta_{1,n}) \in f^{-1}(\zeta_1), \exists(\zeta_{2,1}, \ldots, \zeta_{2,n}) \in f^{-1}(\zeta_2), \\ & & \forall j,\ t_j \in [\![\zeta_{1,j} \wedge \zeta_{2,j}]\!]_{\mathcal{A}} \end{array}$$

- $\zeta = \zeta_1 \vee \zeta_2$. This case is straightforward: we use the fact that, for every mapping f and every subsets S_1, S_2 of the codomain of f, $f^{-1}(S_1 \cup S_2) = f^{-1}(S_1) \cup f^{-1}(S_2)$.

- $\zeta = \neg(\zeta_1 \wedge \zeta_2)$ or $\zeta = \neg(\zeta_1 \vee \zeta_2)$. We are back to the above case after distributing the negation.

- $\zeta = \neg g(\zeta_1, \ldots, \zeta_m)$ where $g \neq f$. This is straightforward

- $\zeta = \neg f(\zeta_1, \ldots, \zeta_n)$. Then

$$\begin{array}{rcl} f(t_1, \ldots, t_n) \in [\![\zeta]\!]_{\mathcal{A}} & \Leftrightarrow & f(t_1, \ldots, t_n) \notin [\![f(\zeta_1, \ldots, \zeta_n)]\!]_{\mathcal{A}} \\ & \Leftrightarrow & \neg(t_1 \in [\![\zeta_1]\!]_{\mathcal{A}} \wedge \ldots \wedge t_n \in [\![\zeta_n]\!]_{\mathcal{A}}) \\ & \Leftrightarrow & t_1 \notin [\![\zeta_1]\!]_{\mathcal{A}} \vee \ldots \vee t_n \notin [\![\zeta_n]\!]_{\mathcal{A}} \end{array}$$

- $\zeta = \neg\underline{s}'$ where $\underline{s}' \in S$. The construction is similar to the case $\underline{s} \in S$. We only have to notice that the language recognized in the state $q \in S$ is the language $\bigcup_{q \in Q} [\![Q]\!]_{\mathcal{A}_d}$, by construction of the deterministic automaton. □

Note that there are several possible definitions of $f^{-1}(\zeta)$ in the case of order-sorted signatures. We did not try to give a minimal one (minimal with respect to the number of elements in $f^{-1}(\zeta)$). For example, we could define $f^{-1}(\bot_S) = \emptyset$ and $f^{-1}(\top_S) = \{(\top_S, \ldots, \top_S)\}$.

2.3 Semantics of Equational Formulae

$\equiv$ denotes the syntactic equality on terms. A (ground) *substitution* σ is a mapping from a finite set $Dom(\sigma) \subseteq X$ into $T(F)$. A substitution can be extended in a unique way into a homomorphism from $T(F,X)$ into $T(F,X)$ that satisfies $x\sigma \equiv x$ for $x \notin Dom(\sigma)$. As usual, we make no difference between the substitution and its extension to $T(F,X)$. Σ denotes the set of all (ground) substitutions.

$\mathcal{U}$ is a finite set of variables (the *principal unknowns*) that is supposed to contain the free variables of the equational formula ψ under consideration. (Initially, $\mathcal{U}$ is the set of free variables of the equational formula to be solved. Then, the inclusion is always satisfied because we will never introduce free variables).

The set of *solutions* $\mathcal{S}(\psi,\mathcal{U},[\![\cdot]\!])$ of ψ with respect to $\mathcal{U}$ and $[\![\cdot]\!]$ is inductively defined as follows:

$$\begin{array}{rcl}
\mathcal{S}(\top,\mathcal{U},[\![\cdot]\!]) & = & \{\sigma \in \Sigma \mid Dom(\sigma) = \mathcal{U}\} \\
\mathcal{S}(s = t,\mathcal{U},[\![\cdot]\!]) & = & \{\sigma \in \Sigma \mid Dom(\sigma) = \mathcal{U} \text{ and } s\sigma \equiv t\sigma\} \\
\mathcal{S}(t \in \zeta,\mathcal{U},[\![\cdot]\!]) & = & \{\sigma \in \Sigma \mid Dom(\sigma) = \mathcal{U}, t\sigma \in [\![\zeta]\!]\} \\
\mathcal{S}(\psi_1 \wedge \psi_2,\mathcal{U},[\![\cdot]\!]) & = & \mathcal{S}(\psi_1,\mathcal{U},[\![\cdot]\!]) \cap \mathcal{S}(\psi_2,\mathcal{U},[\![\cdot]\!]) \\
\mathcal{S}(\exists x:\ \psi,\mathcal{U},[\![\cdot]\!]) & = & \{\sigma_{|\mathcal{U}-\{x\}} \mid \sigma \in \mathcal{S}(\psi,\mathcal{U}\cup\{x\},[\![\cdot]\!])\} \\
\mathcal{S}(\neg\psi,\mathcal{U},[\![\cdot]\!]) & = & \mathcal{S}(\top,\mathcal{U},[\![\cdot]\!]) - \mathcal{S}(\psi,\mathcal{U},[\![\cdot]\!])
\end{array}$$

The definition for other logical connectives can easily be derived from the above ones. It is easy to see that this definition is independent of the representative in an equivalence class of equational formulae.

Two formulae are *semantically equivalent* (w.r.t. $[\![\cdot]\!], \mathcal{U}$) if they have the same set of solutions.

The above definitions imply that $s = t$ may have a solution σ whereas $t\sigma \notin \bigcup_{\underline{s}\in S}[\![\underline{s}]\!]$. In other words, any term t is considered as well-typed unless a statement $t \in \zeta$ leads to a contradiction (this corresponds to the assumption $\top_S \in S$ we made in the previous section). Therefore, if one wants to restrict the solutions of an equation to substitutions satisfying the requirement $t\sigma \in \bigcup_{\underline{s}\in S}[\![\underline{s}]\!]$, this has to be explicitly stated in the problem, using a membership constraint. This makes a difference with usual order-sorted unification algorithms in which only well-sorted terms are considered (Kirchner 1988). In the latter case, one has to be careful, for instance with replacement rules, in order to avoid the production of ill-sorted terms. In our opinion, our framework is simpler since it separates sort deduction and reasoning about terms: here, these two rules can be mixed in an arbitrary way.

3 Solved Forms

Solved forms are particular formulae that follow the following requirements (Comon 1991):

Solvability Each solved form (distinct from $\bot$) must have at least one solution

Completeness Each formula must be (semantically) equivalent to a solved form

Simplicity It must be "easy" to enumerate the set of solutions of a solved form

The last property lies on the definition of "easy". We do not want to discuss this notion here. Thus, we only consider the first two properties, claiming that our solved forms fulfill the last requirement.

Definition 2 *A* definition with constraints *is either $\bot$, $\top$ or a disjunction of existential formulae of the form*

$$\exists \vec{w}: \ x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_k \neq u_k \wedge y_1 \in \zeta_1 \wedge \ldots \wedge y_m \in \zeta_m$$

where

- $x_1, \ldots, x_n$ *are free variables that occur only once in the formula*
- $z_1, \ldots, z_k$ *are variables such that, for all* i, $z_i \notin \mathcal{V}ar(u_i)$
- $y_1, \ldots, y_m$ *are distinct variables*
- *for all* i $[\![\zeta_i]\!]$ *is infinite.*

The fact that definitions with constraints indeed satisfy the solvability conditions is a consequence of a kind of "independence of disequations lemma" (Mal'cev 1971, Lassez, Maher, and Marriot 1986, Colmerauer 1984, Comon and Lescanne 1989):

Lemma 2 *Let* $P \equiv s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n$ *where, for all* i, $s_i \not\equiv t_i$. *Let* $\{x_1, \ldots, x_m\} = \mathcal{V}ar(P)$ *and* $A_1, \ldots, A_m$ *be* m *infinite subsets of* $T(F)$. *Then* P *has a solution* σ *that satisfies* $\forall i, x_i\sigma \in A_i$.

Proof

This lemma is proved by induction on m, noticing that an equation $s = t$ with $s \not\equiv t$ and $|\mathcal{V}ar(s,t)| = 1$ has at most one solution.

When $m = 0$, the result is obvious because every disequation holds. Now, assume that the lemma holds for $m - 1$.

Every equation $s_i = t_i$ has at most one solution w.r.t. x_m, when $x_1, \ldots, x_{m-1}$ are considered as constants. Therefore, the set

$$B_m = \{u \mid \exists i \in \{1, \ldots, n\}, \ s_i\{x_m \mapsto u\} \equiv t_i\{x_m \mapsto u\}\}$$

is finite. Let $\sigma = \{x_m \mapsto t_m\}$ be a substitution such that $t_m \in A_m - B_m$. Such a substitution does exist because A_m is assumed to be infinite. For each i, $s_i\sigma \not\equiv t_i\sigma$ by construction and, since t_m is a ground term, $|\bigcup_{i \neq j} \mathcal{V}ar(s_j\sigma, t_j\sigma)| < m$. Thus, by induction hypothesis, there is a substitution θ such that, for all i, $s_i\sigma\theta \not\equiv t_i\sigma\theta$ and, for $j \leq m-1$, $x_j\theta \in A_j$. Then the substitution $\sigma\theta$ is a solution of P that satisfies $\forall i, x_i\sigma\theta \in A_i$. $\Box$

Corollary 1 *Every definition with constraints distinct from $\bot$ has at least one solution.*

Proof

This result is actually a consequence of lemma 2. Indeed, let $A_i = [\![\zeta_i]\!]$, for $i = 1, \ldots, m$ (we follow the notations of definition 2). By hypothesis, the sets A_i are infinite and θ is a solution of $y_1 \in \zeta_1 \wedge \ldots \wedge y_m \in \zeta_m$ iff, for each i, $y_i\theta \in A_i$ (because $y_1, \ldots, y_m$ are distinct). Then, by lemma 2, there is a solution ρ to $z_1 \neq u_1 \wedge \ldots \wedge y_n \in \zeta_m$. Finally, let σ be the substitution $\{x_1 \mapsto t_1; \ \ldots \ ; x_n \mapsto t_n\}$. Then $\sigma\rho$ is a solution of $x_1 = t_1 \wedge \ldots \wedge y_m \in \zeta_m$. $\Box$

We will see in section 6 that definitions with constraints also fulfill the second requirement (completeness).

4 Transformation rules

In what follows (and until section 7.2), $[\![\cdot]\!]$ is any admissible semantic mapping from $\mathcal{E}(S)$ to $2^{T(F)}$.

A *transformation rule* is the denotation of a (possibly infinite) rewrite system in the algebra $\mathcal{EF}$. For the rules proposed in this paper, the rewrite system they denote is always self-evident. In the description of the rules, we use the following conventions:

- $P\{x \mapsto t\}$ stands for the problem P in which all free occurrences of x in P have been replaced by t.
- When x is replaced by t, the variables of t must not be captured. If necessary, the corresponding bound variables are assumed to be renamed.

The set $\mathcal{R}$ of rules described in figures 2, 3 and 4 consists of a set of transformations of atomic formulae (figure 2), a set of quantifier rules (figure 3), distributivity rules and explosion rules (figure 4). They are given together with the "minimal" control for which we can prove termination. This is a very important feature of our system: there are many possible algorithms that can be derived from the set of rules by strengthening the control. All these algorithms are terminating.

Let us comment a little bit more the rules in $\mathcal{R}$.

- $(D_1), (D_2)$ are classical decomposition rules.
- (D_3) is a decomposition rule resulting from the decomposability property of $[\![\cdot]\!]$.
- $(C_1), (C_2)$ are classical clash rules.
- $(S_1), \ldots, (S_8)$ are straightforward rules on membership conditions.
- $(O_1), (O_2)$ are occur-checks.
- $(Q_1), (Q_2), (Q_3), (Q_4)$ are structural rules on quantifiers. We do not impose the prenex form. The notation $\vec{x}$ (or $\vec{y}$) allows to use any (non-empty) set of bound variables (including the case of a single variable), provided that the condition on Q_1, Q_2 is fulfilled. This condition is necessary: removing it yields a loop using Q_1, Q_2 and $\mathcal{S}_3$.
- Rules $(Q_5), \ldots, (Q_{10})$ are quantifier elimination rules. They are the core of the transformation: termination as well as completeness will follow from their eventual application.
- $(N_1), (N_2)$ are distributivity rules. As announced in the introduction, we do not require their application at each step of transformation: it is now possible to avoid unnecessary duplications. Note that (N_1) and (N_2) apply the distributivity in opposite ways; the way the distributivity is applied depends on the nearest quantifier. On the other hand, we assume that the formulae involved in the distributivity are quantifier-free. This is a restriction (already assumed in Comon (1990)) which we were unable to remove.
- $(Ex_1), (Ex_2)$ express the domain closure axiom: we know from (Comon and Lescanne 1989, Maher 1988) that there should be somewhere such rules. The problem will be, of course, that these rules produce more complex formulae.
- Some classical rules such as $s = s \to \top$ are missing here. Indeed, let us recall that the formulae on which the rules are applied are assumed to be in normal form w.r.t. the rules of figure 1, which includes removing such trivial equations.

$$
\begin{array}{llcl}
(D_1) & f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n) & \to & t_1 = u_1 \wedge \ldots \wedge t_n = u_n \\
(D_2) & f(t_1,\ldots,t_n) \neq f(u_1,\ldots,u_n) & \to & t_1 \neq u_1 \vee \ldots \vee t_n \neq u_n \\
(D_3) & f(t_1,\ldots,t_n) \in \zeta & \to & \bigvee_{(\zeta_1,\ldots,\zeta_n)\in f^{-1}(\zeta)} t_1 \in \zeta_1 \wedge \ldots \wedge t_n \in \zeta_n
\end{array}
$$

$$
\begin{array}{llcl}
(C_1) & f(t_1,\ldots,t_n) = g(u_1,\ldots,u_m) & \to & \perp \\
(C_2) & f(t_1,\ldots,t_n) \neq g(u_1,\ldots,u_m) & \to & \top
\end{array}
$$

If $f \neq g$

$$
\begin{array}{llcl}
(S_1) & t \in \zeta \wedge t \in \zeta' & \to & t \in \zeta \wedge \zeta' \\
(S_2) & t \in \zeta \vee t \in \zeta' & \to & t \in \zeta \vee \zeta' \\
\\
(S_3) & t \in \zeta_0 & \to & \perp \\
(S_4) & t \in \zeta_1 & \to & \top
\end{array}
$$

If $[\![\zeta_0]\!] = \emptyset$ and $[\![\zeta_1]\!] = T(F)$.

$$
(S_5) \quad t \in \zeta \quad \to \quad t = t_1 \vee \ldots \vee t = t_n
$$

If $[\![\zeta]\!] = \{t_1,\ldots,t_n\}$ is finite.

$$
(S_6) \quad t \in \zeta \quad \to \quad t \neq t_1 \wedge \ldots \wedge t \neq t_n
$$

If $[\![\neg\zeta]\!] = \{t_1,\ldots,t_n\}$ is finite.

$$
\begin{array}{llcll}
(S_7) & t \in \zeta & \to & \top & \text{If } t \in [\![\zeta]\!] \\
(S_8) & t \in \zeta & \to & \perp & \text{If } t \in [\![\neg\zeta]\!]
\end{array}
$$

$$
\begin{array}{llcl}
(O_1) & t[s] = s & \to & \perp \\
(O_2) & t[s] \neq s & \to & \top
\end{array}
$$

If t is a non-trivial context

Figure 2: Decomposition, sort expression and occur check rules

$$
\begin{array}{llcl}
(Q_1) & \exists \vec{x}: \; P \vee Q & \rightarrow & (\exists \vec{x}: \; P) \vee (\exists \vec{x}: \; Q) \\
(Q_2) & \forall \vec{y}: \; P \wedge Q & \rightarrow & (\forall \vec{y}: \; P) \wedge (\forall \vec{y}: \; Q) \\
(Q_3) & \exists \vec{z}: \; P & \rightarrow & P \\
(Q_4) & \forall \vec{z}: \; P & \rightarrow & P
\end{array}
$$

If $\vec{z} \cap \mathcal{V}ar(P) = \emptyset$ and $\vec{x} \cap \mathcal{V}ar(P, Q) \neq \emptyset$. Q_1, Q_2 also assume that they do not introduce a redex for P_1, P_2 (from $\mathcal{S}_3$).

$$
\begin{array}{llcl}
(Q_5) & \forall y: \; d \vee y \neq t & \rightarrow & d\{y \mapsto t\} \\
(Q_6) & \exists x: \; c \wedge x = t & \rightarrow & c\{x \mapsto t\}
\end{array}
$$

If $x \notin \mathcal{V}ar(t)$ (resp. $y \notin \mathcal{V}ar(t)$).

$$(Q_7) \quad \forall \vec{y}: z_1 = u_1 \vee \ldots \vee z_n = u_n \vee y_1 \in \zeta_1 \vee \ldots \vee y_m \in \zeta_m \vee d \quad \rightarrow \quad d$$

If

- $z_1, \ldots, z_n$ are variables ($n \geq 0$)
- for all i, $z_i \not\equiv u_i$
- for all i, $\mathcal{V}ar(z_i, u_i) \cap \vec{y} \neq \emptyset$
- $\vec{y} \cap \mathcal{V}ar(d) = \emptyset$
- $y_1, \ldots, y_m \in \vec{y}$
- The rules (S_i) cannot be applied to $y_1 \in \zeta_1 \vee \ldots \vee y_m \in \zeta_m$

$$(Q_8) \quad \exists \vec{x}: \; z_1 \neq u_1 \wedge \ldots \wedge z_n \neq u_n \wedge x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m \wedge d \quad \rightarrow \quad d$$

If

- $z_1, \ldots, z_n$ are variables ($n \geq 0$)
- for all i, $z_i \not\equiv u_i$
- for all i, $\mathcal{V}ar(z_i, u_i) \cap \vec{x} \neq \emptyset$
- $\vec{x} \cap \mathcal{V}ar(d) = \emptyset$
- $x_1, \ldots, x_m \in \vec{x}$
- the rules (S_i) cannot be applied to $x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m$.

Figure 3: Quantifier rules

$$
\begin{array}{llcl}
(N_1) & \forall \vec{y}: \; C[(P_1 \wedge P_2) \vee Q]_u & \rightarrow & \forall \vec{y}: \; C[(P_1 \vee Q) \wedge (P_2 \vee Q)]_u \\
(N_2) & \exists \vec{x}: \; C[(P_1 \vee P_2) \wedge Q]_u & \rightarrow & \exists \vec{x}: \; C[(P_1 \wedge Q) \vee (P_2 \wedge Q)]_u
\end{array}
$$

If P_1, P_2, Q are quantifier-free, $\vec{y} \cap \mathcal{V}ar(P_1) \neq \emptyset$ (resp. $\vec{x} \cap \mathcal{V}ar(P_1) \neq \emptyset$), $P_1(\Lambda) \neq \wedge$ (resp. $P_1(\Lambda) \neq \vee$), there is no redex for N_1, N_2 inside P_1 and there is no symbol $\neg$ and no symbol $\exists, \forall$ along the path to position u.

$$
(Ex_1) \quad \exists \vec{w_1}: \; P \quad \rightarrow \quad \bigvee_{f \in F} \exists \vec{w_1}, \exists \vec{w}: \; x = f(\vec{w}) \wedge P\{x \mapsto f(\vec{w})\}
$$

If x is free in P, no other rule but (Ex_2) can be applied and there is an expression $x \neq t[y]$ or $x = t[y]$ where y is universally bound in P. $\vec{w_1}$ is assumed to be non-empty.

$$
(Ex_2) \quad \forall \vec{y_1}: \; P \quad \rightarrow \quad \bigwedge_{f \in F} \forall \vec{y_1}, \forall \vec{y}: \; x \neq f(\vec{y}) \vee P\{x \mapsto f(\vec{y})\}
$$

If x is free in P, there is an expression $x = t[z]$ or $x \neq t[z]$ in P such that z is existentially bound in P and no other rule can be applied. $\vec{y_1}$ is assumed to be non-empty.

Figure 4: Distributivity and explosion rules

Note that the set of rules is perfectly symmetric w.r.t. $\exists$ and $\forall$ except in one case : the rule (Ex_1) can be applied on a purely universal formula whereas (Ex_2) cannot be applied on purely existential formulae. This slight difference explains the fact that irreducible formulae (w.r.t. the set of rules given in figures 2, 3, 4) are purely existential ones (see section 6).

A rule R is *correct* if, for any equational formulae ψ and ψ', $\psi \rightarrow_R \psi'$ implies that $\mathcal{S}(\psi, \mathcal{U}, \mathcal{A}) = \mathcal{S}(\psi', \mathcal{U}, \mathcal{A})$.

Proposition 1 *All rules given in figures 2, 3, 4 are correct.*

Proof

For most of the rules the correctness is easy to prove. For the rule (D_3), the correctness is a consequence of the decomposability of $[\![\cdot]\!]$. For the rules Q_7, Q_8, the correctness is a consequence of lemma 2. Indeed, let us consider, for example, (Q_8). Let θ be a solution of d w.r.t. the set of principal unknowns $\mathcal{U} \supseteq \mathcal{V}ar(z_1, \ldots, z_n, u_1, \ldots, u_n, d)$ such that $\mathcal{U} \cap \vec{x} = \emptyset$ and $c \equiv x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m$. Since the rules (S_i) do not apply, for every i, $[\![\zeta_i]\!]$ and $[\![\neg\zeta_i]\!]$ must be infinite. Now, from lemma 2, $z_1\theta \neq u_1\theta \wedge \ldots \wedge z_n\theta \neq u_n\theta$ has a solution σ whose domain is $\vec{x}$ and such that, for all $x \in \vec{x}$, $x \equiv x_i \Rightarrow x\sigma \in [\![\zeta_i]\!]$. This proves that θ is a solution of the left hand side. (The converse inclusion is straightforward). $\square$

5 Elimination of quantifiers

For sake of clarity, we split the transformation into two steps: we first eliminate the quantifiers showing the system $\mathcal{R}$ defined by the rules of figures 2, 3 and 4 is terminating. Then, we show in the following section that any irreducible formula w.r.t. $\mathcal{R}$ is a purely existential formula and we exhibit a system $\mathcal{R}'$ which transforms any purely existential formula into a solved form. (A formula is *purely existential* if its $\mathcal{S}_1 - \mathcal{S}_2$ normal form does not involve any $\forall$.)

Theorem 1 *The system $\mathcal{R}$ defined by the rules given in figures 2, 3, 4 does terminate.*

This is the main result (and the most difficult) of this paper. The whole section is devoted to its proof. The reader who is not interested in this proof might skip it and read directly section 6.

It could be possible to add some control to our rule system, thus yielding to a simpler termination proof. However, we aim at proving the termination, leaving as much freedom as possible for the application of the rules.

Intuitively, the termination stems from quantifier elimination: bound variables are popped up until they appear as a member of an atomic formula. Then they are eliminated using one of the rules Q_5, Q_6, Q_7, Q_8. The problem arises with this lifting process. Decomposition rules D_1, D_2, D_3 reduce the atomic formulae until (at least) one of their members is a variable. In a unification procedure, this would be sufficient for eliminating the variable. However, this is not sufficient here, because x may be a member of an equation $x = f(y)$ where y is bound below x, thus preventing the elimination of x by replacing it with $f(y)$. Consider for example the formula:

$$\phi \equiv \exists x \forall y : \ x \neq f(y) \wedge x = z$$

x cannot be eliminated using the rules Q_5, Q_6, Q_7, Q_8. Then, the idea is to try to lift an *innermost* quantified variable (y in the example). To "lift" y means to transform the formula in such a way that y occurs as a member of an equation or a disequation. Then, the (Q_i) rules will apply on y. Actually, we know from previous works (Mal'cev 1971, Comon and Lescanne 1989, Maher 1988) that we have to consider somewhere the *domain closure axiom*. And, indeed, in the above example, expressing that x has to start with 0 or f (assuming that F has only two elements), we get the reduction:

$$\begin{array}{l} \phi \rightarrow_{Ex_1} (\exists x : \ x = 0 \wedge (\forall y : \ (0 \neq f(y) \wedge z = 0))) \\ \qquad\quad \vee(\exists w, x : \ x = f(w) \wedge (\forall y : \ (f(w) \neq f(y) \wedge z = f(w)))) \end{array}$$

Now, the first part of the formula reduces to $z = 0$ and, in the second part, $f(w) \neq f(y)$ becomes $w \neq y$: y has been popped up to the top where it can be eliminated using the rule Q_5.

That is the general way to proceed: Ex_1, Ex_2 are used in order to lift a variable which is bound at a nested depth of quantification n. This step may introduce arbitrarily many new quantified variables, but these new variables are bound at a nested depth strictly smaller than n.

Other problems come from the "context-sensitive" rules: for example, the distributivity is applied in one way if the innermost quantifier is an $\exists$ and it is applied in the reverse way if the innermost quantifier is an $\forall$. In order to deal with such situations, we first interpret the formula, replacing $\forall$ with $\neg\exists\neg$ and normalizing the resulting formula. The ordering on the resulting formula is a reduction relation associated with a set of rules $\mathcal{R}_1$.[3] We first define $\mathcal{R}_1$ (section 5.1) and prove the termination of $\mathcal{R}_1$ without the explosion rule (section 5.2). For this, we use the *associative path ordering* of Bachmair and Plaisted (1985). Then (section 5.3) we show how to handle the explosion rule: we show that there is some interpretation in a well-founded domain which is decreasing on the subsequence of formulae on which explosion is applied (this

[3]One can argue that we do not really need $\mathcal{R}$: we could replace it with $\mathcal{R}_1$, thus avoiding a "coding" step. However, if it is true that the termination proof would become simpler, the use of both quantifiers is more natural in expressing some (first-order) problems. Then the coding part would take place at this level, each time the system is used. That is why we prefered the use of both quantifiers for the system $\mathcal{R}$.

is the most difficult step). Finally, we show how the termination of $\mathcal{R}$ can be reduced to the termination of $\mathcal{R}_1$ (section 5.4).

5.1 The system $\mathcal{R}_1$

Actually, the system $\mathcal{R}_1$ is obtained from the original system, by a syntactic translation of the formulae: we encode the quantifier $\forall$ by $\neg\exists\neg$. More precisely, the syntax of the formulae is defined as follows: $\mathcal{EF}'$ is the smallest set of formulae such that:

- $\top$, $\bot$, $s = t$, $s \neq t$, $t \in \zeta \in \mathcal{EF}'$ for any terms s, t and any sort expression ζ
- $\neg P, P \wedge Q, P \vee Q \in \mathcal{EF}'$ if $P, Q \in \mathcal{EF}'$
- $\exists \vec{x} : P \in \mathcal{EF}'$ if $P \in \mathcal{EF}'$

We assume moreover (as for $\mathcal{EF}$) some syntactic identities between formulae, namely the sets $\mathcal{S}_1, \mathcal{S}_3$ of identities described in figure 1, section 1. Formulae are confused with their normal form with respect to $\mathcal{S}_1$ and $\mathcal{S}_3$ (applied in this order).

The system of rules $\mathcal{R}_1$ is then described in figures 2 and 5.

5.2 Termination of $\mathcal{R}_1$, without explosion

We are going to show how to use the associative path ordering of Bachmair and Plaisted (1985) on an interpretation of the formulae, to prove the termination of $\mathcal{R}_1 - \{(Ex)\}$.

The interpretation is defined as follows:

- First, we assume that every variable is bound at most once in each formula (this may require some renamings).
- Let g, h, a be three function symbols which are not in F. If ϕ is a formula in which x is bound at position p: $\phi|_p \equiv \exists x, \vec{x} : \psi$, then let $N(x, \phi)$ be one plus the number of positions which are prefixes of p and such that $\phi(p) = \neg$. ($N(x, \phi)$ measures the number of quantifier alternations above x.) If x occurs only free in ϕ, then we choose $N(x, \phi) = 0$. Let $sl(x, \phi)$ be the number of variables which are bound at the same position as x, or the number of free variables, if x is free in ϕ.
- Let $G = F \cup \{\exists(1), \in(1), =(2), \neq(2), g(2), h(1), a(0), \top(0), \bot(0), \wedge(2), \vee(2), \neg(1)\}$ (the number in parentheses indicate the arity) and let I be the mapping from $(T(F, X) \cup \mathcal{EF}') \times \mathcal{EF}'$ into $T(G)$ which is defined as follows:
  - $I(P \wedge Q, \phi) \stackrel{\text{def}}{=} I(P, \phi) \wedge I(Q, \phi)$
  - $I(P \vee Q, \phi) \stackrel{\text{def}}{=} I(P, \phi) \vee I(Q, \phi)$
  - $I(\neg P, \phi) \stackrel{\text{def}}{=} \neg I(P, \phi)$
  - $I(s = t, \phi) \stackrel{\text{def}}{=} I(s, \phi) = I(t, \phi)$
  - $I(\top, \phi) \stackrel{\text{def}}{=} \top$
  - $I(\bot, \phi) \stackrel{\text{def}}{=} \bot$
  - $I(s \neq t, \phi) \stackrel{\text{def}}{=} I(s, \phi) \neq I(t, \phi)$
  - $I(f(t_1, \ldots, t_n), \phi) \stackrel{\text{def}}{=} f(I(t_1, \phi), \ldots, I(t_n, \phi)) \qquad$ for every $f \in F$

$$
\begin{array}{lrcl}
(Q_1) & \exists \vec{x}: \ P \vee Q & \rightarrow & (\exists \vec{x}: \ P) \vee (\exists \vec{x}: \ Q) \\
(Q_3) & \exists \vec{z}: \ P & \rightarrow & P
\end{array}
$$

If $\vec{z} \cap \mathcal{V}ar(P) = \emptyset$ and $\vec{x} \cap \mathcal{V}ar(P, Q) \neq \emptyset$. We also assume that Q_1 does not introduce any redex for P_1.

$$(Q_6) \quad \exists x: \ c \wedge x = t \quad \rightarrow \quad c\{x \mapsto t\}$$

If $x \notin \mathcal{V}ar(t)$ (resp. $y \notin \mathcal{V}ar(t)$).

$$(Q_8) \quad \exists \vec{x}: \ z_1 \neq u_1 \wedge \ldots \wedge z_n \neq u_n \wedge x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m \wedge d \quad \rightarrow \quad d$$

If

- $z_1, \ldots, z_n$ are variables
- for all i, $z_i \notin \mathcal{V}ar(u_i)$
- for all i, $\mathcal{V}ar(z_i, u_i) \cap \vec{x} \neq \emptyset$
- $\vec{x} \cap \mathcal{V}ar(d) = \emptyset$
- $x_1, \ldots, x_m \in \vec{x}$
- the rules (S_i) cannot be applied to $x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m$

$$(N_2) \quad \exists \vec{x}: \ C[(P_1 \vee P_2) \wedge Q]_u \quad \rightarrow \quad \exists \vec{x}: \ C[(P_1 \wedge Q) \vee (P_2 \wedge Q)]_u$$

If P_1, P_2, Q are quantifier-free, $\vec{x} \cap \mathcal{V}ar(P_1) \neq \emptyset$, P_1 is not a disjunction, there is no possible reduction by N_2 inside P_1 and there is no symbol $\neg$ along the path to position u.

$$(Ex) \quad \exists \vec{w_1}: P[\neg Q[\exists \vec{y}: R]] \quad \rightarrow \quad \bigvee_{f \in F} \exists \vec{w_1}, \exists \vec{w}: \ x = f(\vec{w}) \wedge P\{x \mapsto f(\vec{w})\}$$

If x is free in P, no other rule applies and there is in R an expression $x \neq t[y]$ or $x = t[y]$. Moreover, $\vec{w_1}$ is assumed to be non-empty.

$$
\begin{array}{lrcl}
(B_1) & \neg\neg P & \rightarrow & P \\
(B_2) & \neg(P \vee Q) & \rightarrow & \neg P \wedge \neg Q \\
(B_3) & \neg(P \wedge Q) & \rightarrow & \neg P \vee \neg Q
\end{array}
$$

These boolean rules are assumed to be applied while they do apply, before any other rule.

Figure 5: The system of rules $\mathcal{R}_1$. Part 2

– $I(\exists \vec{x} : P, \phi) \stackrel{\text{def}}{=} \exists^{|\vec{x}|}(I(P, \phi))$ (i.e. $\underbrace{\exists(...(\exists}_{|\vec{x}| \text{ times}} (I(P, \phi)))\ldots))$

– $I(t \in \zeta, \phi) \stackrel{\text{def}}{=} \in(I(t, \phi))$

– $I(x, \phi) \stackrel{\text{def}}{=} g(h^{N(x,\phi)}(a), h^{sl(x,\phi)}(a))$ when x is a variable.

$\wedge$ and $\vee$ are assumed to be associative and commutative, $=$ and $\neq$ are assumed to be commutative. By taking the flattened representation of terms in $T(G)$, I is independent of the chosen representative of ϕ.

G is ordered by $\geq_G$ in the following way:

$$\neg > g > h > \in > f > a > = > \neq > \top > \perp > \exists > \wedge > \vee$$

where f is any symbol in F (two distinct symbols in F are uncomparable.)

Then $T(G)$ is ordered with the *associative path ordering* defined in (Bachmair and Plaisted 1985), using a lexicographic status for g and a multiset status for all other symbols[4]. Let us recall the definition and basic properties of this ordering in our particular example.

Let (N) be the distributivity rule

$$(N) \quad (P_1 \vee P_2) \wedge Q \quad \rightarrow \quad (P_1 \wedge Q) \vee (P_2 \wedge Q)$$

The corresponding reduction relation $\rightarrow_N$ is convergent (modulo AC). Therefore, each term t in $T(G)$ has a unique normal form $t\downarrow_N$. Now, $s >_{apo} t$ iff

- either $s\downarrow_N >_{rpo} t\downarrow_N$ where $\geq_{rpo}$ is the *recursive path ordering* on flattened terms in $T(G)$, extending the precedence $\geq_G$ with the multiset status for $\wedge$, $\vee$, $=$, $\neq$ and the lexicographic status for g. (See e.g. Dershowitz 1987, Dershowitz and Jouannaud 1990, for a definition of $\geq_{rpo}$.)

- or $s\downarrow_N \equiv t\downarrow_N$ and $|t| > |s|$.

We use here the following properties of the associative path ordering (the reader is referred to Bachmair and Plaisted (1985) for more details):

- $\geq_{apo}$ is compatible with AC identities. (i.e. $s =_{AC} s_1$, $t =_{AC} t_1$ and $s >_{apo} t$ implies $s_1 >_{apo} t_1$.)

- $\geq_{apo}$ is a well-founded ordering on AC equivalence classes of terms in $T(G)$.

- $\geq_{apo}$ has the *subterm property*: for every term t and every non-trivial context u, $u[t] >_{apo} t$

- $\geq_{apo}$ is *monotonic*: for every terms s, t and every context u, if $s >_{apo} t$, then $u[s]_p >_{apo} u[t]_p$.

- $\geq_{apo}$ can be lifted to non-ground terms in $T(G)$ ($\geq_G$ is extended by equality on variables). If x is a variable, a *critical occurrence* of x in s is an occurrence of x in a subterm $x \wedge s'$ of s. Let $\Gamma(s,t)$ be the set of the variables that have a critical occurrence in s or in t. $\Sigma(s,t)$ is now the set of substitutions σ whose domains are subsets of $\Gamma(s,t)$ and such that $x\sigma \equiv x_1 \vee x_2$ for every variable $x \in Dom(\sigma)$. (Beware: $\Sigma(s,t)$ is not a subset of Σ, as substitutions in $\Sigma(s,t)$ are not necessary ground.) If $s\sigma >_{apo} t\sigma$ for every substitution $\sigma \in \Sigma(s,t)$, then $s\theta >_{apo} t\theta$ for every substitution θ. (See Bachmair and Plaisted (1985) for more details.)[5]

[4] We use actually a straightforward extension of the associative path ordering because some symbols are commutative (with multiset status) and some symbols have a lexicographic status.

[5] According to this last property of $\geq_{apo}$, we do not have to consider all possible instances for the variables $P, Q, c, d, \ldots$ which schematize formulae in figure 5; we only have to prove the decreasingness of the rules themselves and the decreasingness of their critical instances, considering $P, Q, c, \ldots$ as constants which are unrelated by $\geq_G$.

Now, we are going to prove that any rule applied on a formula ϕ decreases $I(\phi, \phi)$ w.r.t. $\geq_{apo}$.

Lemma 3 *If $\phi \rightarrow \psi$ and the transformation introduces a redex for P_2, then $I(\phi, \phi) >_{apo} I(\psi, \psi)$.*

Proof

A redex for P_2 may occur only when the rule applied on ϕ erases a part of ϕ which includes at least one symbol $\neg$. Now, since $\neg$ is the largest symbol in the precedence, $I(\phi, \phi)\downarrow_{\mathcal{D}} >_{rpo} I(\psi, \psi)\downarrow_N$ and the inequality follows. $\Box$

From now on, in this section, we assume that reducing ϕ does not introduce any redex for P_2.

Lemma 4 *If $\phi \rightarrow \psi$ using one of the rules of figure 2, then $I(\phi, \phi) >_{apo} I(\psi, \psi)$.*

Proof

For every variable x occurring in ϕ, $N(x, \phi) \geq N(x, \psi)$ and, in case of equality, $sl(x, \phi) \geq sl(x, \psi)$. Therefore, for every variable x, $I(x, \phi) \geq_{apo} I(x, \psi)$. (In general, the interpretations are identical, except when, by a side effect, a rule such as C_1, together with the normalization rules of figure 1, removes some quantifier. In this case, either N or sl may decrease.) Therefore, it is sufficient to prove that $I(\phi, \phi) >_{apo} I(\psi, \phi)$.

By monotonicity of $\geq_{apo}$, we may assume that the rule is applied at the root of the formula. For the rules D_1, D_2, the decreasingness follows from the subterm property of $\geq_{apo}$ and the precedence $= >_G \wedge$, $\neq >_G \vee$. Similarly, for the rule (D_3), this follows from $\in >_G \vee$ and $\in >_G \wedge$.

The result is obvious for the rules $(C_1), (C_2), (S_3), (S_4), (S_7), (S_8), (O_1), (O_2)$.

For the rules (S_1), $I(t \in \zeta \wedge t \in \zeta', \phi) = \in(I(t, \phi)) \wedge \in(I(t, \phi))$ and $I(t \in \zeta \wedge \zeta', \phi) = \in(I(t, \phi))$. Therefore, decreasingness follows from the subterm property of $\geq_{apo}$. Similarly, I is decreasing for the rule (S_2).

For the rules (S_5) and (S_6), remember that $t_1, \ldots, t_n$ are ground terms. Since $\in >_G f$ for every function symbol $f \in F$, $\in(I(t, \phi)) >_{apo} u$, for every term t, every formula ϕ, and every ground term u. Finally, because $\in >_G \wedge$, $\in >_G =$, $\in >_G \vee$ and $\in >_G \neq$, $I(\phi, \phi) >_{apo} I(\psi, \phi)$. $\Box$

Lemma 5 *If $\phi \rightarrow_{(Q_1),(N_2)} \psi$, then $I(\phi, \phi) >_{apo} I(\psi, \psi)$.*

Proof

First, for every variable x, and under the hypotheses of the lemma, $I(x, \phi) \equiv I(x, \psi)$. Therefore, $I(\psi, \psi) \equiv I(\psi, \phi)$. For these three rules, $I(\psi, \phi)\downarrow_N \equiv I(\psi, \psi)\downarrow_N$. Therefore, the lemma follows straightforward from properties of the polynomial interpretation. $\Box$

Lemma 6 *If $\phi \rightarrow_{(Q_8),(B_1)} \psi$, then $I(\phi, \phi) >_{apo} I(\psi, \psi)$.*

Proof

Again, under the hypotheses of the lemma, for every variable x, $I(x, \phi) \geq_{apo} I(x, \psi)$. (Actually, there are some variables for which the inequality is strict.) Indeed, N can only decrease by application of these rules and sl may increase only when N has decreased.

Then, the lemma follows from the properties of $\geq_{apo}$ because the right hand sides of the rules considered here are embedded in the corresponding left hand sides. $\Box$

Lemma 7 *If $\phi \rightarrow_{(B_2),(B_3)} \psi$, then $I(\phi,\phi) >_{apo} I(\psi,\psi)$.*

Proof

The two rules B_2 and B_3 do not change the quantifier structure: $I(\psi,\psi) \equiv I(\psi,\phi)$. Moreover, from the monotonicity property of $\geq_{apo}$, we may assume that the rules are applied at the root of the formula. Finally, by the last property of $\geq_{apo}$, it is sufficient to prove that $l\sigma >_{apo} r\sigma$ for every substitution $\sigma \in \Sigma(l,r)$, if $l \rightarrow r$ is one of the rules $(B_2),(B_3)$.

According to the definition of $\Sigma(s,t)$, only the identity has to be considered for rule (B_2) (there is no critical substitution for this rule) and we have to consider the following substitutions for (B_3): the identity and the substitutions

$$\{P \mapsto P_1 \vee P_2\}, \{Q \mapsto Q_1 \vee Q_2\},$$

Because of symmetry w.r.t P and Q, we may drop the study of the last case.

- $\neg(P \wedge Q) >_{apo} \neg P \vee \neg Q$ because $\neg >_G \vee$. Similarly, $\neg(P \vee Q) >_{apo} \neg P \wedge \neg Q$ because $\neg >_G \wedge$.
- Let $\sigma = \{P \mapsto P_1 \vee P_2\}$. $(\neg(P \wedge Q)\sigma) \downarrow_N \equiv \neg((P_1 \wedge Q) \vee (P_2 \wedge Q))$. On the other hand, $((\neg P \vee \neg Q)\sigma) \downarrow_N \equiv \neg(P_1 \vee P_2) \vee \neg Q$. Because $\neg >_G \vee$, the former formula is larger than the latter one. □

Lemma 8 *If $\phi \rightarrow_{Q_6} \psi$, then $I(\phi,\phi) >_{apo} I(\psi,\psi)$.*

Proof

As previously, we may assume that the rule is applied at a root position. The lemma follows from the basic remark that $I(x,\phi) >_{apo} I(t,\psi)$ (the interpretation has been chosen exactly for proving that). Moreover, for every variable y, $I(y,\phi) \geq_{apo} I(y,\psi)$. Then, replacing $I(x,\phi)$ by $I(t,\psi)$ and $I(y,\phi)$ by $I(y,\psi)$ leads to smaller terms, by the monotonicity property.

Therefore, we only have to prove $I(x,\phi) >_{apo} I(t,\psi)$ and $I(y,\phi) \geq_{apo} I(y,\psi)$ for other variables y.

The first inequality is a consequence of the three following remarks:

- every $y \in \mathcal{V}ar(t)$ is bound "above" x. More formally, $\forall y \in \mathcal{V}ar(t)$, $N(y,\phi) \leq N(x,\phi)$. Moreover, if the equality holds, then $sl(y,\phi) = sl(x,\phi)$.
- for every $y \in \mathcal{V}ar(t)$, either $N(y,\phi) < N(x,\phi)$ or $N(y,\phi) = N(x,\phi) = N(y,\psi)$ and $sl(y,\phi) > sl(y,\psi)$. In the latter case, this follows from the fact that a variable, which is bound at the same level than y, disappeared during the transformation.
- every function symbol $f \in F$ is smaller than g and h.

The second inequality holds because we do not add any quantifier along the transformation.
□

Lemma 9 *If $\phi \rightarrow_{Q_3} \psi$, then $I(\phi,\phi) \geq_{apo} I(\psi,\psi)$.*

Proof

$I(\psi,\psi)$ is obtained from $I(\phi,\phi)$ by

1. replacing $sl(z)$ by $sl(z) - 1$ for every $z \in \vec{z}$
2. removing a symbol $\exists$

In any case, the inequality given in the lemma is straightforward. $\square$

Summing up previous results, we get:

Lemma 10 *$\mathcal{R}_1 - \{(Ex)\}$ is terminating.*

We actually proved a slightly more general termination result: we proved the termination of $\mathcal{R}_1 - \{(Ex)\}$ when the rule (N_2) is not constrained by any condition.

5.3 Handling the explosion rule

The problem with the rule (Ex) is that the formula increases when it is applied. Even worse, between two consecutive explosions, the formula may have increased w.r.t. the ordering defined in the previous section; we cannot expect a decreasing property of this kind on the subsequence of formulae on which the explosion is applied.

Example 2

$$\phi \equiv \exists z:\ (\neg\exists y:\ x = f(f(y))) \wedge (\neg\exists y':\ z = f(y') \wedge (\neg\exists w:\ y' = f'(x,w)))$$

where x is a free variable. Note that $I(x,\phi) = g(a, h(a))$, $I(z,\phi) = g(h(a), h(a))$, $I(y,\phi) = I(y',\phi) = g(h^2(a), h(a))$ and $I(w,\phi) = g(h^3(a), h(a))$. No rule, but explosion, can be applied on ϕ. A "good" choice would be to explode y' (leading indeed to a smaller formula after several reduction steps). However, we want to allow the elimination of *any* quantifier first. Therefore, we allow the "bad choice" consisting in the explosion of x (in order to eliminate y). This is a bad choice, because it does not simplify the innermost quantified part of the formula. More precisely, after some reduction, we get as a subformula:

$$\psi \equiv \exists z, z':\ x = f(z') \wedge ((\neg\exists y:\ z' = f(y)) \wedge (\neg\exists y':\ z = f(y') \wedge (\neg\exists w:\ y' = f'(f(z'), w))))$$

The variable y has been popped up during this transformation (whereas some variables which are bound at deeper positions stay at the same position: that is why the system indeed terminates), but the (intuitively) heaviest part of the formula has grown. Indeed, $\neg\neg(y' = f'(f(z'), w))$ (which is embedded in the above formula) is larger than ϕ itself: because $\neg >_G \exists$ and $\neg >_G \wedge$, we have only to prove the two inequalities:

$$\left\{\begin{array}{lll} I(\neg\neg(y' = f'(f(z'), w)), \psi) & >_{apo} & I(\neg\exists y: x = f(f(y)), \phi) \\ I(\neg\neg(y' = f'(f(z'), w)), \psi) & >_{apo} & I(\neg\exists y':\ z = f(y') \wedge (\neg\exists w:\ y' = f'(x,w))) \end{array}\right.$$

i.e.

$$\left\{\begin{array}{l} \neg\neg(g(h^2(a), h(a)) = f'(f(g(h(a), h^2(a))), g(h^3(a), h(a)))) >_{apo} \\ \qquad\qquad\qquad \neg(\exists(g(a, h(a)) = f(f(g(h^2(a), h(a)))))) \\ \neg\neg(g(h^2(a), h(a)) = f'(f(g(h(a), h^2(a))), g(h^3(a), h(a)))) >_{apo} \\ \qquad\qquad\qquad \neg(\exists(g(h(a), h(a)) = f(h^2(a), h(a)) \\ \qquad\qquad\qquad \wedge(\neg(\exists(g(h^2(a), h(a)) = f'(g(a, h(a)), g(h^3(a), h(a))))))) \end{array}\right.$$

The first inequality is a consequence of the maximality of $\neg$: since there are two heading $\neg$ on the left and only one occurrence of $\neg$ on the right, the second $\neg$ allows to erase everything on the right. Similarly, for the second inequality, after erasing a $\neg$ on both sides and two symbols smaller than $\neg$ on the right side, we are left to prove the two inequalities

$$\begin{cases} \neg(g(h^2(a),h(a)) = f'(f(g(h(a),h^2(a))),g(h^3(a),h(a)))) >_{apo} g(h(a),h(a)) = f(h^2(a),h(a)) \\ \neg(g(h^2(a),h(a)) = f'(f(g(h(a),h^2(a))),g(h^3(a),h(a)))) >_{apo} \\ \qquad \neg(\exists(g(h^2(a),h(a)) = f'(g(a,h(a)),g(h^3(a),h(a))))) \end{cases}$$

Again, the first inequality follows the maximality of $\neg$ and , since $=>\exists$, the second inequality can be reduced to

$$\{g(h^2(a),h(a)), f'(f(g(h(a),h^2(a))),g(h^3(a),h(a)))\} \gg_{apo} \{g(h^2(a),h(a)), f'(g(a,h(a)),g(h^3(a),h(a)))\}$$

where $\gg_{apo}$ is the multiset extension of $>_{apo}$. Now, removing the common parts of the multisets, the inequality becomes straightforward because the right term is embedded in the left term. (Remember that $>_{apo}$ contains embedding).

We have to design another ordering which is not necessarily decreasing by application of any rule, but which is decreasing on the subsequence of formulae on which the explosion is applied.

Roughly, in order to avoid the above example, we are going to consider an interpretation in which all "non relevant" terms (such as $x, f(\vec{w})$: their variables are bound at a strictly higher level than y) are identified.

Before starting to define this new ordering, let us notice that the number of nested negations in a formula is always decreasing:

Definition 3 *The* height *of an equational formula ϕ at position p (ϕ is considered here as a non-flattened term; the definition is independent of the AC representative) is the integer*

$$H(\phi,p) = |\{q < p \mid \phi(q) = \neg\}|$$

The height *$H(\phi)$ of a formula ϕ is the maximum of $H(\phi,p)$ for every position p in ϕ.*

Lemma 11 *If $\phi \rightarrow_{\mathcal{R}_1} \psi$, then $H(\phi) \geq H(\psi)$.*

Proof

This is straightforward, because no rule does introduce negation. The duplication of a path does not change the height of the formula. $\square$

This lemma proves that, starting from ϕ, the height of the formulae cannot exceed $H(\phi)$. Let us fix the reduction sequence and call n the maximal height of the formulae in the sequence.

Now, for each $1 \leq i \leq n$, we define the interpretation functions β_i as follows. $\beta_i(\phi) = I(\psi,\phi)$ where ψ is the normal form of ϕ w.r.t. the following set of rewrite rules (ω is a new constant).

$$\begin{array}{rcll} b & \rightarrow & \omega & \text{If } b \text{ is a constant, } b \not\equiv \omega \\ f(\omega,\ldots,\omega) & \rightarrow & \omega & \text{For every } f \in F \\ x & \rightarrow & \omega & \text{If } N(x,\phi) < i \\ \omega = \omega & \rightarrow & \omega & \\ \omega \neq \omega & \rightarrow & \omega & \\ \omega \in \zeta & \rightarrow & \omega & \\ \neg\omega & \rightarrow & \omega & \\ P \wedge \omega & \rightarrow & P & \\ \omega \vee \omega & \rightarrow & \omega & \\ \exists\vec{x}:\ P & \rightarrow & P & \text{If } \vec{x} \cap \mathcal{V}ar(P) = \emptyset \\ \neg P \vee \omega & \rightarrow & \neg P & \\ (\exists\vec{x}:\ P) \vee \omega & \rightarrow & \exists\vec{x}:\ P & \text{If } \exists\vec{x}: P \text{ is irreducible} \end{array}$$

Intuitively, these rules forget irrelevant parts of a formula. The last two rules are actually "critical pairs": they have been added in order to ensure the uniqueness of normal forms.

First, we have to prove that it makes sense to speak of "normal form" of a formula w.r.t. these rewrite rule systems (call them $\mathcal{IR}_i$)[6]:

Lemma 12 *The systems $\mathcal{IR}_i$ are convergent (modulo AC).*

Proof

Termination is obvious (the size is decreasing). The system is almost ground, so there are few critical pairs to consider (including critical pairs between extended rules). They are all confluent. □

The terms in $T(G \cup \{\omega\})$ are compared with $\geq_{apo}$ extending the precedence of the previous section by inserting ω between $\perp$ and $\exists$. (Hence, ω is the smallest term).

For each i, let $\geq_i$ denote the corresponding quasi-ordering on $\mathcal{EF}'$ defined by

$$\phi \geq_i \psi \quad \text{iff} \quad \beta_i(\phi) \geq_{apo} \beta_i(\psi)$$

Let $\sim_i$ be the associated equivalence relation. ($\phi \sim_i \psi$ iff $\beta_i(\phi) \equiv \beta_i(\psi)$.)

Roughly, β_i "forgets" everything which does not contain any variable bound at a depth larger than i. This allows to identify terms which are not "relevant". For example, in the example 2, $f(z')$ and x are identified to ω by β_i, $i \geq 2$. On this example, β_3 is constant and β_2 is decreasing. (β_1 is increasing.)

Let β be the tuple $(\beta_n, \ldots, \beta_0)$. Images by β are ordered using the lexicographic combination $(\geq_n, \ldots, \geq_0)$:

$$\beta(\phi) \geq \beta(\psi) \quad \text{iff} \quad \exists i \geq 1,\ \forall j \geq i,\ \phi \sim_i \psi \text{ and } \phi >_{i-1} \psi$$

Of course, $\geq$ is well founded (on classes modulo AC) as a lexicographic composition of well founded (quasi)-orderings.

First, we prove a basic property of the interpretation β.

Lemma 13 *If t' is a subterm of t, then $t \geq_i t'$.*

Proof

We may restrict our attention to the case where t' is irreducible w.r.t. $\mathcal{IR}_i$. Moreover, it is sufficient to prove the lemma when $t \equiv C[t']_p$ with $|p| = 1$. If t is itself irreducible for the rewrite system $\mathcal{IR}_i$, then the lemma is a consequence of the subterm property of $\geq_{apo}$. It only remains to consider contexts which introduce a redex for $\mathcal{IR}_i$. In all cases, the only possiblility is to reduce t into t', thus yielding an equivalent term. □

Lemma 14 *$\geq_i$ is monotonic: assuming that the variables freely occurring in Q also occur free in P, then $P >_i Q \Rightarrow C[P] >_i C[Q]$ for every context C.*

Proof

Assume $P >_i Q$. We may assume that P and Q are irreducible w.r.t. $\mathcal{IR}_i$ (since they are equivalent modulo $\sim_i$ to their respective $\mathcal{R}_i$-normal form). We have to show that $C[P]_j >_i C[Q]_j$ for every context C and every position j. j may be assumed of length 1 without losing generality

[6] It is not possible to add $P \vee \omega \rightarrow P$ because, in such a case (N_2) would possibly increase the formula.

(inclusion of free variables is stable by context). Finally, we may assume that there is no redex for $\mathcal{IR}_i$ in the context C: the only cases that have to be investigated are redexes overlapping C and P (or Q). But, again, if $C[Q]$ is reducible and $C[P]$ is not, then the property is obvious since the normal form of $C[Q]$ is embedded in $C[Q]$ (and by properties of $\geq_{apo}$). We may therefore also assume $C[P]$ reducible at the top position (and only at that position). We look now at the three non-trivial situations:

$C[P] \equiv \exists \vec{x} : P$ In order to apply a rule of $\mathcal{IR}_i$, the condition $\vec{x} \cap \mathcal{V}ar(P) = \emptyset$ has to be satisfied. Then, because the free variables of Q are also free variables of P, $C[Q]$ also reduces to Q, which shows $C[P] >_i C[Q]$.

$C[P] \equiv P \vee \omega$ **and** $P \equiv \neg P'$ In this case, $C[P] \downarrow_i \equiv \neg P'$. Now, $\neg > \vee$ and $\neg > \omega$. Therefore, $I(\neg P, \phi) >_{apo} I(Q \vee \omega, \phi)$ whenever $I(\neg P, \phi) > I(Q, \phi)$.

$C[P] \equiv P \vee \omega$ **and** $P \equiv \exists \vec{x} : P'$ This case is similar to the above one: $\exists > \vee$ and every formula (except ω itself) is greater than ω.

□

Assume that $\phi \rightarrow_{Ex} \bigvee_{f \in F} \phi_f$ with $\phi \equiv \exists \vec{w_1} : P$ and $\phi_f \equiv \exists \vec{w_1}, \vec{w} : x = f(\vec{w}) \wedge P\{x \mapsto f(\vec{w})\}$. Because of the control on Ex, there is a variable y in P such that $N(y, \phi) > N(x, \phi)$ and an equation $x = t[y]$ (or a disequation $x \neq t[y]$) occurs in P. We say in that case that $L_\phi = N(y, \phi)$ is the *level of explosion*.

Consider moreover a sequence $\phi_f \rightarrow \ldots \rightarrow \psi_f$ where ψ_f is irreducible w.r.t. $\mathcal{R}_1 - \{(Ex)\}$. Let i be the maximal integer such that $\beta_i(\phi_f) \neq \beta_i(\psi_f)$. (Because of the control, $i \geq L_\phi$).

Lemma 15 *For every* $\phi_f \rightarrow^*_{\mathcal{R}_1 - \{(Ex)\}} \phi'$, $\phi_f \geq_i \phi'$.

Proof

First, we only have to consider a one step transformation $\phi \rightarrow_{\mathcal{R}_1 - \{(Ex)\}} \phi'$. No rule does introduce free variable. Therefore, from lemma 14, we may assume that ϕ reduces to ϕ' at the root. Now, because we already proved the decreasingness of the rules of $\mathcal{R}_1 - \{(Ex)\}$ for the interpretation I, we only have to consider here the cases where some part of ϕ is reduced to ω. Let $\phi \rightarrow_R \phi'$. We may assume, without loss of generality, that R does not introduce a redex for P_2. Indeed, if it does, then a $\neg$ has been erased and the result is obvious.

We have to investigate a lot of cases:

$R \in \{(D_1), (D_2), (D_3)\}$. The same reasoning as in the previous section works for proving the decreasingness, except when the whole left hand side can be reduced to ω. But, in such a case, the right hand side reduces to ω too, because of the rules $\omega \vee \omega \rightarrow \omega$ and $\omega \wedge P \rightarrow P$. (We see here why these rules are necessary.)

$R \in \{(C_1), (C_2), (S_3), (S_4), (S_7), (S_8), (O_1), (O_2), (Q_8), (B_1)\}$. The result is obvious.

$R \in \{(S_1), (S_2), (S_5), (S_6)\}$. The only problem comes up when t reduces to ω. But, in such a case, the whole left hand side, as well as the right hand side, reduces to ω. (Actually, ground terms always reduce to ω, for all β_i.)

$R = Q_1$. Assume that $P \sim_i \omega$. Then ϕ is interpreted as

$$\exists^{|\vec{x} \cap \mathcal{V}ar(I(Q,\phi))|}((\omega \vee I(Q \downarrow_i, \phi)) \downarrow_i).$$

ϕ' is interpreted as

$$\exists^{|\vec{x} \cap \mathcal{V}ar(Q)|}(I(Q\downarrow_i, \phi)),$$

thanks to the rule $(\exists \vec{x} : P) \vee \omega \rightarrow \exists \vec{x} : P$. In the worst case, $Q \equiv Q_1 \vee Q_2$ where $Q_1 \downarrow_i \equiv \omega$, leading to the following interpretation of ϕ:

$$\exists^{|\vec{x} \cap \mathcal{V}ar(I(Q_2, \phi))|}(I(Q\downarrow_i, \phi)).$$

In which case the two members do have the same interpretation.

$R = Q_3$. Both sides are always identical with respect to β_i

$R = Q_6$. If x is interpreted by ω, as already noticed in the previous section, t must be interpreted by ω. In such a case, the right hand side of the rule is embedded in the left hand side.

Now, if c is interpreted by ω, then the inequality is straightforward.

$R \in \{(B_2), (B_3)\}$. If $P \sim_i \omega$ or $Q \sim_i \omega$, then both sides of the rules are identical.

$R = N_2$. There are a lot of way to apply the rules of $\mathcal{IR}_i$ to the left hand side of the rule:

Q collapses to ω Then the result is straightforward: both sides reduce to the same problem.

P_1 and P_2 both reduce to ω Then the result is again straightforward.

P_1 reduces to ω and P_2 reduces to $\omega \vee P_2'$ This case actually does not occur (if it is possible, then the interpretation may increase in such a case). Indeed, P_1 and P_2 are quantifier free and $\vec{x} \cap \mathcal{V}ar(P_1) \neq \emptyset$. On the other hand, if P_1 is interpreted by ω, this means that every variable in $\vec{x}$ (and thus every variable bound above $\vec{x}$) is interpreted by ω. Therefore, every variable of P_2 is interpreted by ω, which means that P_2 itself reduces to ω.

P_2 reduces to ω and P_1 reduces to $\omega \vee P_1'$ This is again impossible. Indeed, if $P_1 \equiv P_3 \wedge (P_4 \vee P_5)$, where P_5 does not collapse to ω, then either $\mathcal{V}ar(P_5) \cap \vec{x} \neq \emptyset$ and there is a redex w.r.t. N_2 in P_1 (this is in contradiction with the control imposed on N_2) or $\mathcal{V}ar(P_3, P_4) \cap \vec{x} \neq \emptyset$. In this latter case, either P_3 or P_4 does not reduce to ω which means that P_1 cannot reduce to $\omega \vee P_1'$.

Other cases The other possibilities may stem from the rule $\neg P \vee \omega \rightarrow \neg P$. But the rules B_2 and B_3 are supposed to be applied before N_2. Therefore the negation symbol cannot occur in P_1, P_2. □

Actually, this lemma is not true if we do not impose the innermost strategy to (N_2). This is shown by the following example (which also shows, by the way, that it is not possible to add the rule $P \vee \omega \rightarrow P$ to the system $\mathcal{IR}_i$):

Example 3

$$\phi \equiv \exists x : \ ((a \neq b \wedge (x \neq t \vee a \neq b)) \vee a \neq b) \wedge x \neq u$$

where a, b, t, u are constants.

$$\beta_1(\phi) \equiv \exists((g(h(a), h(a)) \neq \omega \vee \omega) \wedge g(h(a), h(a)) \neq \omega)$$

On the other hand

$$\phi \rightarrow_{N_2} \exists x : \ ((a \neq b \wedge (x \neq t \vee a \neq b)) \wedge x \neq u) \vee (a \neq b \wedge x \neq u) \equiv \phi'$$

and

$$\beta_1(\phi') \equiv \exists(((g(h(a),h(a)) \neq \omega \vee \omega) \wedge g(h(a),h(a)) \neq \omega) \vee g(h(a),h(a)) \neq \omega)$$

which is, of course, larger than $\beta_1(\phi)$.

Such a problem only occurs when P_1 (or P_2) is not "significant" (i.e. $\sim_i \omega$).

Lemma 16 *Assume that* $\phi \rightarrow_{Ex} \bigvee_{f \in F} \phi_f$*,* $\phi_f \rightarrow^*_{\mathcal{R}_1 - \{(Ex)\}} \psi_f$ *and that every* ψ_f *is irreducible w.r.t.* $\mathcal{R}_1 - \{(Ex)\}$*. Then, for every* f*, there is an index* $i \geq L_\phi$ *such that*

- *For every* $j > i$*,* $\beta_j(\phi_f) \equiv \beta_j(\psi_f)$
- $\beta_i(\phi_f) > \beta_i(\psi_f)$*.*

Proof

Let i be the largest index such that $\beta_i(\psi_f) \not\equiv \beta_i(\phi_f)$. We know, by the control on (Ex) that such an index does exist and that it is larger than L_ϕ.

From the lemma 15, $\beta_i(\phi_f) \geq \beta_i(\psi_f)$, and since, by definition, they are distinct, we must have $\beta_i(\phi_f) > \beta_i(\psi_f)$. $\Box$

Lemma 17 *If* $\phi \rightarrow_{Ex} \bigvee_{f \in F} \phi_f \rightarrow^*_{\mathcal{R}_1 - \{(Ex)\}} \psi \rightarrow_{(Ex)} \ldots$*, then* $\beta(\phi) > \beta(\psi)$*.*

Proof

Note that $\beta_{L_\phi}(\phi) \equiv \beta_{L_\phi}(\phi_f)$ for every f, by definition of L_ϕ and β. It is not possible to transform two different parts of the problem, ϕ_f and ϕ_g by the same rule, unless one part becomes equal to $\top$ in the transformation and then the whole problem is equal to $\top$ and the measure has decreased. Then $\psi \equiv \bigvee_{f \in F} \psi_f$, with $\phi_f \rightarrow_{\mathcal{R}_1 - (Ex)} \psi_f$. Because of lemma 16, $\beta(\phi) > \beta(\psi_f)$ for every $f \in F$. We have to discuss two cases.

- If $\phi(\Lambda) \neq \vee$, then $\phi(\Lambda) > \vee$. Therefore $\psi < \phi$.
- If $\phi(\Lambda) = \vee$, then, let $\phi \equiv \bigvee_{i=1}^n \phi_i$ where, for every i, $\phi_i(\Lambda) \neq \vee$.
  - If $\psi_f \equiv \bigvee_{j=1}^m \psi_{f,j}$ and, for each j, $\psi_{f,j}(\Lambda) \neq \vee$, then the multiset of formulae $\psi_{f,j}$ is strictly smaller than the multiset of formulae ϕ_i (according to the multiset extension of $\geq$).
  - Otherwise, there exists i such that $\psi_f < \phi_i$

In every case, this shows that $\phi > \psi$. $\Box$

Lemma 18 *If* $\phi \rightarrow_{(Ex)} \rightarrow^*_{\mathcal{R}_1 - \{(Ex)\}} \psi \rightarrow_{(Ex)} \rightarrow \ldots$*, then* $\beta(\phi) > \beta(\psi)$*.*

Proof

The previous lemma shows this result when (Ex) is applied at the top position in ϕ. If we consider the rules, we can notice that no rule can modify the context C in which the sub-problem concerned by the explosion is contained, unless this sub-problem becomes equal to $\top$ or $\bot$ and the normalization changes C. But in the latter case, the part which disappears is deeper than the part which is modified. β decreases.

In both cases, the lemma is proved. $\Box$

As a straightforward corollary,

Lemma 19 $\mathcal{R}_1$ *is terminating.*

5.4 Reduction of the termination of $\mathcal{R}$ to the termination of $\mathcal{R}_1$

We are going to show that each rule of the system $\mathcal{R}$ can be simulated by a sequence of rules in $\mathcal{R}_1$.

Here is how we interpret a problem to translate the system $\mathcal{R}$ into the system $\mathcal{R}_1$:

1. we suppress each universal quantifier, thanks to the rule:

$$\forall \vec{y} P \rightarrow \neg \exists \vec{y} \neg P$$

2. then, we normalize the problem by the rules (B_i).

If ϕ is a formula, then let $\widetilde{\phi}$ be the above interpretation.

Lemma 20 *If $\phi \rightarrow_{\mathcal{R}} \phi'$, then $\widetilde{\phi} \rightarrow^+_{\mathcal{R}_1} \widetilde{\phi'}$.*

Proof

We may restrict our attention to the rules themselves and some *critical contexts* and *critical instances* corresponding to overlaps between the rules (B_i) and left hand sides of the rules of $\mathcal{R}$. We do not need to consider overlaps with right hand sides because the rules (B_i) belongs to $\mathcal{R}_1$.

There is no critical substitution nor critical context corresponding to the rule B_1. Indeed, we assume the formulae irreducible with respect to the rules P_1, P_2, which means that there cannot be any redex for B_1 in the formula ϕ where every $\forall \vec{x}$ has been replaced with a $\neg\exists\vec{x}\neg$.

Actually the only critical substitutions that have to be considered are replacements of P with $P_1 \vee P_2$ or with $P_1 \wedge P_2$ when P occurs below a $\forall\vec{y}$: this is the only case where instantiating P creates an overlapping redex for B_2, B_3. Similarly, the only critical context is $\neg$. It has to be considered each time the top symbol of the left hand side is $\vee$ or $\wedge$.

We show now, for every rule $R \in \mathcal{R}$ that $\phi \rightarrow_R \psi \Rightarrow \widetilde{\phi} \rightarrow^+_{\mathcal{R}_1} \widetilde{\phi'}$.

The rules of figure 2. First, they all belong to $\mathcal{R}_1$, so the result is obvious for the rules themselves. For none of them there is any critical substitution. Finally, there is no additional problem with the critical context $\neg$ because the set of left hand sides of the rules is closed by negation.

$Q_1, Q_3, Q_6, Q_8, Q_{10}, Ex_1$ All these rules belong to $\mathcal{R}_1$ and there is neither critical substitutions nor critical contexts. Therefore, the result of the lemma is straightforward with them.

Q_2, Q_5, Q_7, Q_9 There is no critical context nor critical substitution for these rules. On the other hand, their "negative counterpart" belongs to $\mathcal{R}_1$: the coding is straightforward.

Q_4 Encoding the rule itself using Q_3 is straightforward. We have however to consider two critical substitutions: $\{P \mapsto P_1 \wedge P_2\}$ and $\{P \mapsto P_1 \vee P_2\}$. The two cases are actually similar. Let us consider the first one:

$$\forall\vec{y} : \widetilde{P_1 \wedge P_2} \equiv \neg\exists\vec{y} : \widetilde{\neg P_1} \vee \widetilde{\neg P_2}$$

But

$$\neg\exists\vec{y} : \widetilde{\neg P_1} \vee \widetilde{\neg P_2} \rightarrow_{\mathcal{R}_1} \neg(\widetilde{\neg P_1} \vee \widetilde{\neg P_2}) \rightarrow^+_{\mathcal{R}_1} \widetilde{P_1} \wedge \widetilde{P_2}$$

which corresponds to the interpretation of the right hand side of Q_4.

N_2 This rule belongs to $\mathcal{R}_1$ and there is no critical context. Moreover, it is assumed that there is no symbol $\neg$ along the path to u. Therefore, reductions by boolean rules can only take place at positions which are uncomparable with u.

N_1 There is no critical context and, as above, no critical substitution. We have however to consider the rule itself. Let us call $\widetilde{C}$ the context C, normalized by the boolean rules. Then

$$\neg\exists\vec{x}:\ \widetilde{C}[(\widetilde{\neg P_1} \vee \widetilde{\neg P_2}) \wedge \widetilde{\neg Q}]_u$$

can be reduced by N_2 to the interpretation of the right hand side of N_1

Ex_2 The rule itself is easily encoded using Ex_1. There is no critical context. We have however to consider the two critical substitutions $\{P \mapsto P_1 \vee P_2\}$ and $\{P \mapsto P_1 \wedge P_2\}$. But, actually, the second case cannot occur because the left hand side is assumed to be irreducible with respect to (Q_2). Let us consider the first case.

$$\begin{array}{lll} \widetilde{\forall \vec{y_1} :\ P} & \equiv & \neg\exists\vec{y_1}:\ \widetilde{\neg P_1} \wedge \widetilde{\neg P_2} \\ & \to_{Ex_1} & \neg\bigvee_{f\in F}\exists\vec{y_1},\vec{y}:\ x = f(\vec{y}) \wedge \widetilde{\neg P_1}\{x \mapsto f(\vec{y})\} \wedge \widetilde{\neg P_2}\{x \mapsto f(\vec{y})\} \\ & \to^+_{\mathcal{R}_1} & \bigwedge_{f\in F}\neg\exists\vec{y_1},\vec{y}:\ x = f(\vec{y}) \wedge \widetilde{\neg P_1}\{x \mapsto f(\vec{y})\} \end{array}$$

This last problem encodes the right hand side. □

From the lemmas 19 and 20 we can conclude the proof of theorem 1.

6 Completeness Results

Inspecting the formulae which are irreducible w.r.t. $\mathcal{R}$, it is possible to restrict our attention to formulae involving at most one quantifier along each path (section 6.1). Then we show in section 6.2 how to reduce such formulae to purely existential ones. Finally, we reduce existential formulae to solved forms (section 6.3). This shows the decidability of the theory and provides complete axiomatizations of structures describing regular languages as summarized in section 7.

6.1 Formulae that are irreducible w.r.t. $\mathcal{R}$

Lemma 21 *If $C[\phi]$ is irreducible by $\mathcal{R}$, then ϕ is also irreducible by $\mathcal{R}$.*

Proof

Indeed, the only contextual conditions in the control part of the rules of $\mathcal{R}$ are irreducibility conditions; the only situations in which the control on the rules may prevent the reduction $C[\phi] \to_{\mathcal{R}} C[\psi]$ when $\phi \to_{\mathcal{R}} \psi$ are those where there is some redex in C. □

Let us call a *simple formula* an equational formula in which there is no path containing two distinct quantifiers.

Lemma 22 *Irreducible formulae (w.r.t. $\mathcal{R}$) are simple.*

Proof

Assume that ϕ is irreducible for $\mathcal{R}$ and that there is a subformula ψ of ϕ such that

$$\psi \equiv \exists\vec{x}:\ P[\forall\vec{y}:\ Q]_q$$

where Q is quantifier free. In such a case, we may assume, without loss of generality, that there is no existential quantifier along the path q. We want to derive a contradiction.

Q must be a disjunction or an atomic formula, otherwise (Q_2) would be applicable: $Q \equiv A_1 \vee \ldots \vee A_n$. Moreover, each A_i is an atomic formula or else $\mathcal{V}ar(A_i) \cap \vec{y} = \emptyset$. Indeed, if these conditions are not met, then (N_1) can be applied. This shows that every occurrence of $y \in \vec{y}$ must be in an atomic formula A_i.

On the other hand, ψ is not reducible by (Ex_1). Therefore, every occurrence of $y \in \vec{y}$ must be in one of the following kinds of atom:

- $t[y] \in \zeta$ where $\zeta \in \mathcal{E}(\mathcal{S})$.
- $t[y] = u$ where u is not a variable or $u \in \vec{y}$
- $t[y] \neq u$ where u is not a variable or $u \in \vec{y}$

We exclude now some cases:

- y occurs in $t[y] \in \zeta$ where t is not y itself. Then (D_3) applies.
- y occurs in $t[y] = u$ (resp. $t[y] \neq u$) and neither t nor u is a variable. Then (D_1) or (C_1) (resp. (D_2) or (C_2)) applies.
- y occurs in $t[y] = y$ (resp. $t[y] \neq y$). Then (O_1) (resp. (O_2)) applies.
- y occurs in $y \neq t$ where $y \notin \mathcal{V}ar(t)$. Then, (Q_5) applies on $\forall \vec{y}: A_1 \vee \ldots \vee A_n$ because one of the A_i's is $y \neq t$.

Hence, each $y \in \vec{y}$ only occurs in atomic formulae of one of the following forms:

- $y \in \zeta$
- $y = t$
- $y' = t[y]$ where $y' \in \vec{y}$ and $y' \not\equiv y$

If we summarize the above facts, $\forall \vec{y}: Q$ has the form:

$$\forall \vec{y}: \; y_1 = t_1 \vee \ldots \vee y_n = t_n \vee y'_1 \in \zeta_1 \ldots \vee y'_m \in \zeta_m \vee d$$

where $\mathcal{V}ar(d) \cap \vec{y} = \emptyset$ and $y_1, \ldots, y_n, y'_1, \ldots, y'_m \in \vec{y}$. Therefore, either one of the rules (S_i) can be applied or else (Q_9) or (Q_7) can be applied. This leads to a contradiction.

The case where the two nested quantifiers are in reverse order is similar. $\square$

6.2 Reduction of simple formulae to existential formulae

A formula is *purely existential* if it does not contain any universal quantifier. The next step towards solved forms, is to transform simple formulae into purely existential formulae. Actually, a simple formula is purely existential if it starts with an existential quantification. Thus, we only have to use a simple trick: add a (dummy) existential quantifier at the top of the formula, in such a way that no further transformation will remove it from this top position. Then reduce the formula to its simple form and remove the dummy part. Note that this trick could be used at once as well: we may assume that purely simple formulae are purely existential. Now let us show the technique in more details.

1. add a dummy existential quantifier at the top of the formula; if $x, z \notin \mathcal{V}ar(\phi)$ and $f \in F$ is a non-constant function symbol, let $\phi' \equiv \exists z: \; x = f(z, \ldots, z) \wedge \phi$.

2. normalize ψ w.r.t. $\mathcal{R}$. The existential quantification $\exists z$ cannot disappear, except if the whole formula collapses to $\perp$ of $\top$. Therefore, the result of the $\mathcal{R}$-normalization is a formula
$$\bigvee_{i \in I} \exists z, \exists \vec{w_i} : \ x = f(z, \ldots, z) \wedge P_i$$
where P_i is quantifier-free. (The result should be a simple formula and there is an existential quantifier on each path.)

3. The last step consists in removing the dummy part of the formula: we get
$$\bigvee_{i \in I} \exists \vec{w_i} : \ P_i$$

Of course this sequence of transformations terminates. Moreover, the set of restrictions to $X - \{x\}$ of the solutions is kept along the transformation. This proves the correctness of this transformation.

6.3 Reduction of existential formulae to solved forms

The last step is to transform purely existential formulae into a solved form. For that, we assume now that the rule (Q_1) is applied with the highest priority, which means that formulae are kept in the form
$$\bigvee_{i \in I} \exists \vec{x_i} P_i$$
where P_i is a quantifier-free conjunction.

Then we use the rules of figures 2 and 6. This defines a transformation system $\mathcal{R}_\exists$.

Note that we use both merge and replacement rules in this system. Of course, this is redundant since (M) is a particular instance of (R_1). But, since we prove the termination with both rules, it is possible to remove e.g. merge rules and keep the termination property; our result leaves more freedom in choosing the strategy.

Theorem 2 *$\mathcal{R}_\exists$ terminates.*

Proof

We interpret formulae as follows: $I_0(\bigvee_{i \in I} \exists \vec{x_i} : \ P_i)$ is the multiset $\{I_1(\exists \vec{x_i} : \ P_i)\}$ for $i \in I$, where the interpretation function I_1 is the lexicographic composition of the following interpretations:

- $n(\exists x_i : \ P_i)$ which is the natural number $|\vec{x_i}| + |U(P_i)|$ where $U(P_i)$ is the number of free variables of P_i which are not *solved*. A variable x is solved in a conjunction P if $P \equiv x = t \wedge P'$ and x occurs only once in P.
- $I_2(\exists \vec{x_i} : \ P_i)$, which is a term in $T(G)$ defined as follows. $G = \{\in, \vee, \wedge\} \cup \{a_n \mid n \in \mathbf{N}\}$ and
  - $I_2(\exists \vec{x_i} : \ P_1) \equiv I_2(P_i)$,
  - each equation $s \neq t$ and each inequation $s \neq t$ in P_i is interpreted as $a_{\max(|s|,|t|)}$
  - each membership constraint $s \in \zeta$ is interpreted as $\in(a_{|s|})$.
  - $I_2(Q_1 \wedge Q_2) = I_2(Q_1) \wedge I_2(Q_2)$ and $I_2(Q_1 \vee Q_2) = I_2(Q_1) \vee I_2(Q_2)$.
- $M(\vec{x_i} : \ P_i)$ is the number of redexes in P_i for the rule (M).

$$(R_1)\quad \exists\vec{w}:\ x = t \wedge P \ \rightarrow\ \exists\vec{w}:\ x = t \wedge P\{x \mapsto t\}$$

If

- x is free, $x \notin \mathcal{V}ar(t)$ and $x \in \mathcal{V}ar(P)$
- either t is not a variable or $t \in \mathcal{V}ar(P)$

$$(Q_1)\quad \exists\vec{x}:\ P \vee Q \ \rightarrow\ (\exists\vec{x}:\ P) \vee (\exists\vec{x}:\ Q)$$

This rule is assumed to be used before any other rule. It used in order to avoid the introduction of redexes for (P_2).

$$(Q_6)\quad \exists w:\ w = t \wedge c \ \rightarrow\ c\{w \mapsto t\}$$

If $w \notin \mathcal{V}ar(t)$.

$$(M)\quad x = t \wedge x = u \ \rightarrow\ x = t \wedge t = u$$

If x is a variable, t is not a variable and $|t| \leq |u|$.

$$(N_3)\quad (P \vee Q) \wedge R \ \rightarrow\ (P \wedge R) \vee (Q \wedge R)$$

Figure 6: Replacement and Merging rules

The ordering on $T(G)$ is the associative path ordering extending the following precedence on G:

$$\in > \ldots > a_n > a_{n-1} > \ldots > a_0 > \wedge > \vee$$

G is infinite, but, nevertheless, the ordering is well founded (instead of an infinite set of constants, we could use a unary function symbol and a constant).

For other components, we use the classical ordering on natural numbers and the classical lexicographic and multiset extensions of orderings. Of course, the quasi-ordering defined by

$$\phi > \psi \text{ iff } I_0(\phi) > I_0(\psi)$$

is well founded. It remains only to show that $> \supseteq \rightarrow$.

- If $\phi \rightarrow \psi$ using a decomposition rule, then the decreasingness property stems from the fact that each atomic formula on the right hand side involves (one or) two terms that are both strictly smaller (in size) than the original maximal term on the left.
- If $\phi \rightarrow \psi$ using a rule (S_i), then the decreasingness property is a consequence of the leadership of $\in$ in the precedence.
- If $\phi \rightarrow_{R_1} \psi$, then $n(\phi) > n(\psi)$ because x was free and unsolved in ϕ whereas, x is solved in ψ. On the other hand, t cannot be a solved variable of ϕ, hence every variable which is solved in ϕ is also solved in ψ.
- If $\phi \rightarrow_{Q_6} \psi$, then $n(\phi) > n(\psi)$ because the number of existentially bound variables is smaller on the right than on the left.
- (M) cannot increase the number of unsolved variables because we assume that t is not a variable. (M) does not change the interpretation I_2 because we impose $|t| \leq |u|$. On the other hand, of course, the interpretation function M decreases because x is a variable.
- Decreasingness when (N_3) is applied is straightforward $\square$

We finally state the completeness w.r.t. solved forms:

Theorem 3 *Every purely existential formula which is irreducible w.r.t. $\mathcal{R}_\exists$ is a solved form.*

Proof

By (N_3), we know that an irreducible formula must be a disjunction of formulae

$$\exists \vec{x} : a_1 \wedge \ldots \wedge a_n$$

where $a_1, \ldots, a_n$ are atomic formulae. By $(D_i), i = 1, 2, 3$ and $(C_i), i = 1, 2$, we know that each irreducible atomic formula a_i must be either $x = t$ or $x \neq t$ or $x \in \zeta$ where x is a variable. This means that each irreducible formula is a disjunction of formulae

$$\exists \vec{w} : x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_k \neq u_k \wedge y_1 \in \zeta_1 \wedge \ldots \wedge y_m \in \zeta_m$$

where $x_1, \ldots, x_n, y_1, \ldots, y_k, z_1, \ldots, z_m$ are variables. Moreover,

- $x_1, \ldots, x_n$ must be free in the formula (otherwise, it is possible to apply (Q_6))
- $x_1, \ldots, x_n$ occur only once. Indeed, for each i, either t_i is not a free variable and then x_i occurs only once by irreducibility w.r.t. $(R_1), (O_2)$, or else t_i and x_i are both free variables. Then, by irreducibility w.r.t. (R_1) either x_i or t_i occurs only once. By symmetry, x_i can be chosen to be this variable.

- For all i, $z_i \notin \mathcal{V}ar(u_i)$, otherwise, it is possible to apply (O_2).
- $y_1, \ldots, y_m$ are distinct, otherwise it is possible to apply (S_1).
- for all i, $[\![\zeta_i]\!]$ is infinite, otherwise, it is possible to apply (S_5).

Now, all conditions for solved forms are met. $\square$

7 Summary of the method and its consequence

7.1 From equational formulae to solved forms

Summing up everything, we get a decision procedure for the validity of equational formulae in the interpretatioin we consider:

1. reduce the formula, using the rewrite system $\mathcal{R}$ described in figures 2, 3, 4. At each step, the formulae are kept in a reduced form, i.e. they are normalized using the rules of figure 1.
2. if necessary, add the dummy heading, as described in section 6.2, normalize again by $\mathcal{R}$ and remove the dummy part.
3. normalize by $\mathcal{R}_\exists$

As a consequence of termination and completeness results of sections 5 and 6, we get:

Theorem 4 *If $[\![\cdot]\!]$ is admissible, the above procedure terminates on every equational formula, yielding an equivalent (w.r.t. $[\![\cdot]\!]$) finite disjunction of solved forms.*

7.2 Complete axiomatizations of rational languages of trees

From theorem 4 and lemma 1, we can conclude:

Corollary 2 *Given an arbitrary equational formula ϕ and an automaton $\mathcal{A}$, it is possible to decide whether ϕ has a solution w.r.t $\mathcal{A}$ or not.*

In other words, let us consider the first order language $\mathcal{L}$ over the alphabet F of function symbols and the alphabet $\{=\} \cup \{\in \zeta \mid \zeta \in \mathcal{E}(\mathcal{S})\}$ of predicate symbols. $\mathcal{L}$, together with an automaton $\mathcal{A}$ defines a first order structure $\mathcal{L}(\mathcal{A})$, which is confused with the language recognized by $\mathcal{A}$, which is again confused with $\mathcal{A}$ itself. The above corollary shows that this first order structure is decidable. However, it is even possible to derive a stronger result, because our set of transformation rules actually provides an axiomatization of $\mathcal{L}(\mathcal{A})$. Let $\mathcal{T}(\mathcal{A})$ be the set of formulae given in figure 7, then

Theorem 5 *$\mathcal{T}(\mathcal{A})$ is a complete (recursive) axiomatization of $\mathcal{L}(\mathcal{A})$.*

Proof

It is sufficient to prove that all rules that are used to decide the validity in $\mathcal{L}(\mathcal{A})$ are logical consequences of axioms in $\mathcal{T}(\mathcal{A})$. This is straightforward for all rules. Let us show it for (Q_7) and (Q_8) (which are easily deducible one from each other). These rules are also easy to deduce from lemma 2. If $P \equiv s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n \wedge x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m$ satisfies the hypotheses of lemma 2, then there are ground terms $t_1, \ldots, t_m$ such that, for all i, $t_i \in [\![\zeta_i]\!]_\mathcal{A}$ and, for all j,

Equality Axioms

$$\begin{array}{ll}
\forall x & x = x \\
\forall x, y & x = y \rightarrow y = x \\
\forall x, y, z & x = y \wedge y = z \rightarrow x = z \\
\forall x_1, \ldots, x_n, y_1, \ldots, y_n & x_1 = y_1 \wedge \ldots \wedge x_n = y_n \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)
\end{array}$$

for $n \in \mathbf{N}$ and $f \in F$ with arity n.

Finite trees axioms

$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n : \; f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \quad \rightarrow \quad x_1 = y_1 \wedge \ldots \wedge x_n = y_n$$

for $n \in \mathbf{N}$ and $f \in F$ with arity n.

$$\forall x_1, \ldots x_n, y_1, \ldots, y_m \quad f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m)$$

for $n, m \in \mathbf{N}$, $f, g \in F$ of arities respectively n and m and $f \neq g$

$$\forall x \quad x \neq t[x]_p$$

for all terms t and positions $p \in Pos(t) - \{\Lambda\}$.

$$\forall x, \bigvee_{f \in F} \exists \vec{z} : \; x = f(\vec{z})$$

Membership axioms

$$\begin{array}{lrcl}
\forall x : & x \in \zeta \wedge x \in \zeta' & \leftrightarrow & x \in \zeta \wedge \zeta' \\
\forall x : & x \in \zeta \vee x \in \zeta' & \leftrightarrow & x \in \zeta \vee \zeta' \\
\forall x : & \neg x \in \zeta & \leftrightarrow & x \in \neg \zeta
\end{array}$$

For all sort expressions ζ and ζ'.

$$\forall x : \quad x \in \zeta \quad \leftrightarrow x = t_1 \vee \ldots \vee x = t_n$$

for every sort expression ζ such that $[\![\zeta]\!]_{\mathcal{A}} = \{t_1, \ldots, t_n\}$. (When $n = 0$, the right hand side should be understood as $\perp$).

$$a \in \zeta$$

for every sort expression ζ and every constant $a \in [\![\zeta]\!]_{\mathcal{A}}$.

$$\forall t_1, \ldots, t_n : \quad f(t_1, \ldots, t_n) \in \zeta \quad \leftrightarrow \bigvee_{(\zeta_1, \ldots, \zeta_n) \in f^{-1}(\zeta)} t_1 \in \zeta_1 \wedge \ldots \wedge t_n \in \zeta_n$$

for every sort expression ζ and every non-constant $f \in F$.

Figure 7: An axiomatization of the tree language recognized by $\mathcal{A}$

$s_j\{x_1 \mapsto t_1; \ldots, x_m \mapsto t_m\} \not\equiv t_j\{x_1 \mapsto t_1; \ldots; x_m \mapsto t_m\}$. Now, from the two last membership axiom schemas, it is possible to deduce $t_i \in \zeta_i$ for every i. And from the finite trees axioms (decompositions and clashes), it is possible to deduce

$$x_1 = t_1 \wedge \ldots \wedge x_m = t_m \;\Rightarrow\; s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n.$$

Hence,

$$\mathcal{T}(\mathcal{A}) \;\models\; \exists x_1, \ldots x_m : s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n \wedge x_1 \in \zeta_1 \wedge \ldots \wedge x_m \in \zeta_m$$

whenever $s_i \not\equiv t_i$ for all i and $[\![\zeta_j]\!]$ is infinite for all j. The correctness of rules $(Q_7), (Q_8)$ follows.
□

Example 4 Assume that $\mathcal{A}$ defines the inclusion $\underline{s}_1 \leq \underline{s}_2$. Then we are going to show that

$$\mathcal{T}(\mathcal{A}) \;\models\; \forall x. x \in \underline{s}_1 \Rightarrow x \in \underline{s}_2$$

Actually, the formula is an abbreviation for $\forall x. \neg(x \in \underline{s}_1) \vee x \in \underline{s}_2$. By membership axioms,

$$\mathcal{T}(\mathcal{A}) \models (\forall x. \neg(x \in \underline{s}_1) \vee x \in \underline{s}_2) \leftrightarrow (\forall x. x \in (\neg \underline{s}_1 \vee \underline{s}_2))$$

Since, by hypothesis, $[\![\neg(\neg \underline{s}_1 \vee \underline{s}_2)]\!]_{\mathcal{A}}$ is empty, by the fourth membership axiom schema, we get

$$\mathcal{T}(\mathcal{A}) \;\models\; \forall x. x \in \neg(\neg \underline{s}_1 \vee \underline{s}_2) \leftrightarrow \bot$$

which can be reformulated as

$$\mathcal{T}(\mathcal{A}) \;\models\; \forall x. x \in (\neg \underline{s}_1 \vee \underline{s}_2)$$

using the third membership axiom.

8 Extension to a class of non-regular tree languages

All the job we described in previous sections assumes that the semantic mapping $[\![\cdot]\!]$ is admissible. This condition can be relaxed to some extent. It is possible actually to weaken the decomposability condition. Of course, this leads to another decomposition rule. But, taking care of the decreasingness of formulae (w.r.t. the ordering defined in section 5), all the results of previous sections can be extended to a more general class of interpretations. This new class is a significant extension of the former one, since it contains the $\mathrm{REC}_=$ class of tree languages described by Bogaert and Tison (1992).

8.1 Weakening the decomposability requirement

Definition 1 is now replaced with the following:

Definition 4 *A semantic mapping $[\![\cdot]\!]$ from $\mathcal{E}(S)$ to $2^{T(F)}$ is* E-decomposable *if there is a (effectively computable) mapping which associates to each function symbol $f \in F$, of arity $n \geq 1$ and each sort expression ζ a finite set $f^{-1}(\zeta)$ of triples (w, A, B) where $w \in \mathcal{E}(S)^n$, and $A, B \subseteq \{1, \ldots, n\}^2$ in such a way that*

$$f(t_1, \ldots, t_n) \in [\![\zeta]\!] \quad \textit{iff}$$
$$\exists\, ((\zeta_1, \ldots \zeta_n), A, B) \in f^{-1}(\zeta),\ t_1 \in [\![\zeta_1]\!], \ldots, t_n \in [\![\zeta_n]\!] \wedge \bigwedge_{(i,j) \in A} t_i \equiv t_j \wedge \bigwedge_{(i,j) \in B} t_i \not\equiv t_j$$

$[\![\cdot]\!]$ *is* E-admissible *if it is E-decomposable and if there is a terminating algorithm which decides, for every sort expression ζ, whether $[\![\zeta]\!]$ is finite or not and, in case of finiteness, effectively computes $[\![\zeta]\!]$.*

Assuming that $[\![\cdot]\!]$ is E-admissible, everything works as described in previous sections, except that we have to replace the rule (D_3) with the following:

$$(D_3') \quad f(t_1,\ldots,t_n) \in \zeta \quad \rightarrow \bigvee_{((\zeta_1,\ldots,\zeta_n),A,B)\in f^{-1}(\zeta)} (t_1 \in \zeta_1 \wedge \ldots \wedge t_n \in \zeta_n \wedge (\bigwedge_{(i,j)\in A} t_i = t_j) \wedge (\bigwedge_{(i,j)\in B} t_i \neq t_j))$$

Replacing (D_3) with (D_3'), we have to check that the rules are still decreasing. This holds because we always choose $\in > \wedge$, $\in > \vee$ and $\in >=$ in the precedences: adding any number of equalities and disequalities between subterms in the right hand side of the rule does not change the way the rule is ordered. Hence, we get the following result:

Theorem 6 *If $[\![\cdot]\!]$ is E-admissible, then there is a terminating set of rules which transform every equational formula into an equivalent finite disjunction of solved forms.*

Now, we have to show examples of E-admissible (and not necessary admissible) interpretations.

8.2 Tree automata with equality tests

We recall here definitions and some results on tree automata with equality tests, as described by Bogaert (1990) and Bogaert and Tison (1992).

Definition 5 (Bogaert 1990)
An automaton with equality tests *is a tuple $(Q, Q_f, F, \mathcal{P})$ where F is a finite alphabet of function symbols together with their arity, Q is a finite set of* state symbols, *Q_f is a subset of Q called* final states *and $\mathcal{P}$ is a finite set of rewrite rules of the form:*

$$f(q_1(x_1),\ldots,q_n(x_n)) \rightarrow q(f(x_1,\ldots,x_n)) \mathit{If}\ c$$

where $f \in F$ has arity n, $q_1,\ldots,q_n,q \in Q$, $x_1,\ldots,x_n$ are variables and c is a boolean combination of equations $x_i = x_j$ where $1 \leq i,j \leq n$.

Note that this definition corresponds to a *bottom-up* recognizer. It is possible to assume (or not) that c contains negations, without changing the basic results, but, if negation is not allowed, we have to use a different definition of moves than the one below (see (Bogaert 1990, Bogaert and Tison 1992): we would need "strongly controlled strategies").

Definition 6 (Bogaert 1990)
Given an automaton with equality tests $\mathcal{A}$, there is a move of $\mathcal{A}$ from $t \in T(F \cup Q)$ to $t' \in T(F \cup Q)$ (which is denoted $t \underset{\mathcal{A}}{\longrightarrow} t'$) if there is a rule r : $[f(q_1(x_1),\ldots,q_n(x_n)) \rightarrow q(f(x_1,\ldots,x_n))\ \ \mathit{If}\ c]$ in $\mathcal{A}$, a position $p \in Pos(t)$ and an assignment σ from X into $T(F)$ such that

- *$t \xrightarrow[r\sigma]{p} t'$ (with the usual definition of rewriting, see e.g. (Dershowitz and Jouannaud 1990))*
- *σ is a solution of c*

The language generated by such an automaton $\mathcal{A}$ is the set of terms $t \in T(F)$ such that $t \xrightarrow[\mathcal{A}]{*} q(t)$ where $q \in Q_f$.

The following results are proved by Bogaert (1990):

Theorem 7 (Bogaert 1990)
The class of tree languages that are recognized by a tree automaton with equality tests (which is called $REC_=$) is closed under finite intersection, finite union, strict alphabetic morphism and negation.

Each language recognized by an automaton with equality tests is also recognizable by a deterministic automaton with equality tests.

Emptiness and finiteness of the language recognized by an automaton with equality tests are decidable. Moreover, in case of finiteness, there is a terminating algorithm which effectively computes the language.

For convenience, we will only consider deterministic automata in which the conditions c appearing in the rules are conjunctions of equations and disequations. (This is possible without loosing any generality.) With such assumptions, we identify a condition c with a set of equations and disequations.

8.3 E-admissibility of interpretations in the class REC$_=$

Lemma 23 *Let $\mathcal{A}$ be an automaton with equality tests whose set of states contains S. Then the mapping $[\![\cdot]\!]_{\mathcal{A}}$ which associates to each state $\underline{s} \in S$ the language recognized by $\mathcal{A}$ in state $\underline{s}$ is E-admissible.*

Proof
Because of theorem 7, we only have to prove E-decomposability of $[\![\cdot]\!]_{\mathcal{A}}$.

The proof is similar to lemma 1. We give two examples of the construction of f^{-1} below.

- if $\underline{s} \in S$, then

$$f^{-1}(\underline{s}) = \{((\underline{s}_1, \ldots, \underline{s}_n), A, B) \mid [f(\underline{s}_1(x_1), \ldots, \underline{s}_n(x_n)) \to \underline{s}(f(x_1, \ldots, x_n)) \text{ If } c] \in P, \\ A = \{(i,j) \mid x_i = x_j \in c\}, B = \{(i,j) \mid x_i \neq x_j \in c\}\}$$

-

$$f^{-1}(\zeta \wedge \zeta') = \{(((\zeta_1 \wedge \zeta'_1, \ldots, \zeta_n \wedge \zeta'_n), A \cup A', B \cup B') \mid ((\zeta_1, \ldots, \zeta_n), A, B) \in f^{-1}(\zeta), \\ ((\zeta'_1, \ldots, \zeta'_n), A', B') \in f^{-1}(\zeta')\}$$

It is a routine work to define f^{-1} for all constructions and to check the lemma in each case. □

Now, as a consequence of theorem 6 and the above lemma, we get:

Theorem 8 *Given any automata with equality tests $\mathcal{A}$, there is a terminating procedure which transforms any equational formula into an equivalent (w.r.t. $[\![\cdot]\!]_{\mathcal{A}}$) disjunction of solved forms. In particular, the validity in the structure defined by $\mathcal{A}$ of any sentence is decidable: the first order theory of the structure defined by $\mathcal{A}$ is decidable.*

9 Application to inductive proofs

The problem of automatizing inductive proofs in equational theories (and sufficient completeness of algebraic specifications, which is a related problem) has been studied by many authors. (See e.g. Jouannaud and Kounalis (1986)). It appears that the crucial problem is to decide the so-called *inductive reducibility*. Given a term rewriting system $\mathcal{R}$, a term t is inductively reducible iff all ground instances of t are reducible by $\mathcal{R}$. Let us take a non-trivial example:

$$\mathcal{R} = \{ \quad \begin{array}{rcl} p(s(x)) & \rightarrow & x \\ x+x & \rightarrow & 0 \\ p(x)+y & \rightarrow & p(x+y) \end{array} \quad \begin{array}{rcl} s(p(x)) & \rightarrow & x \\ 0+x & \rightarrow & x \\ x+p(y) & \rightarrow & p(x+y) \end{array} \quad \begin{array}{rcl} s(s(0)) & \rightarrow & 0 \\ x+0 & \rightarrow & x \\ & \} & \end{array}$$

Is $x+y$ inductively reducible? [7]

In this example, the language of trees in $T(\{0,s,p\})$ that are irreducible by $\mathcal{R}$ is regular[8] (but the language of all reducible ground terms may be not regular because of the non linear left hand side $x+x$). Let us call it NF. Then, $x+y$ is inductively reducible iff, for every terms $t, t' \in NF$, $t+t'$ can be matched against a left hand side of a rule in $\mathcal{R}$. In other words, $x+y$ is inductively reducible iff the following formula reduces to $\top$:

$$\begin{array}{rl} \phi \equiv \forall x, y;\ x \notin NF \vee y \notin NF \vee (\ \exists x_1, x_2; & x_1 + x_1 = x + y \vee x_1 + 0 = x + y \\ & \vee 0 + x_1 = x + y \vee x_1 + p(x_2) = x + y \vee p(x_1) + x_2 = x + y) \end{array}$$

This can be done using the rules of the previous sections and, as an example of transformation, we are going to show it. But let us first emphasize that, although algorithms for deciding inductive reducibility are known (Plaisted 1985,Kapur, Narendran, and Zhang 1987,Comon 1988), they are all very complex. They are generally based on "test sets" : they check the reducibility of finitely many ground instances of t instead of checking it for all ground instances of t. All ground instances of t that have to be considered form the test set. In order to give an idea of the complexity, consider the example above. For checking the inductive reducibility of $x+y$, the test set has $2^{4\times 9^{17^{17}}+2}$ elements (from Kapur, Narendran, and Zhang (1987)). This bound has been slightly improved, reducing the size of the test set from one exponential factor. Anyway, the technique we propose seems more efficient. [9]

Now we show the transformation of ϕ:

$$\begin{array}{lll} \phi & \rightarrow^*_{D_1} & \forall x, y:\ x \in \neg NF \vee y \in \neg NF \vee (\exists x_1, x_2:\ (x_1 = x \wedge x_1 = y) \vee (x_1 = x \wedge y = 0) \\ & & \qquad \vee (0 = x \wedge y = x_1) \vee (x_1 = x \wedge p(x_2) = y) \vee (p(x_1) = x \wedge x_2 = y)) \\ & \rightarrow^*_{Q_i} & \forall x, y:\ x \in \neg NF \vee y \in \neg NF \vee x = y \vee y = 0 \vee x = 0 \\ & & \qquad \vee (\exists x_2:\ y = p(x_2)) \vee (\exists x_1:\ p(x_1) = x) \\ & \rightarrow_{Ex_2} & [\forall x, y:\ x \neq 0 \vee 0 = 0 \vee \ldots] \\ & & \wedge [\forall x, y, x':\ x \neq s(x') \vee s(x') \in \neg NF \vee y \in \neg NF \vee s(x') = y \\ & & \qquad \vee s(x') = 0 \vee y = 0 \vee (\exists x_1:\ s(x') = p(x_1)) \vee (\exists x_2:\ y = p(x_2))] \\ & & \wedge [\forall x, y, x':\ x \neq p(x') \vee p(x') \in \neg NF \vee y \in \neg NF \vee p(x') = y \\ & & \qquad \vee p(x') = 0 \vee y = 0 \vee (\exists x_1:\ p(x') = p(x_1)) \vee (\exists x_2:\ y = p(x_2))] \\ & & \wedge [\forall x, y, x'_1, x'_2:\ x \neq x'_1 + x'_2 \vee x'_1 + x'_2 \in \neg NF \vee y \in \neg NF \vee y = x'_1 + x'_2 \\ & & \qquad \vee x'_1 + x'_2 = 0 \vee (\exists x_2:\ y = p(x_2)) \vee (\exists x_1:\ x'_1 + x'_2 = p(x_1))] \end{array}$$

[7] This problem is equivalent (under some conditions that hold in this example) to the "sufficient completeness" of the specification of $+$ w.r.t. these rules.

[8] Here, this is quite clear since every left hand side of a rule either involves $+$ or is linear (i.e. every variable occurs at most once in it). But it may happen that a language of irreducible ground terms is regular although the rules are not left-linear. Actually, we have an algorithm which, we conjecture, decides the regularity of the language of irreducible ground terms. (See e.g. Comon (1989)).

[9] Actually, there is a lower bound for the complexity of inductive reducibility: it is at least $O(2^{cn/\log(n)})$ where c is a constant and n is the size of the rewrite system, assuming that the term t has depth 1 (Kapur, Narendran, Rosenkratz, et al. 1991). On the other hand, algorithms derived from our set of rules are at least exponential, because some determinization of finite automata may be necessary.

The first part is actually $\top$ because of the equation $0 = 0$, the third part is reduced to $\top$ using D_1 and Q_6. Thus, we can write the last formula as a conjunction of two formulae $\phi_1 \wedge \phi_2$ (we will use these notations in further reductions).

Now, let us precise the automaton which recognizes $\neg NF$ (i.e. the terms that either contain an occurrence of $+$ or are reducible by a rule): q_f is the final state and the transitions are

$s(s(0))$ is reducible :	$0 \rightarrow q_0$	$s(q_0) \rightarrow q_1$	$s(q_1) \rightarrow q_f$
for any term x, $p(s(x))$ is reducible:	$s(q_2) \rightarrow q_3$	$p(q_3) \rightarrow q_f$	
For any x, $s(p(x))$ is reducible:	$p(q_2) \rightarrow q_4$	$s(q_4) \rightarrow q_f$	
q_2 contains all terms	$q_0 \rightarrow q_2$	$q_1 \rightarrow q_2$	$q_f \rightarrow q_2$
	$q_3 \rightarrow q_2$	$q_4 \rightarrow q_2$	
$x + y$ contains an occurrence of $+$:	$q_2 + q_2 \rightarrow q_f$		
If $t\|_p$ is reducible, then so is t:	$s(q_f) \rightarrow q_f$	$p(q_f) \rightarrow q_f$	
No redex and no $+$ have been found yet	$p(q_2) \rightarrow q_2$	$s(q_2) \rightarrow q_2$	

$[\![q_0]\!] = \{0\}$, $[\![q_1]\!] = \{s(0)\}$, $[\![q_3]\!] = \{s(x) \mid x \in T(F)\}$, $[\![q_4]\!] = \{p(x) \mid x \in T(F)\}$. These states correspond to the recognition of a subterm of one of the left hand sides $p(s(x)), s(p(x)), s(s(0))$. $[\![q_2]\!] = T(F)$ and $[\![q_f]\!] = \{t \in T(F) \mid \exists p \in Pos(t), \exists t_1, t_2 \in T(F), t|_p \equiv t_1 + t_2$ or $t|_p \equiv s(s(0))$ or $t|_p \equiv s(p(t_1))$ or $t|_p \equiv p(s(t_1))\}$.

Coming back to our formula, ϕ_2 is reduced to $\top$ by D_3 and ϕ_1 is reduced using C_1, Q_4, Q_5 and D_3; $\phi \rightarrow^* \phi_1$ and

$$\begin{array}{lll}
\phi_1 & \rightarrow^*_{C_1} & \forall x, y, x' : \ x \neq s(x') \vee s(x') \in \neg NF \vee y \in \neg NF \vee y = s(x') \vee y = 0 \\
& & \qquad \vee(\exists x_2 : \ y = p(x_2)) \\
& \rightarrow^*_{Q_4,Q_5} & \forall y, x' : \ s(x') \in \neg NF \vee y \in \neg NF \vee y = s(x') \vee y = 0 \vee (\exists x_2 : \ y = p(x_2)) \\
& \rightarrow_{D_3} & \forall y, x' : \ x' \in \neg NF \vee x' \in q_1 \vee x' \in q_4 \vee y \in \neg NF \vee y = s(x') \\
& & \qquad \vee y = 0 \vee (\exists x_2 : \ y = p(x_2))
\end{array}$$

since $s^-1(q_f) = \{q_1, q_4, q_f\}$. Now, using S_2 and Ex_2 on y:

$$\begin{array}{lll}
\phi & \rightarrow^* & \forall y, x' : \ x' \in (\neg NF \vee q_1 \vee q_4) \vee y \in \neg NF \vee y = s(x') \vee y = 0 \vee (\exists x_2 : \ y = p(x_2)) \\
& \rightarrow_{Ex_2} & \forall y, x' : \ y \neq 0 \vee x' \in (\neg NF \vee q_1 \vee q_4) \vee 0 \in \neg NF \vee 0 = s(x') \\
& & \qquad \vee 0 = 0 \vee (\exists x_2 : \ 0 = p(x_2)) \\
& & \wedge \forall y, x', y' : \ y \neq s(y') \vee x' \in (\neg NF \vee q_1 \vee q_4) \vee s(y') \in \neg NF \vee s(y') = s(x') \\
& & \qquad \vee s(y') = 0 \vee (\exists x_2 : \ s(y') = p(x_2)) \\
& & \wedge \forall y, x', y' : \ y \neq p(y') \vee x' \in (\neg NF \vee q_1 \vee q_4) \vee p(y') \in \neg NF \vee p(y') = s(x') \\
& & \qquad \vee p(y') = 0 \vee (\exists x_2 : \ p(y') = p(x_2)) \\
& & \wedge \forall y, x', y_1, y_2 : \ y \neq y_1 + y_2 \vee x' \in (\neg NF \vee q_1 \vee q_4) \vee y_1 + y_2 \in \neg NF \\
& & \qquad \vee y_1 + y_2 = s(x') \vee y_1 + y_2 = 0 \vee (\exists x_2 : \ y_1 + y_2 = p(x_2))
\end{array}$$

Now, as after the explosion of x, this formula is simplified: the first part is actually $\top$ because of the equation $0 = 0$, the third part is reduced to $\top$ using D_1 and Q_6 and the last part reduces to $\top$ using D_3. Finally, the second part is reduced using C_1, Q_4, Q_5, D_3 and S_2. We get the formula

$$\phi_3 \equiv \forall x', y' : \ x' \in (\neg NF \vee q_1 \vee q_4) \vee y' \in (\neg NF \vee q_1 \vee q_4) \vee s(x') = s(y')$$

Now, $[\![\neg(q_1 \vee q_4 \vee \neg NF)]\!] = \{0\}$ is finite. Therefore, by rule S_6:

$$\begin{array}{lll}
\phi_3 & \rightarrow^*_{S_6} & \forall x', y' : \ (x' \neq 0 \vee y' \neq 0 \vee s(x') = s(y')) \\
& \rightarrow_{D_1} & \forall x', y' : \ (x' \neq 0 \vee y' \neq 0 \vee x' = y') \\
& \rightarrow_{Q_5} & \forall y' : \ (y' \neq 0 \vee y' = 0) \\
& \rightarrow_{Q_5} & \top
\end{array}$$

This shows that $x + y$ is indeed inductively reducible.

Provided that the set NF is recognized by a tree automaton with equality tests, this technique can be applied in the general case for deciding the inductive reducibility of t. When t is a term $f(x_1, \ldots, x_n)$ one has only to require that the language of irreducible ground terms in which f does not occur can be recognized by a tree automaton with equality tests (which includes, of course, the regular case). This was the case in the example above. This is always the case in problems derived from sufficient-completeness tests.

Acknowledgment

We acknowledge Laurence Puel for her help. We would like also to acknowledge the referees who spent a lot of time on the manuscript and gave relevant comments which helped in improving the paper.

References

Aristide, 1989. A logic of semi-functions, inclusion and equality. The setting. Research Report 89-R-058, Centre de Recherche en Informatique, Nancy, France.

Leo Bachmair and David A. Plaisted, 1985. Termination orderings for Associative-Commutative rewriting systems. *J. Symbolic Computation*, Vol. 1, No. 4, pp. 329–349.

B. Bogaert, 1990. *Automates d'arbres avec tests d'égaliés.* PhD thesis, Université des Sciences et Techniques de Lille Flandres-Artois.

B. Bogaert and S. Tison, 1992. Equality and disequality constraints on brother terms in tree automata. In A. Finkel, editor, *Proc. 9th Symp. on Theoretical Aspects of Computer Science*, Paris, Springer-Verlag.

A. Colmerauer, 1984. Equations and Inequations on Finite and Infinite Trees. In *FGCS'84 Proceedings*, pp. 85–99.

Hubert Comon, 1988. Unification et Disunification: Théorie et Applications. Thèse de Doctorat, Institut National Polytechnique de Grenoble, France.

Hubert Comon, 1989. Inductive Proofs by Specifications Transformation. In *Proc. 3rd Rewriting Techniques and Applications 89, Chapel Hill, LNCS 355*, pp. 76–91. Springer-Verlag.

Hubert Comon, 1990. Equational Formulas in Order-Sorted Algebras. In *Proc. ICALP, Warwick.* Springer-Verlag.

Hubert Comon, 1991. Disunification: a Survey. In Jean-Louis Lassez, Gordon Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson.* MIT Press.

Hubert Comon and Pierre Lescanne, 1989. Equational Problems and Disunification. *J. Symbolic Computation*, Vol. 7, pp. 371–425.

Nachum Dershowitz, 1987. Termination of Rewriting. *J. Symbolic Computation*, Vol. 3, No. 1, pp. 69–115.

Nachum Dershowitz and Jean-Pierre Jouannaud, 1990. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pp. 243–309. North-Holland.

M. Gécseg and M. Steinby, 1984. *Tree Automata.* Akademia Kiadó, Budapest.

Joseph Goguen, Jean-Pierre Jouannaud, and José Meseguer, 1985. Operational Semantics for Order-Sorted Algebra. In W. Brauer, editor, *Proc. 12th ICALP, Nafplion, LNCS 194.* Springer-Verlag.

Joseph Goguen and José Meseguer, 1987. Order-Sorted Algebra I: Partial and Overloaded Operators, Errors and Inheritance. Draft, Computer Science Lab., SRI International.

Joseph Goguen and José Meseguer, 1987. Order-Sorted Algebra Solves the Constructor-Selector, Multiple Representation and Coercion Problems. In *Proc. 2nd IEEE Symp. Logic in Computer Science, Ithaca, NY.*

Jean-Pierre Jouannaud and E. Kounalis, 1986. Automatic Proofs by Induction in Equational Theories without Constructors. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*

Deepak Kapur, Paliath Narendran, Daniel Rosenkratz, et al., 1991. Sufficient-completeness, ground-reducibility and their complexity. *Acta Informatica*, Vol. 28, pp. 311–350.

Deepak Kapur, Paliath Narendran, and Hantao Zhang, 1987. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, Vol. 24, No. 4, pp. 395–415.

Claude Kirchner, 1988. Order-Sorted Equational Unification. In *Proc. 5th Int. Conference on Logic Programming, Seattle.*

Claude Kirchner, Hélène Kirchner, and José Meseguer, 1988. Operational Semantics of OBJ-3. In *Proc. 15th ICALP, Tampere, LNCS 317.* Springer-Verlag.

K. Kunen, 1987. Negation in Logic Programming. *J. Logic Programming*, Vol. 4, pp. 289–308.

J.-L. Lassez, M. J. Maher, and K. G. Marriot, 1986. Unification Revisited. In *Proc. Workshop on Found. of Logic and Functional Programming, Trento, LNCS 306.* Springer-Verlag.

M. J. Maher, 1988. Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees. In *Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh*, pp. 348–357.

A. I. Mal'cev, 1971. Axiomatizable Classes of Locally Free Algebras of Various Types. In *The Metamathematics of Algebraic Systems. Collected Papers. 1936-1967*, pp. 262–289. North-Holland.

M. Mehl, 1988. *Gleichungsdefinierte Probleme. Lösungsmethoden und Anwendungen bei algebraischen Spezifikationen.* Projektarbeit, Universität Kaiserslautern. In german.

P.D. Mosses, 1989. Unified Algebras and Institutions. In *Proc. 4th IEEE Symp. Logic in Computer Science.*

D. Plaisted, 1985. Semantic Confluence Tests and Completion Methods. *Information and Control*, Vol. 65, pp. 182–215.

G. Smolka, W. Nutt, J. A. Goguen, et al., 1989. Order-Sorted Equational Computation. In Hassan Aït-Kaci, Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures, Volume 2, Rewriting Techniques*, chapter 10, pp. 297–367. Academic Press, New York, N.Y.

Gert Smolka, 1986. Order-Sorted Horn Logic: Semantics and Deduction. Research Report SR-86-17, Univ. Kaiserslautern.

Gert Smolka, 1989. *Logic programming over polymorphically order-sorted types.* PhD thesis, Univ. Kaiserslautern.