J. Symbolic Computation (1997) **11**, 1–000

Completion of Rewrite Systems with Membership Constraints Part II: Constraint Solving†

Hubert Comon

CNRS and LRI,

Bat. 490, Université de Paris Sud,

91405 ORSAY cedex, France.

E-mail Hubert.Comon@lri.fr

(Received April 92, revised June 94)

We consider a constraint system involving equations and membership constraints. Membership constraints are interpreted as the membership to some recognizable tree languages. The terms may contain a restricted kind of *context variables*. We give a set of constraint solving rules which is proved to be complete and terminating. This gives a unification algorithm for an order-sorted logic (without any restriction on the signature) which contains some restricted kind of second order variables.

Introduction

In a first part of this work we have shown that order-sorted equational logic can be seen as a constrained equational logic where the constraints are membership conditions to recognizable tree languages. We made no assumption on the signature (i.e. the tree automaton). The rules of the constrained rewrite system do not assume any hypothesis (such as "sort decreasingness"). However, as it is always the case in non-trivial constrained rewrite systems, the classical critical pair lemma does not hold any more. We gave a set of additional deduction rules which allow to recover this lemma. The new deduction rules may introduce some *context variables* which in turn may occur in the constraints. This left us to solve some constraints involving equations, context variables and membership constraints.

The aim of this paper is to show how to solve the constraints that are involved in the deduction mechanism of the first part. This may be interesting in its own since this provides with a unification algorithm for an order-sorted logic with context variables and can be read independently of the first part.

This can also be compared with *unification of term schemes* of various kind (Chen & Hsiang, 1991; Salzer, 1992; Comon, 1995; R. Galbavý and M. Hermann, 1992). Indeed,

† This research was partly supported by *GRECO de programmation du CNRS* and partly by the ESPRIT Working Group CCL.

0747–7171/90/000000 + 00 $03.00/0 © 1997 Academic Press Limited

context variables provide with a schematization of terms which is actually orthogonal to the above-cited formalisms. For example, the set of all (ground) terms of the form

$$f(g^n(a), h^n(a))$$

cannot be represented in our formalism, because we have essentially *regular* tree languages and we cannot express that there are as many g's as h's (though we can still express $f(g^n(a), g^n(a))$ as we will see). In contrast, this schematization is possible in (Chen & Hsiang, 1991; Salzer, 1992; Comon, 1995; R. Galbavý and M. Hermann, 1992). However, in none of these papers, it is possible to express a schematization involving a regular expression of star height larger than 2. For example, $g^{n_1}hg^{n_2}h \dots g^{n_k}ha$ with $k \geq 0, n_1, \dots, n_k \geq 0$ is easily represented in our formalism as $X(a)|X \in (g^*h)^*$ and cannot be represented in (Chen & Hsiang, 1991; Salzer, 1992; Comon, 1995; R. Galbavý and M. Hermann, 1992).

We recall the main components of the constraint system in section 1, then we give the transformation rules and prove their correctness in sections 2 (for the decomposition rules), 3 (for the merge rules) and section 4 (for the other rules). Finally, we prove the termination and completeness of the rule system in sections 5 and 7 respectively.

1. The Constraint System

We recall here the main components of our constraint system and refer to the first part of our work for more details.

1.1. Terms

$T(F, \mathcal{X}, \mathcal{CX})$ is the set of all terms that are built over the finite (graded) alphabet F, the set of variable symbols $\mathcal{X}$ and the set of *context variables* $\mathcal{CX}$. Elements from $\mathcal{X}$ are considered as having arity 0, and elements of $\mathcal{CX}$ as having arity 1. *Positions* are, as usual, strings of integers defining paths in the terms. $t|_p$ is the subterm of t at position p and $t[u]_p$ is the term obtained by replacing $t|_p$ with u in t, at position p. $T(F)$ is the set of terms which do not contain any variable.

$GC(F, \mathcal{X}, \mathcal{CX})$ is the set of *general contexts*: it is the subset of terms in $T(F \cup \{\circ\}, \mathcal{X}, \mathcal{CX})$ which contain exactly one occurrence of the constant $\circ$, the "hole". $C(F)$ is the set of general contexts of the form $t[\circ]_p$ for some $t \in T(F)$ and $p \in Pos(t)$. While writing the contexts, we will sometimes omit the $\circ$, writing simply X instead of $X(\circ)$. The binary operation $\cdot$ on $GC(F, \mathcal{X}, \mathcal{CX})$ is defined by: $t[\circ]_p \cdot u[\circ]_q \stackrel{\text{def}}{=} t[u[\circ]_q]_p$. We define in a similar way the mapping "$\cdot$" from $C(F, \mathcal{X}, \mathcal{CX}) \times T(F, \mathcal{X}, \mathcal{CX})$ into $T(F, \mathcal{X}, \mathcal{CX})$.

1.2. Sort expressions and Context Expressions

Q is a finite set of *sort symbols*. The set $\mathcal{SE}$ of *sort expressions* and the set $\mathcal{CE}$ of *context expressions* are the least sets which satisfy

(i) $Q \subseteq \mathcal{SE}$ and $\epsilon \in \mathcal{CE}$

(ii) $\mathcal{SE}$ and $\mathcal{CE}$ are closed under Boolean operations (this includes the presence of $\bot_S, \top_S \in \mathcal{SE}$ and $\bot_C, \top_C \in \mathcal{CE}$).

(iii) $f(q_1, \ldots, q_n) \in \mathcal{SE}$ whenever $q_1, \ldots, q_n \in \mathcal{SE}$ and $f \in F$

(iv) $f(q_1, \ldots, q_{i-1}, C, q_{i+1}, \ldots, q_n) \in \mathcal{CE}$ whenever $q_1, \ldots, q_n \in \mathcal{SE}$ and $C \in \mathcal{CE}$.

(v) $C \cdot q \in \mathcal{SE}$ if $C \in \mathcal{CE}$ and $q \in \mathcal{SE}$

(vi) $C \cdot C' \in \mathcal{CE}$ if $C, C' \in \mathcal{CE}$

(vii) $C^* \in \mathcal{CE}$ if $C \in \mathcal{CE}$.

1.3. Formulas

The set of formulas we consider as a constraint language is built over the atomic formulas:

(i) "$t \in C$" where $t \in GC(F, \mathcal{X}, \mathcal{CX})$ and $C \in \mathcal{CE}$

(ii) "$t \in q$" where $t \in T(F, \mathcal{X}, \mathcal{CX})$ and $q \in \mathcal{SE}$

(iii) "$s = t$" where $s, t \in T(F, \mathcal{X}, \mathcal{CX})$

(iv) "$X = t$" where $X \in \mathcal{CX}$ and $t \in GC(F, \mathcal{X}, \mathcal{CX})$.

and the logical connectives $\wedge, \vee, \exists x, \exists X$. In addition $\perp$ is the empty disjunction and $\top$ is the empty conjunction. Let $\mathcal{F}$ be the set of all such formulas.

A *membership constraint* is a formula $\phi \in \mathcal{F}$ which does not contain any equation. A *membership solved form* is a conjunction of constraints of the form $X \in C$ and $x \in q$ where X and x are variables occurring only once.

We do not consider here the general problem of solving formulas in $\mathcal{F}$. We rather consider the restricted class of formulas defined below. A *general unification problem* (in our framework) is an (existentially quantified) disjunction of formulas of the form:

$$s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge M$$

where

1 . M is a membership constraint,

2 . $s_1, \ldots, s_n, t_1, \ldots, t_n \in T(F, \mathcal{X}, \mathcal{CX})$.

3 . For every two occurrences of a context variable X in $s_1, t_1, \ldots, s_n, t_n$ the corresponding arguments of X are identical.

In these problems $=$ is assumed to be symmetric. Which means that we make no difference between the equation $s = t$ and $t = s$.

Example 1.1. *Assume that the alphabet consists of f (binary), and a (constant).*

$$\begin{aligned} \exists y, X, X_2, X_3.\ & f(X(x), Y(f(z, x))) = f(f(z, a), X(x)) \wedge f(x, y) = Y(f(z, x)) \\ & \wedge X_1(a) = f(a, X_2(a)) \wedge X_2(a) = f(X(x), X_3(a)) \wedge X_1 \in C \wedge z \in \underline{s} \end{aligned}$$

is a general unification problem, whereas $f(X(x), a) = f(a, X(y))$ is not a general unification problem (condition 3 is not satisfied).

The condition 3 is quite strong. It was inspired by the kind of problems we considered in Part I and has been shown to be stable by completion, as soon as it is stable by constraint solving. In other words, general unification problems cover the kind of problems we need to solve during the completion process. Removing condition 3, we would have, as a particular instance, associative unification, which is known to be a very hard problem. Anyway, we suspect that the problem would become undecidable, because of the undecidability of second-order unification which is a similar problem.

1.4. Semantics

Sort expressions are interpreted as recognizable sets of trees and context expressions as recognizable sets of trees "with one hole", i.e. subsets of $C(F)$.

Given a tree automaton $\mathcal{A}$ whose set of states contains Q, the interpretation $[\![q]\!]_{\mathcal{A}}$ of a basic sort symbol q is the set of trees that are accepted by the automaton in state q. ϵ is interpreted as $\circ$ (the identity on trees).

Logical connectives are interpreted according to the Boolean lattice of recognizable tree languages ($\wedge$ is intersection, $\neg$ is complement,...). The application of a function symbol to sort expressions (resp. sort expressions and a context expression) is interpreted according to the structure of F-algebra of recognizable tree languages. Finally, $C \cdot q$ (resp. $C \cdot C'$, resp. C^*) are interpreted according to the concatenation of recognizable tree languages (i.e. the monoid structure of $C(F)$): for example,

$$[\![C \cdot q]\!]_{\mathcal{A}} = \{t[u]_p \in T(F) \mid t[\circ]_p \in [\![C]\!]_{\mathcal{A}}, u \in [\![q]\!]_{\mathcal{A}}\}$$

Example 1.2. *Consider the automaton $\mathcal{A}$ defined by:*

$$\begin{array}{rcl@{\qquad}rcl}
0 & \to & \mathsf{nat} & 0 & \to & \mathsf{even} \\
s(\mathsf{nat}) & \to & \mathsf{nat} & s(\mathsf{even}) & \to & \mathsf{odd} \\
s(\mathsf{odd}) & \to & \mathsf{even} & \mathsf{nat}+\mathsf{nat} & \to & \mathsf{nat} \\
\mathsf{int}+\mathsf{int} & \to & \mathsf{int} & \mathsf{neg}+\mathsf{neg} & \to & \mathsf{neg} \\
-(\mathsf{nat}) & \to & \mathsf{neg} & -(\mathsf{neg}) & \to & \mathsf{nat} \\
\mathsf{nat} & \to & \mathsf{int} & \mathsf{odd} & \to & \mathsf{int} \\
\mathsf{even} & \to & \mathsf{int} & \mathsf{neg} & \to & \mathsf{int}
\end{array}$$

w.r.t this automaton, $[\![\mathsf{odd}]\!] = \{s^{2n+1}(0) \mid n \in \mathbf{N}\}$. Also, $[\![-(\mathsf{neg})]\!] \cup \{0\} = [\![\mathsf{nat}]\!]$ and $[\![s(\mathsf{odd})]\!] \cup \{0\} = [\![\mathsf{even}]\!]$.

A ground assignment σ is a *solution* of an atomic formula $t \in q$ (resp. $t \in C$) if $t\sigma$ belongs to $[\![q]\!]_{\mathcal{A}}$ (resp. $t\sigma \in [\![C]\!]_{\mathcal{A}}$). σ is a solution of an equation if the two members become identical after applying σ. Then, a solution of a general unification problem is defined following the usual interpretation of logical connectives.

Example 1.3. *Let us consider again example 1.1. Assume that the interpretation is given by the automaton:*

$$\begin{array}{rcl@{\qquad}rcl}
a & \to & q_i & q_i & \to & \underline{s} \\
f(q_i, q_i) & \to & q_p & f(q_p, q_i) & \to & q_i \\
f(q_i, q_p) & \to & q_i & f(q_p, q_p) & \to & q_p
\end{array}$$

$[\![q_i]\!]$ is the set of all trees with an odd number of leaves, $[\![q_p]\!]$ is the set of all trees with

an even number of leaves. Assume moreover that C in example 1.1 is the expression

$$\Big(\; f(q_p,\epsilon) \;\vee\; f(\epsilon,q_p) \;\vee\; \epsilon \;\vee\; f(q_i,(f(q_p,\epsilon)\vee f(\epsilon,q_p))^*)\cdot(f(\epsilon,q_i)\vee f(q_i,\epsilon)) \\ \vee\; f((f(q_p,\epsilon)\vee f(\epsilon,q_p))^*\cdot(f(\epsilon,q_i)\vee f(q_i,\epsilon)),q_i) \; \Big)^*$$

This expression is a bit complex. It expresses any context with an even number of a's. An automaton would be more appropriate for such a definition. Since the formulations are equivalent, we may use automata in the following.

Then the following assignment is a solution of the problem given in example 1.1: $\{x \mapsto a;\ z \mapsto a;\ Y \mapsto \epsilon;\ X_1 \mapsto f(a, f(f(a,a), f(a,\epsilon)))\}$

1.5. S_0-Unification problems

Some of our transformation rules will not preserve general unification problems. That is why we introduce another class of constraints which is invariant by transformations. This new class will be shown to contain general unification problems.

We call a sequence $S_0 = (s_1, t_1, \ldots, s_n, t_n) \in T(F, \mathcal{X}, \mathcal{CX})$ *admissible* if for all variable $X \in \mathcal{CX}$ and each two occurrences of X in any term(s) of the sequence, the corresponding two arguments of X are identical. For example, a conjunction $s_1 = t_1 \wedge \ldots \wedge s_n = t_n$ is a general unification problem iff the sequence $(s_1, t_1, \ldots, s_n, t_n)$ is admissible.

Let S_0 be an admissible sequence. A *S_0-unification problem* (or a *unification problem w.r.t. S_0*) is a disjunction of existentially quantified conjunctions of formulas of the form

$$\phi \equiv \quad s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge S \wedge M$$

where

1 M is a membership constraint,

2 S is a conjunction of *solved equations*, i.e. a conjunction $x_1 = u_1 \wedge \ldots \wedge x_m = u_m$ where $x_1, \ldots, x_m \in \mathcal{X} \cup \mathcal{CX}$ and they do not occur in $s_1 = t_1, \ldots, s_n = t_n$. Moreover, for every $i \geq 1$, x_i does not occur in $u_i, x_{i+1}, u_{i+1}, \ldots, x_m, u_m$. (This corresponds to *DAG solved forms* in (Jouannaud & Kirchner, 1991)).

3 $s_1, \ldots, s_n \in T(F, \mathcal{X})$,

4 Each term t_i either belongs to $T(F, \mathcal{X})$ or has the form $Y_i(t'_i)$ where $t'_i \in T(F, \mathcal{X})$ and Y_i does not occur in any other term t_j.

5 For every solution σ of the set of equations of ϕ, for every term $X(s_0)$ occurring in S_0 and and every (first-order) term $X(s)$ occurring in some equation of ϕ, $s\sigma \equiv s_0\sigma$.

6 For all $X \in \mathcal{CX}$ occurring in ϕ, for every solution σ of the equations of ϕ and for every $t, t' \in \mathrm{Arg}_\phi(X)$, $t\sigma \equiv t'\sigma$. $\mathrm{Arg}_\phi(X)$ is defined as the least set such that:

$$\begin{cases} t \in \mathrm{Arg}_\phi(X) & \text{If } X(t) \text{ occurs in some term of } S_0 \text{ or in some term of the equational part of } \phi \\ t[u]_p \in \mathrm{Arg}_\phi(Y) & \text{If } X = c[Y(t[\circ]_p)] \text{ is an equation in } S \text{ and } u \in \mathrm{Arg}_\phi(X) \end{cases}$$

Informally, Arg_ϕ is the function which returns the value of the argument of a variable $X \in \mathcal{CX}$. The initial value of Arg (i.e. in S_0) can only be changed in a consistent way. That is the meaning of condition 5. In the following, we will show that S_0-unification problems are transformed by the constraint solving rules into S_0-unification problems, hence properties 5 and 6 are kept along the transformations. This means in particular that, starting with a general unification problem such that $(s_1, t_1, \ldots, s_n, t_n)$ is admissible, then any substitution σ which is computed by the transformation rules will be such that $(s_1\sigma, t_1\sigma, \ldots, s_n\sigma, t_n\sigma)$ is admissible. This meets the condition of lemma 3.2 of (Comon, 1993a).

Note that it might be very difficult to check whether a constraint is a S_0-unification problem. But we never need such a general algorithm: this will only be an invariant of our transformation rules.

EXAMPLE 1.4. *$\exists Y, \exists x_1.x = f(x_1, a) \wedge y = X(f(x_1, a)) \wedge X = Y \cdot Z$ is a S_0- unification problem where $S_0 = (f(X(x), a), Z(x))$.*

EXAMPLE 1.5. *Using the same alphabet as in example 1.1, the following is a unification problem w.r.t. $S_0 = (f(X(x), Y(f(z, x))))$*

$$\begin{aligned} \exists y, x_1, x_2, X_1, X_2, X_3.\ x_1 = X(x) \wedge x_2 = Y(f(z,x)) \wedge f(x_1, x_2) = f(f(z,a), x_1) \\ \wedge f(x,y) = x_2 \wedge X_1 = f(a, X_2) \wedge X_2 = f(x_1, X_3) \wedge X_1 \in C \wedge z \in \underline{s} \end{aligned}$$

It is logically equivalent to the unification problem of example 1.1. This is actually a general rule as shown by the next lemma.

LEMMA 1.6. *Every general unification problem ϕ is logically equivalent to a unification problem w.r.t. the sequence of terms occurring in an equation of ϕ.*

PROOF. We use the following *variable abstraction rule*:

$$P \mapsto \exists x.x = X(s) \wedge P[X(s)/x]$$

where $P[X(s)/x]$ is the problem in which all occurrences of $X(s)$ are replaced with x.

This rule is applied only when X occurs at a non-root position in an equation $t_1 = t_2$ between terms in $T(F, \mathcal{X}, \mathcal{CX})$ or when it occurs at the root of an equation $X(s) = Y(t)$, or when $s \in T(F, \mathcal{X})$ and $X(s)$ is a subterm of some U_i or when $X(s)$ has two occurrences in the problem. This variable abstraction rule terminates since it can be applied at most as many times as there are occurrences of context variables.

Of course the variable abstraction rule transforms a formula in a logically equivalent one. On the other hand, it is not difficult to check that all properties of unification problems are satisfied on the irreducible forms w.r.t. the variable abstraction rule.

□

Hence, without loss of generality, from now on, we only consider S_0-unification problems and, by abuse of language still call them unification problems.

1.6. NORMALIZATION RULES

In what follows, we will use some rules which do not really transform a unification problem into a unification problem. For example, an atomic formula may be transformed into a disjunction of unification problems. In order to keep the transformation on unification problems, we are going to assume that a set of *normalizing rules* is applied after each transformation step. These rules are given in figure 1

The first rules keep the formulae in disjunctive normal form and remove the trivial

$$\begin{array}{rcl} P \wedge (Q \vee R) & \rightarrow & (P \wedge Q) \vee (P \wedge R) \\ P \wedge \top & \rightarrow & P \\ P \wedge \perp & \rightarrow & \perp \\ s = s & \rightarrow & \top \end{array}$$

$$\phi \quad \rightarrow \quad \phi^{-X} \vee (\phi \wedge X \in \neg\epsilon)$$

Figure 1. Normalizing rules

equations. The second rule is applied only if a new variable X is introduced in a formula ϕ at some transformation step. It chooses non-deterministically whether X is an empty context or not. ϕ^{-X} is the formula obtained from ϕ by removing all occurrences of X (i.e. X is replaced with the empty context $\circ$).

In what follows, we always assume an eager application of these normalizing rules. By "result of the application of rule RR" one should now understand the formula obtained by applying the rule RR and then normalizing according to rules of figure 1, which of course preserve the solutions of the problem. These rules (especially the first and fourth one) can be very expensive. They are not mandatory. However, we keep them for sake of clarity. We also have to take care of them in the termination proofs.

2. Decomposition Rules

We are going to design a constraint solving algorithm for our unification problems. To this end, we will define a set of rules (scheme of rewrite rules on unification problems) and prove their correctness and termination. Then, inspecting the irreducible formulas, we will show that every irreducible formula is either $\perp$ or solvable. Actually, irreducible formulas will be in *solved form* which means something stronger than solvability or failure and will be explained in section 7.

Classical unification rules involve decompositions, clashes, occur-check and merge and/or variable-elimination (see (Jouannaud & Kirchner, 1991)). We will need similar rules, and, in addition, we have to design some rules similar to the *imitation* rule of higher-order unification, in order to cope with the context variables. We also need to decompose membership constraints. Roughly, the goal is to design rules which reduce the general case of unification problems to unification problems where each atomic formula contains at least one variable among its arguments. Then this variable is eliminated using a Merge rule.

In all our unification problems, we will assume that each context variable is constrained (in each conjunction) by a condition $X \in C$ such that $\circ \notin [\![C]\!]_{\mathcal{A}}$. This is ensured by the normalization rules of the former section.

We start with the decomposition rules in this section. This involves technical constructions on automata for the decomposition of membership constraints. These rules are given in figure 2 † (except for the so-called "flexible/flexible" case of (Huet, 1976) which is combined with a merge rule and displayed as the last rule of figure 4). Some functions (such as f^{-1} and C^{-1}) are used in the rules and explained below.

† We use a "inference like" presentation for the unification rules but the reader should keep in mind that these inference rules are destructive; we rather rewrite the formulas during the unification process. We chose to keep the notation of inference rules because it is easier to display the rules in this way.

$$\frac{f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n)}{t_1 = u_1 \wedge \ldots \wedge t_n = u_n}$$

$$\frac{f(t_1,\ldots,t_n) \in q}{\bigvee_{(q_1,\ldots,q_n)\in f^{-1}(q)} t_1 \in q_1 \wedge \ldots \wedge t_n \in q_n}$$

$$\frac{f(t_1,\ldots,t_n)[U]_i \in C}{\bigvee_{(q_1,\ldots,q_{i-1},C_i,q_{i+1},\ldots,q_n)\in f^{-1}(C)} U \in C_i \wedge \bigwedge_{i\neq j} t_j \in q_j}$$

$$\frac{X(s) \in q \wedge X \in C}{\bigvee_{(C',q')\in C^{-1}(q)} X \in C' \wedge s \in q'}$$

$$\frac{X(u) \in C}{\bigvee_{(C_0,C_1)\in \mathrm{split}(C)} X \in C_0 \wedge u \in C_1}$$

$$\frac{f(s_1,\ldots,s_n) = X(s)}{\bigvee_{1\leq i\leq n} \exists Y:\ X = f(s_1,\ldots,s_n)[Y]_i \wedge s_i = Y(s)}$$

Figure 2. Decomposition Rules

The first lemma is adapted from (Comon, 1990; Comon & Delor, 1994). It allows to decompose expressions $f(t_1,\ldots,t_n) \in \underline{s}$ and expressions $f(t_1,\ldots,t_{i-1},t_i[\circ]_p,t_{i+1},\ldots,t_n) \in C$. The function f^{-1} can be derived from an automaton accepting $[\![\underline{s}]\!]_{\mathcal{A}}$ (resp. $[\![C]\!]_{\mathcal{A}}$), and reading the automaton in reverse direction. We use a direct construction below, avoiding to construct an automaton for every expression encountered in the constraint solving process.

Next, we will consider decompositions of membership expressions headed by a context variable.

Before stating and proving the lemma, let us show an example of how the decomposition rule works.

EXAMPLE 2.1. *Assume for example given the semantics of example 1.3.* $f(x,y) \in q_p$ *can be decomposed into* $(x \in q_p \wedge y \in q_p) \vee (x \in q_i \wedge y \in q_i)$*; there is a set of pairs of states* $f^{-1}(q_p)$ *(here* $\{(q_p,q_p),(q_i,q_i)\}$*) such that* $f(x,y) \in q_p$ *iff* $(x,y) \in f^{-1}(q_p)$.

LEMMA 2.2. *For every* $f \in F$*, there is a function* f^{-1} *which associates to each* $\underline{s} \in \mathcal{SE}$ *a finite set of tuples of sort expressions, in such a way that*

$$f(t_1,\ldots,t_n) \in [\![\underline{s}]\!]_{\mathcal{A}} \;\Leftrightarrow\; \exists(\underline{s}_1,\ldots,\underline{s}_n) \in f^{-1}(\underline{s}), \forall i \in \{1,\ldots,n\}, t_i \in [\![\underline{s}_i]\!]_{\mathcal{A}}$$

Similarly, for every $f \in F$*, there is a function, again denoted* f^{-1}*, which associates to each* $C \in \mathcal{CE}$ *a finite set of tuples* $(\underline{s}_1,\ldots,\underline{s}_{i-1},C_i,\underline{s}_{i+1},\ldots,\underline{s}_n)$ *where each* $\underline{s}_i$ *is a sort* *expression and* C_i *is a context expression, in such a way that*

$$f(t_1,\ldots,t_{i-1},t_i[\circ]_p,t_{i+1},\ldots,t_n) \in [\![C]\!]_{\mathcal{A}} \;\Leftrightarrow\; \exists(\underline{s}_1,\ldots,\underline{s}_{i-1},C_i,\underline{s}_{i+1},\ldots,\underline{s}_n) \in f^{-1}(C), \forall i \neq j, t_j \in [\![\underline{s}_j]\!]_{\mathcal{A}} \text{ and } t_i[\circ]_p \in [\![C_i]\!]_{\mathcal{A}}$$

The proof of this technical lemma is given in appendix A.

The following lemma corresponds to decompositions of q into $C \cdot q'$ and of C into $C_0 \cdot C_1$. This will allow to decompose expressions $X(s) \in \underline{s}$ and $X(u) \in C$.

Note that it cannot be reduced to finding the set of prefixes of a regular expression, because we want the *decompositions* in two parts: we need *simultaneously* the prefix and the corresponding remainder.

As before, let us show the rule on an example

EXAMPLE 2.3. *Let us go on with the automaton of example 1.3. We want now to decompose* $X(x) \in q_p$. *For, we can derive from the automaton* $\mathcal{A}$ *an automaton on contexts where the "hole" holder is given by the special symbol* Ω. *The states are then copied. One sample is marked with the additional superscript* Ω *and indicates that there is exactly one* Ω *in any tree recognized in this state. For example, with our automaton, we get the automaton on contexts by adding the rules:*

$$\begin{array}{rclrcl}
f(q_i,q_i^\Omega) & \to & q_p^\Omega & f(q_i^\Omega,q_i) & \to & q_p^\Omega \\
f(q_p^\Omega,q_p) & \to & q_p^\Omega & f(q_p,q_p^\Omega) & \to & q_p^\Omega \\
f(q_i^\Omega,q_p) & \to & q_i^\Omega & f(q_p,q_i^\Omega) & \to & q_i^\Omega \\
f(q_p^\Omega,q_i) & \to & q_i^\Omega & f(q_i,q_p^\Omega) & \to & q_i^\Omega \\
\Omega & \to & q_p^\Omega & & &
\end{array}$$

q_i^Ω *accepts the contexts with an odd number of a's and* q_p^Ω *accepts the contexts with an even number of a's. Then,* $X(x) \in q_p$ *iff either* X *contains an even number of a's and* x *contains also an even number of a's or else, they contain both an odd number of a's: the result of the decomposition is* $(X \in q_p^\Omega \wedge x \in q_p) \vee (X \in q_i^\Omega \wedge x \in q_i)$. *Of course, membership to any* q^Ω *can be expressed using a context expression.*

That is roughly how we proceed in the proof of the next lemma:

LEMMA 2.4. *For every context expression* C, *there is a function* C^{-1} *mapping every sort expression* $\underline{s}$ *to a finite set of pairs* $(C',\underline{s}')$ *of a context expression and a sort expression, in such a way that*

$$(t[s]_p \in [\![\underline{s}]\!]_{\mathcal{A}} \wedge t[\circ]_p \in [\![C]\!]_{\mathcal{A}}) \;\Leftrightarrow\; \exists(C',\underline{s}') \in C^{-1}(\underline{s}), t[\circ]_p \in [\![C']\!]_{\mathcal{A}} \wedge s \in [\![\underline{s}']\!]_{\mathcal{A}}$$

For every context expression C, *there is a finite set* $\mathrm{split}(C)$ *of context expressions such that* $t[u[\circ]_r]_q \in [\![C]\!]_{\mathcal{A}}$ *iff*

$$\exists(C_0,C_1) \in \mathrm{split}(C) : \; t[\circ]_q \in [\![C_0]\!]_{\mathcal{A}} \wedge u[\circ]_r \in [\![C_1]\!]_{\mathcal{A}}$$

The proof is given in appendix A.†

LEMMA 2.5. *All rules given in figure 2 preserve the set of solutions.*

† Viewing $C(F)$ as a free monoid on an infinite alphabet, the lemma can be derived from classical results in formal language theory.

PROOF. This is a consequence of lemmas 2.4 and 2.2, except for the last decomposition rule of figure 2. Concerning this last rule, assume that a solution σ assigns a context $t[\circ]_p$ to X. Then $p \neq \Lambda$ (as we assume that no context variable is assigned to the empty context). Then, we let i be the first symbol of p: $p = i \cdot q$. $t[s\sigma]_p \equiv f(s_1\sigma, \ldots, s_n\sigma)$ and $t|_i[s\sigma]_q \equiv s_i\sigma$. Then it is sufficient to assign $t|_i[\circ]_q$ to Y.

The converse inclusion of solutions is straightforward. □

Decomposition rules also preserve unification problems:

LEMMA 2.6. *If ϕ is a S_0-unification problem and if ψ is deduced from ϕ by decomposition, then ψ is a S_0-unification problem.*

PROOF. This is straightforward for all decomposition rules except the last one. Let us investigate all properties of unification problems

Condition 2 is satisfied since, on one hand, by condition 4 on the source formula, X, before applying the rule, only occurs in $f(s_1, \ldots, s_n) = X(s)$ and in $S \wedge M$. Hence, after applying the rule, X only occurs in the solved part and in the membership part. Moreover, by condition 2, before applying the rule, the occurrences of X in the solved part cannot be as a member of an equation. Hence, if $x_1 = u_1 \wedge \ldots \wedge x_m = u_m$ was the solved part before applying the rule, it becomes $x_1 = u_1 \wedge \ldots \wedge x_{m+1} = u_{m+1}$ after applying the rule, where $x_{m+1} \equiv X$ and $u_{m+1} = f(s_1, \ldots, s_n)[Y]_i$, and S is still a DAG solved form.

Condition 3 is obviously satisfied.

Condition 4 is satisfied since $s_i \in T(F, \mathcal{X})$ and Y does not occur in any other t_j.

Condition 5 is satisfied since Y does not occur in S_0 and we do not introduce new expressions $X(s)$.

Let $\phi \mapsto \phi'$ be a transformation by this decomposition rule. Then $\mathrm{Arg}_{\phi'}(X) \subseteq \mathrm{Arg}_{\phi}(X)$ and $\mathrm{Arg}_{\phi'}(Y) = \mathrm{Arg}_{\phi}(X) = \mathrm{Arg}_{\phi'}(X) \cup \{s\}$. Hence property 6 is an invariant of the transformation. □

3. Merge Rules

We cannot use here the *variable elimination* rule which replaces everywhere a variable x with a term t as soon as the equation $x = t$ is deduced. Indeed, such a rule would not preserve unification problems. A classical alternative is to use *merge rules*. Such a rule can be stated as follows (see e.g. (Jouannaud & Kirchner, 1991):

$$x = s \wedge x = t \quad \rightarrow \quad x = s \wedge s = t$$

if x is a variable, s, t are not variables and $|s| \leq |t|$

Unfortunately it refers to the sizes of s, t which are not adequate here. For, consider the system $x = X(s) \wedge x = Y(t)$ and assume that $|s| \leq |t|$ following the above rule, we only keep the smallest equation, leading to $x = X(s) \wedge X(s) = Y(t)$. We have not shown yet how to decompose $X(s) = Y(t)$, but it should be clear that there are at least two families of solutions: 1. X is a "prefix" of Y (i.e. $Y = X(Z)$ for some Z) 2. Y is a prefix of X (i.e. $X = Y(Z)$ for some Z). Both situations have to be considered for this "flexible/flexible" decomposition. Assume for example that X is a prefix of Y. Then we have, for some Z, $x = Y(Z(s)) \wedge Z(s) = t$. We can see already that replacing X with $Y(Z)$ may violate the size assumption we made before. But let us go further: assume that $t = s = x$. We get the sequence

$$x = X(x) \wedge x = Y(x) \to \exists Z.x = X(x) \wedge X = Y(Z) \wedge x = Z(x)$$

which contains, up to renaming, the original problem. Hence, the size condition is not adequate.

We may observe that there is a positive occur-check in the original problem. (Which is not a problem for standard unification which can be done without occur-check producing a unification algorithm for rational trees). Requiring that there is no (extended) positive occur-check might be an alternative to the size condition. In the case of standard unification, the termination when the size condition is replaced with the absence of extended positive occur-check was stated as an open problem in (Dershowitz *et al.*, 1991). It has not been solved yet. Here is the problem: we could use the merge rule and detect afterwards a positive occur-check which was not present in the original problem. Even more, the deduction of this positive occur-check could be delayed indefinitely (this could be possible since the occur-check graph is not growing monotonically). In such a case, the "potential occur-check" could still produce the engine of non-termination as above, without any occurrence of a failure. Nevertheless, we are going to follow this idea (giving a partial answer to this open question in a more general framework): we will assume that the merge rule is only applied to maximal variables w.r.t. the occur-check relation. Then we know that no cycle can ever occur which involves this variable. Then, the termination proof will use the (possibly infinite) union of all occur-check graphs. Of course, this relation will never be computed, but it can be used for the termination proof purposes only.†

3.1. Solved part of a unification problem and extended occur-check

Solved variables (in a conjunction ϕ) and the *solved part* of ϕ are defined as the minimal sets which satisfy:

(i) a variable which occurs only once in ϕ and which occurs as a member of an equation or as a member of a membership constraint is solved.

(ii) an equation whose member is a solved variable is in the solved part of ϕ

(iii) a membership constraint whose member is a solved variable is in the solved part of ϕ

(iv) if $x = s$ (resp. $X = t$) is an equation of ϕ such that x is a variable which does not occur in s (resp. X is a second-order variable not occurring in t) and if the only other occurrences of x (resp. X) are in the solved part of ϕ, then x is solved. (resp. X is solved)

(v) if $x \in \underline{s}$ (resp. $X \in C$) is a membership constraint of ϕ such that x is a variable (resp. X is a second order variable) which has no other occurrence in the unsolved part of ϕ, then x (resp. X) is solved .

† In a first version of this paper we used a more complex condition, involving an occur-check on terms which was monotonic (following a suggestion of J.-P. Jouannaud). We found later this simpler argument and use it already for standard unification (Comon, 1993b).

Merge

$$\frac{x \in q \wedge x \in q'}{x \in q \wedge q'}$$

$$\frac{X \in C \wedge X \in C'}{X \in C \wedge C'}$$

$$\frac{X = t[Y]_p \wedge X \in C}{X = t[Y]_p \wedge t[Y]_p \in C} \quad \text{If } X \not\equiv Y$$

Figure 3. The merge rules: part 1

It is possible to associate the solved part of a formula ϕ with a substitution σ_ϕ defined as follows:

(i) If x (resp. X) does not occur as a member of an equation, then $x\sigma_\phi \equiv x$ (resp. $X\sigma_\phi \equiv X$)

(ii) If $x = s$ (resp $X = U$) is a solved equation, then $x\sigma_\phi \equiv s\sigma_\phi$ (resp. $X\sigma_\phi \equiv U\sigma_\phi$)

This defines a unique substitution (see (Jouannaud & Kirchner, 1991)) indeed.

By definition of unification problems, the set ψ of equations in ϕ between second-order terms is always solved (regardless to membership constraints). Therefore, it is possible to associate ϕ with another substitution, $\tau_\phi \stackrel{\text{def}}{=} \sigma_\psi$.

Finally, a variable x is *almost solved* in a conjunction of equations and membership constraints ϕ if its only occurrence at a non-root position of a first-order term are in the solved part of ϕ or in the membership part of ϕ.

Let ϕ be a quantifier-free conjunction of atomic formulas. The *strict equality* $=_{S,\phi}$ is the least equivalence relation on the first-order (resp. context) variables of ϕ such that

$$x = y \text{ is in } \phi \quad \Rightarrow \quad x =_{S,\phi} y.$$

The *occur-check* relation $\geq_\phi$ is the smallest quasi-ordering (i.e. reflexive and transitive relation) on the first-order (resp. context) variables of ϕ which satisfies

$$x = t[y]_p \text{ is in } \phi \quad \Rightarrow \quad x \geq_\phi y$$

Finally, the *strict occur-check relation* is the relation $>_\phi \stackrel{\text{def}}{=} \geq_\phi \setminus =_{S,\phi}$.

No transformation rule will apply to the solved part of a conjunction. This part does not play any role in the termination proof. We say that a variable x is *maximal* in ϕ if $>_\phi$ is irreflexive and x is maximal w.r.t. $>_\phi$ among the variables of the unsolved part of ϕ. It can be easily checked whether $>_\phi$ is irreflexive or not.

3.2. The merge rules

The merge rules are displayed on figures 3 and 4

Let us elaborate a little bit on the last rule of figure 4. This rule is actually a number of merges followed by decompositions. It solves the "flexible/flexible" case. First

$$\frac{x = s \wedge x \in q}{x = s \wedge s \in q}$$

$$\frac{x = f(s_1, \ldots, s_m) \wedge x = g(t_1, \ldots, t_n)}{x = f(s_1, \ldots, s_m) \wedge f(s_1, \ldots, s_m) = g(t_1, \ldots, t_n)}$$

$$\frac{x = f(s_1, \ldots, s_m) \wedge x = X(t)}{x = f(s_1, \ldots, s_m) \wedge f(s_1, \ldots, s_m) = X(t)}$$

$$\frac{x = X_1(s_1) \wedge \ldots \wedge x = X_n(s_n)}{\bigvee_{i=1}^{n} \phi_i \vee \bigvee_{\substack{n \geq m \geq 2 \\ \Phi \text{ surjective} \\ \Phi : \{1, \ldots, n\} \to \{1, \ldots, m\} \\ f \in F, k = a(f) \geq m \\ 1 \leq i_1 < \ldots < i_m \leq k \\ \{j_1, \ldots, j_m\} = \mathrm{Inv}(\Phi, \{1, \ldots, n\})}} \psi_{\Phi, f, i_1, \ldots, i_m}}$$

where

$$\phi_i \stackrel{\text{def}}{=} \exists Z_1, \ldots, Z_{j-1}, Z_{j+1}, \ldots, Z_n.\ x = X_i(s_i) \wedge \bigwedge_{j \neq i} s_i = Z_j(s_j) \wedge X_j = X_i \cdot Z_j$$

$$\psi_{\Phi, f, i_1, \ldots, i_m} \stackrel{\text{def}}{=} \exists Z_0, \ldots, Z_n, \exists x_1, \ldots x_k.\ \bigwedge_{j=1}^{n} X_j = Z_0(E[Z_j]_{i_{\Phi(j)}})$$

$$\wedge \bigwedge_{j=1}^{n} x_{i_{\Phi(j)}} = Z_j(s_j) \wedge\ x = Z_0(f(x_1, \ldots, x_k))$$

$$E \stackrel{\text{def}}{=} f(x_1, \ldots, x_k)[Z_{j_1}(s_{j_1})]_{i_{\Phi(j_1)}} \ldots [Z_{j_m}(s_{j_m})]_{i_{\Phi(j_m)}}$$

Let ϕ is the source formula of the rule. All these Merge rules assume that there is no positive occur-check, that x is a maximal variable in the unsolved part of ϕ, w.r.t. the occur-check relation $\geq_\phi$ and that no other rule than a merge can be applied.
Moreover, the last rule assumes that $x = X_1(s_1) \ldots, x = X_n(s_n)$ are all equations of ϕ whose one member is x and that no other merge rule can be applied.

Figure 4. Merge rules. Part 2

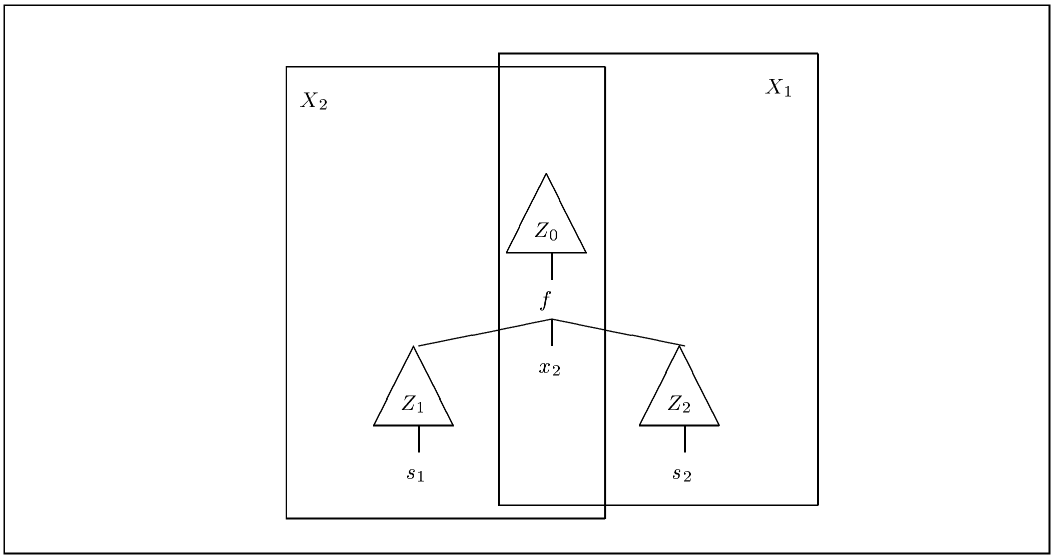

Figure 5. Merge + Decomposition, an example where the "holes" have incomparable positions

let us explain the notations. We use $t[t_1]_{p_1} \ldots [t_n]_{p_n}$ to denote the simultaneous replacement of terms $t_1, \ldots, t_n$ at uncomparable positions $p_1, \ldots, p_n$ of t. We also denote $\mathrm{Inv}(\Phi, \{1, \ldots, n\})$ any set of representatives of the classes under the equivalence relation $\sim_\Phi$ defined on $\{1, \ldots, n\}$ by $i \sim_\Phi j$ iff $\Phi(i) = \Phi(j)$.

The rule may look complicated and some restrictions may seem unnecessary. Let us give some informal explanations of both the rule and the restrictions on its application. For simplicity, assume $n = 2$. There are actually two large disjunctions in the target of the rule. These disjunctions correspond to two cases: for every solution $x\sigma \equiv X_1\sigma[s_1\sigma]_{p_1} \equiv X_2\sigma[s_2\sigma]_{p_2}$, either p_1 and p_2 are comparable (or, for $n > 2$: there is a position which is smaller than all other positions): this is the first disjunction. Or they are not comparable (more generally, there is no minimal position): this is the second disjunction. Consider the first case. With only this case, we can simplify the rule into

$$\begin{array}{l} x = X_1(s_1) \wedge x = X_2(s_2) \vdash (\exists Y : \ x = X_1(s_1) \wedge s_1 = Y(s_2) \wedge X_2 = X_1 \cdot Y) \\ \qquad\qquad\qquad\qquad\qquad\quad \vee (\exists Y : \ x = X_2(s_2) \wedge s_2 = Y(s_1) \wedge X_1 = X_2 \cdot Y) \end{array}$$

each of the two conjunctions corresponding either to $p_1 \leq p_2$ or to $p_2 \leq p_1$. If we have only unary function symbols, then this rule alone is sufficient. But there is still a difficulty. Indeed, we have to choose carefully which equations are kept. For, consider the problem $x = X(s) \wedge x = Y(t)$ where $X(s)$ has more nodes than $Y(t)$ (hence $X(s)$ is greater w.r.t. the size). If we merge the two equations, keeping the smallest one, we get $x = Y(t) \wedge Y(t) = X(s)$ which leads, by decomposition to the two problems: $\exists Z : x = Y(t) \wedge X = Y \cdot Z \wedge t = Z(s)$ and $\exists Z : x = Y(t) \wedge Y = X \cdot Z \wedge s = Z(t)$. But this last problem is no longer a unification problem because Y occurs both as a member of an equation and inside an equation between first order terms. This situation is avoided in our set of rules because we choose which equation is kept (among $x = X(s)$, $x = Y(t)$) *after* the decomposition step.

Now, let us consider the last large disjunction (the most complicated one), which corresponds to (some) uncomparable positions of the holes in a solution. For example, $x = X(s) \wedge x = Y(t)$ will always have $x = f(s,t) \wedge X = f(t, \circ) \wedge Y = f(\circ, s)$ as a solution, for every binary function symbol f. For $n = 2$, the situation is depicted in figure 5. We guess where is the least common ancestor of the two positions p_1, p_2. We call Z_0 the context above this position and f (which must have an arity greater than 2) the label at this position. Then, below, there are two contexts Z_1 and Z_2 before we find the terms s_1 and s_2.

There is a simpler (and more natural) expression of the case of uncomparable positions: X should be $Z(t, \circ)$ and Y should be $Z(\circ, s)$. However this requires a non monadic second-order variable Z which is beyond our logic. Moreover, the terms below Z are not identical: we would loose one of the basic properties of our unification problems (and might fall into the general case second-order unification, which is undecidable).

Now, the last merge rule is an expression of the general situation in which we have not only two equations, but we merge n equations simultaneously: we consider a context Z_0 which is the maximal context shared by all X_i's. Then there should be below one function symbol of arity k larger or equal to 2: f. Next, the terms $s_1, \ldots, s_n$ as well as the remaining part $Z_1, \ldots, Z_n$ of the contexts have to fit in some arguments of f. By maximality of Z_0, the number m of arguments of f in which $Z_j(s_j)$ fit is larger or equal to 2. It is also smaller than n. It needs not be equal to n as some of the Z_j's may share larger contexts (or even be a prefix of each other). Then Φ is the mapping which associates each $Z_j(s_j)$ with the argument of f in which it fits. When two $Z_j(s_j)$ take place in the same argument, we have a possible new application of the merge rule on $x_{\Phi(j)} = Z_j(s_j) \wedge x_{\Phi(j')} = Z_{j'}(s_{j'})$ (because $\Phi(j) = \Phi(j')$ in this case).

It is possible to design a simpler rule which only reduces two equations $x = X_j(s_j)$. Applying $n-1$ times such a rule, we would also get some similar result. However, using such a presentation, we would have problem in the control design. For example, considering all equations whose one member is x at the same time, allows to consider the *maximal shared context* Z_0 which is not possible considering each pair of equations successively. Actually, considering all equations $x = X_j(s_j)$ at once is necessary for termination purposes.

LEMMA 3.1. *All rules given in figures 3 and 4 preserve the set of solutions.*

PROOF. The lemma is straightforward, except for the last merge rule of figure 4 for which we have to decipher the target formula. First, let us consider the easy direction: let σ be a solution of the target formula. Let θ be an assignment to the existentially quantified variables such that $\sigma\theta$ is a solution of the unquantified formula. $\sigma\theta$ is a solution of one of the conjunctions:

(i) If $\sigma\theta$ is a solution of $x = X_i(s_i) \wedge \bigwedge_{j \neq i} s_i = Z_j(s_j) \wedge X_j = X_i \cdot Z_j$, then, for all $j \neq i$,

$$x\sigma \equiv x\sigma\theta \equiv X_i\sigma(s_i\sigma) \equiv X_i\sigma(Z_j\theta(s_j\sigma)) \equiv X_j\sigma(s_j\sigma)$$

using successively $x = X_i(s_i), s_i = Z_j(s_j)$ and $X_j = X_i \cdot Z_j$. Thus, for all $k \in \{1, \ldots, n\}$, $x\sigma \equiv X_k\sigma(s_k\sigma)$.

(ii) Assume now that $\sigma\theta$ is a solution of

$$\begin{array}{ll}
\bigwedge_{\substack{1 \leq j \leq n \\ \{j_1,\ldots,j_m\} \subseteq \mathrm{Inv}(\Phi,\{1,\ldots,n\})}} & X_j = Z_0(f(x_1,\ldots,x_k)[Z_{j_1}(s_{j_1})]_{i_{\Phi(j_1)}} \ldots [Z_{j_m}(s_{j_m})]_{i_{\Phi(j_m)}}[Z_j]_{i_{\Phi(j)}}) \\
\wedge \bigwedge_{j=1}^{n} x_{\Phi(j)} = Z_j(s_j) & \\
\wedge\, x = Z_0(f(x_1,\ldots,x_k)) &
\end{array}$$

for some $n \geq m \geq 2$, Φ surjective from $\{1,\ldots,n\}$ into $\{1,\ldots,m\}$, $f \in F$ whose arity k is larger than m, some subset $\{i_1,\ldots,i_m\}$ of $\{1,\ldots,k\}$. Then, using backward the two last families of equations:

$$\begin{array}{rcl}
x\sigma & \equiv & Z_0\theta(f(x_1\theta,\ldots,x_k\theta)) \\
 & \equiv & Z_0\theta(f(x_1\theta,\ldots,x_k\theta)[Z_{j_1}\theta(s_{j_1}\sigma)]_{i_{\Phi(j_1)}} \ldots [Z_{j_m}\theta(s_{j_m}\sigma)]_{i_{\Phi(j_m)}})
\end{array}$$

Now, for every j, $Z_j\theta(s_j\sigma) \equiv Z_{\widetilde{j}}\theta(s_{\widetilde{j}}\sigma)$ where $\widetilde{j} \in \{j_1,\ldots,j_m\}$ is the representative of j modulo $\sim$ (see the definition of Inv). Hence, for every j, it is possible to replace $Z_{\widetilde{j}}(s_{\widetilde{j}})$ with $Z_j(s_j)$. Moreover we have, by definition, $i_{\Phi(j)} = i_{\Phi(\widetilde{j})}$:

$$\begin{array}{rcl}
x\sigma & \equiv & Z_0\theta(f(x_1\theta,\ldots,x_k\theta)[Z_{j_1}\theta(s_{j_1}\sigma)]_{i_{\Phi(j_1)}} \ldots [Z_{j_m}\theta(s_{j_m}\sigma)]_{i_{\Phi(j_m)}}[Z_j\theta(s_j\sigma)]_{i_{\Phi(j)}}) \\
 & \equiv & X_j\sigma(s_j\sigma)
\end{array}$$

using the first conjunction.

Consider now the other direction: let σ be a solution of $x = X_1(s_1) \wedge \ldots \wedge x = X_n(s_n)$. Let $X_1\sigma \equiv t_1[\circ]_{p_1},\ldots,X_n\sigma \equiv t_n[\circ]_{p_n}$. Let p_0 be the greatest common prefix of $p_1,\ldots,p_n$. $t_i[\circ]_{p_0}$ should be independent of i since $t_1[s_1\sigma]_{p_1} \equiv \ldots \equiv t_n[s_n\sigma]_{p_n}$. Let $t_0[\circ]_{p_0}$ be this common value.

First case: p_0 is one of the p_i's
In this case, each t_j, $i \neq j$ can be written $t_j[\circ]_{p_j} \equiv t_0[u_j[\circ]_{q_j}]_{p_0}$ where $s_i\sigma \equiv u_j[s_j\sigma]_{q_j}$. Now, put $\theta = \{Z_j \mapsto u_j[\circ]_{q_j} \mid j \neq i\}$ (and choose any value for Z_i), and we have a solution $\sigma\theta$ of $x = X_i(s_i) \wedge \bigwedge_{j \neq i} s_i = Z_j(s_j) \wedge X_j = X_i \cdot Z_j$.

Second case: p_0 is distinct from all p_i's
Let $p_i = p_0 \cdot p'_i$ for all i and $t_0(p_0) = f$. None of the p'_i's is ϵ. Let $p'_i = \gamma_i \cdot q_i$ with $\gamma_i \in \{1,\ldots,k\}$ and k is the arity of f. By maximality of p_0, at least two p'_i's do not have any non-empty common prefix: for at least two indices i_1 and i_2, $\gamma_{i_1} \neq \gamma_{i_2}$. Therefore, k should be larger or equal to 2 and actually, $k \geq m = |\{\gamma_1,\ldots,\gamma_n\}| \geq 2$. We call Φ the mapping which associates to each $i \in \{1,\ldots,n\}$ an index $\Phi(i) \in \{1,\ldots,m\}$ such that $\gamma_{i_1} < \gamma_{i_2}$ iff $\Phi(i_1) < \Phi(i_2)$. For $1 \leq l \leq n$, $j_{\Phi(l)} = \gamma_l$. Now, let $\alpha = \{x_1 \mapsto x\sigma|_{p_0\cdot 1},\ldots x_k \mapsto x\sigma|_{p_0\cdot k}\}$ and, for all j, $t_j[\cdot]_{p_j} \equiv t_0[f(x_1\alpha,\ldots,x_k\alpha)[u_j[\circ]_{q_j}]_{i_{\Phi(j)}}]_{p_0}$. We define $Z_0\theta \equiv t_0[\circ]_{p_0}$, for all j, $Z_j\theta \equiv u_j[\circ]_{q_j}$ and $\theta|_{\{x_1,\ldots,x_k\}} = \alpha$. Now,

$$X_j\sigma \equiv t_j[\circ]_{p_j} \equiv t_0[f(x_1\theta,\ldots,x_k\theta)[u_{j_1}[s_{j_1}\sigma]_{q_{j_1}}]_{i_{\Phi(j_1)}} \ldots [u_{j_m}[s_{j_m}\sigma]_{q_{j_m}}]_{i_{\Phi(j_m)}}[u_j[\circ]_{q_j}]_{i_{\Phi(j)}}]_{p_0}$$

since $x_l\alpha \equiv u_\lambda[s_\lambda\sigma]_{q_\lambda}$ when $l = i_{\Phi(\lambda)}$. Hence, $\sigma\theta$ is a solution of all equations

$$X_j = Z_0(f(x_1,\ldots,x_k)[Z_{j_1}(s_{j_1})]_{i_{\Phi(j_1)}} \ldots [Z_{j_m}(s_{j_m})]_{i_{\Phi(j_m)}}[Z_j]_{i_{\Phi(j)}})$$

For the same reason, $\alpha\theta$ is a solution of all equations $x_{i_q} = Z_j(s_j)$. Hence, σ is a solution of the target formula. □

LEMMA 3.2. *The rules given in figures 3 and 4 do transform S_0-unification problems into S_0-unification problems.*

PROOF. The lemma is obvious for the rules of figure 3 and the first rule of figure 4 since these rules only modify the membership part of the formula. Now, for the last three rules of figure 4, assume that they are applied on some unification problem ϕ, resulting in a formula ψ.

Condition 2 is satisfied since no equations in the premisses of the rules are solved equations. Note moreover that, in the last merge rules, the equations $x = X_i(s), X_j = X_i \cdot Z_j$ on one hand and the equations $X_j = Z_0(E[Z_j]_{i_{\Phi(j)}})$ on the other hand are solved in ψ.

It is straightforward to check that condition 3 is satisfied.

Consider now a (new) equation $s = Y(t)$ in ψ where s is not a variable. It is easy to check that t does not contain any context variable. We have also to check that Y does not occur in any other unsolved equation. For all rules, this is a consequence of this same property for ϕ, except concerning the last merge rule. The only difficulty with this rule is to check what are the indices of all context variables. The possibly unsolved equations can be found in the conjunction $\bigwedge_{j=1}^{n} x_{\Phi(j)} = Z_j(s_j)$. But in this conjunction, each Z_j occurs only once.

New context variables (introduced by the rule) are not concerned with property 5. On the other hand, for all context variable occurring in an equation of ϕ, $\mathrm{Arg}_\psi(X) \subseteq \mathrm{Arg}_\phi(X)$, hence property 5.

Concerning property 6, only new context variables that occur twice in ψ have to be considered. This may only happen in the last merge rule. First, Z_0 occurs twice and $\mathrm{Arg}_\psi(Z_0) = \{f(x_1, \ldots, x_k)\} \bigcup_{i=1}^{n} \{E[Z_j(t)]_{i_{\Phi(j)}} \mid t \in \mathrm{Arg}_\psi(X_j)\}$. Let σ be a solution of the equational part of ψ. We have to show that $f(x_1\sigma, ..., x_n\sigma) \equiv E[Z_j(t)]_{i_{\Phi(j)}}\sigma$ for all j and all $t \in \mathrm{Arg}_\psi(X_j)$. But this is a consequence of the correctness lemma as $x\sigma \equiv X_j(s_j)\sigma$. Now, we also have to consider variables Z_j which occur twice. This case is simple: Z_j has always s_j as argument: $\mathrm{Arg}_\psi(Z_j) = \{s_j\} \cup \mathrm{Arg}_\psi(X_j) = \mathrm{Arg}_\phi(X_j)$. Hence property 6 follows from the same property on ϕ. □

4. Other Rules

Other transformation rules are much simpler. Clash, Occur Check and Coalesce rules are described in figure 6. Other simplification rules are described in figure 7.

The two following lemmas are straightforward.

LEMMA 4.1. *All the rules given in figures 7 and 6 do transform S_0-unification problems into S_0-unification problems.*

LEMMA 4.2. *All the rules given in figures 7 and 6 do preserve the set of solutions.*

Clash

$$\frac{f(s_1,\ldots,s_n) = g(t_1,\ldots,t_m)}{\bot} \quad \text{if } f \neq g$$

Occur Check

$$\frac{\phi}{\bot} \quad \text{If there is a variable } x \text{ occurring in } \phi \text{ such that } x >_\phi x$$

Coalesce

$$\frac{x = y \wedge P}{x = y \wedge P\{x \mapsto y\}} \quad \text{If } x, y \text{ are two distinct variables which occur both in } P$$

In order to make the termination proof easier, we also assume for the **Coalesce** rule that x is almost solved whenever y is almost solved.

Figure 6. Clash, Occur-Check and Coalesce rules

$$\frac{x \in q}{\bot} \text{ If } [\![q]\!]_{\mathcal{A}} = \emptyset \qquad \frac{X \in C}{\bot} \text{ If } [\![C]\!]_{\mathcal{A}} = \emptyset$$

$$\frac{\exists x : x = t \wedge P}{P} \text{ If } x \notin Var(t, P) \qquad \frac{\exists X : X = U \wedge P}{P} \text{ If } X \notin Var(U, P)$$

$$\frac{\exists x : x \in q \wedge P}{P} \quad \text{If } x \notin Var(P) \text{ and } [\![q]\!]_{\mathcal{A}} \neq \emptyset$$

$$\frac{\exists X : X \in C \wedge P}{P} \quad \text{If } X \notin Var(P) \text{ and } [\![C]\!]_{\mathcal{A}} \neq \emptyset$$

Figure 7. Removing useless parts of a constraint

5. Termination

This is the main result of our paper: the set of rule is terminating and the irreducible formulas are solved forms.

THEOREM 5.1. *The rules of figures 2, 6, 3, 4 and 7 do terminate on unification problems.*

PROOF. Let us proceed by contradiction and assume an infinite sequence of transformations $\phi_1 \vdash \ldots \vdash \phi_n \vdash \ldots$ where all ϕ_i are assumed to be normalized according to the rules of figure 1. If there is such an infinite sequence, then there is an infinite sequence $\psi_1, \ldots, \psi_n, \ldots$ consisting only of existentially quantified conjunctions of atomic formulae and such that $\psi_i \vdash \psi_{i+1} \vee D_i$ for some D_i. Indeed, to see that, it is enough to consider the derivation $\phi_1 \vdash \ldots \phi_n$ as a tree T_n whose nodes are labeled with existentially quantified conjunctions of atomic formulae. (It is actually an *or* tree). The tree is computed as follows: initially, if $\phi_1 \equiv \bigvee_{j=1}^n \omega_j$, T_1 contains a root labeled with $\top$ and n sons labeled with $\omega_1, \ldots, \omega_n$ respectively. Now, at each reduction step, ϕ_i is the disjunction of all labels of the leaves of T_i. Since each reduction step only involves a conjunction ω of atomic formulae, assume $\omega \vdash \bigvee_{i=1}^m \omega'_j$, then the next tree T_{i+1} is obtained from T_i by growing m sons from ω, respectively labeled with $\omega'_1, \ldots, \omega'_m$. If the limit tree T_∞ is infinite, it should contain an infinite path since the branching is always finite. The sequence $\psi_1, \ldots$ is the sequence of labels of that path. Hence, from now on, we assume that ϕ_i is an existentially quantified conjunction of atomic formulae and ϕ_{i+1} is one element of the disjunction obtained by one reduction step from ϕ_i. Furthermore, we may assume that no positive occur check is ever applied in the sequence ϕ (otherwise the sequence is finite).

The idea of the proof is simple: we interpret the ϕ_i's in an algebra of (ground) terms which contains the AC symbol $\wedge$ and which is ordered using the associative path ordering (see (Bachmair & Plaisted, 1985); actually, we only need here a recursive path ordering on flatten terms since only one AC function symbol is involved). The target alphabet G contains F, $\wedge$, $=_1$ (for the equality between first-order terms), $=_2$ (for the equality of second order terms), the unary function symbol $\in$ and the new symbols a, b, h, k, g_1, g_2. The interpretation itself uses the infinite sequence, through the *limit occur-check relation* $\geq_{occ}$ (which may be not an ordering) obtained as the (reflexive-transitive closure of the) union of all occur-check relations obtained along the sequence. $=_{occ}$ is $\geq_{occ} \cap \leq_{occ}$ and $>_{occ}$ is $\geq_{occ} \setminus =_{occ}$. Of course, this relation needs not be computed, it is only used for the purpose of this (non-constructive) proof. We show roughly that the interpretation of the sequence is strictly decreasing w.r.t. the associative path ordering, hence contradicting it's well-foundedness.

First of all, for every conjunction of atomic formulas ϕ, we do not consider the solved part of ϕ; we preprocess the formula, removing all solved parts. ϕ will be interpreted as $I_1(\phi, \phi)$ where I_1 and I_2 are defined as follows:

$$
\begin{array}{rcll}
I_1(t \in q, \phi) & = & \in (I_2(t, \phi)) & \\
I_1(c \in C, \phi) & = & \in (I_2(c, \phi)) & \\
I_1(P \wedge Q, \phi) & = & I_1(P, \phi) \wedge I_1(Q, \phi) & \\
I_1(\exists Y.P, \phi) & = & I_1(P, \phi) & \\
I_1(\exists x.P, \phi) & = & I_1(P, \phi) & \\
I_1(s = t, \phi) & = & I_1(s, \phi) =_1 I_1(t, \phi) & \text{if } s, t \in T(F, \mathcal{X}, \mathcal{CX}) \\
I_1(s = t, \phi) & = & I_1(s, \phi) =_2 I_1(t, \phi)) & \text{if } s, t \in C(F, \mathcal{X}, \mathcal{CX}) \\
I_j(f(s_1, \ldots, s_n), \phi) & = & f(I_j(s_1, \phi) \ldots, I_j(s_n, \phi)) & \text{for every } f \in F \text{ and } j = 1, 2 \\
I_j(x, \phi) & = & h^{2n(x,\phi)}(k^{m(x,\phi)}(a)) & \text{if } x \in \mathcal{X} \text{ is almost solved} \\
 & & & \text{and for } j = 1, 2 \\
I_j(x, \phi) & = & h^{2n(x,\phi)+1}(a) & \text{if } x \in \mathcal{X} \text{ is not almost solved} \\
 & & & \text{and for } j = 1, 2 \\
I_1(X(s), \phi) & = & g_1(b, I_1(s, \phi)) & \text{if } X \in \mathcal{CX} \text{ and } s \in T(F, \mathcal{X}, \mathcal{CX}) \\
I_2(X(s), \phi) & = & g_1(g_2^{N(X,\phi)}(b), (I(s, \phi))) & \text{if } X \in \mathcal{CX} \text{ and } s \in T(F, \mathcal{X}, \mathcal{CX}) \\
I_j(t[\circ]_p, \phi) & = & I_j(t[b]_p, \phi) & \text{for } j = 1, 2
\end{array}
$$

where

(i) $n(x, \phi)$ is the number of (first-order) variables in ϕ which are neither almost solved in ϕ nor larger than x w.r.t. $>_{occ}$

(ii) $m(x, \phi)$ is the number of occurrences of almost solved variables y such that $y =_{occ} x$.

(iii) $N(X, \phi)$ is the number of second order variables in ϕ which are not larger than X w.r.t $>_{occ}$.

We write simply $I_1(\psi)$ when the second argument is self evident.

For example, the formula

$$\exists Y.X = Y \cdot Z \wedge f(x) = Z(y) \wedge X \in C$$

is interpreted as

$$g_1(b,b) =_2 g_1(b, g_1(b,b)) \wedge f(h(a)) =_1 g_1(b, h(a)) \wedge \in (g_1(g_2^2(b), b))$$

if we assume that $\geq_{occ}$ is $\geq_\phi$.

The precedence on function symbols of the target signature is given by:

$$=_1 > =_2 > \left\{ \begin{array}{c} h > k \\ a \end{array} \right\} > g_1 > g_2 > f > b > \in > \wedge$$

for every $f \in F$. Moreover, g_1 is assumed to have a lexicographic status.

The problem with such an ordering is that the interpretation of a term is in general context-dependent. This is due to the components $n(x,\phi)$, $m(x,\phi)$ and $N(X,\phi)$ which depend on ϕ. In order to reduce the situations which are context sensitive, we have these two interpretations I_1, I_2. I_2 is the interpretation of terms *below a membership constraint*. On the other hand, $N(X,\phi)$ is only used in the definition of I_2. In other words, the context can affect the interpretation of $X \in \mathcal{CX}$, *only below a membership constraint*. Concerning first-order variables, the purpose of the nest few lemmas is to show that their interpretation is decreasing.

LEMMA 5.2. *If x is almost solved in ϕ then either it does not occur in ϕ_{i+1}, or it is solved in ϕ_{i+1}, or it is still almost solved in ϕ_{i+1}.*

PROOF. This can be checked on the rules: as soon as a variable x only occurs as a member of an equation or in a membership constraint, then it cannot be introduced inside any term of an equation, except via **Coalesce**. But we prevent this situation in the control on the rule. □

LEMMA 5.3. *If x is almost solved in ϕ and $x >_\phi y$ (resp. $x =_{occ} y$) then $x >_{occ} y$ (resp. $x =_{S,\phi} y$).*

PROOF. By contradiction, assume $\phi \vdash^* \psi_0 \vdash \psi$ and x is almost solved in ϕ (hence in ψ_0 and ψ thanks to lemma 5.2) and that $x >_{\psi_0} y$ (resp. $x \neq_{S,\psi_0} y$) and $x \leq_\psi y$ (resp. $x =_{S,\psi} y$). The step $\psi_0 \vdash \psi$ must create an equality $z = t[x]$ (resp. $z = x$) with $z =_{S,\psi_0} y$. Since x is almost solved in ψ_0 and ψ, the only possibility is to create an equality $z = x$ using the **Coalesce** rule. Which means that ψ_0 contains a conjunction $z' = z \wedge x = z'$ in which case $x =_{S,\psi_0} z =_{S,\psi_0,y}$, a contradiction. □

LEMMA 5.4. *if $\phi \vdash \psi$ and x is a variable occurring in both the unsolved part of ϕ and the unsolved part of ψ, then $n(x,\phi) \geq n(x,\psi)$.*

PROOF. Only the last merge rule introduces new first-order variables. Hence, for all rules but the last merge rule, $n(x,\phi) \geq n(x,\psi)$, thanks to lemma 5.2. Consider now the last merge rule. In the target formula, the variable x is solved and the variables $x_1, \ldots, x_k, y_1, \ldots, y_m$ are almost solved. Hence these new variables do not change the interpretation $n(z,\phi)$ of other first-order variables occurring in ϕ.

□

LEMMA 5.5. *If $\phi \vdash \psi$ then for every x occurring both in the unsolved part of ϕ and ψ, we have $I_j(x, \phi) \geq_{apo} I_j(x, \psi)$.*

PROOF. According to lemmas 5.4 and 5.2, we only have to investigate the case where x is almost solved in ψ, assuming $n(x, \phi) = n(x, \psi)$. There are two situations:

1 If x is not almost solved in ϕ, then $I_j(x, \phi) = h^{2n+1}(a)$ and $I_j(x, \psi) = h^{2n}(k^{m(x,\psi)}(a))$. Then $I_j(x, \phi) >_{apo} I_j(x, \psi)$ since $h > k$.

2 If x is almost solved in ϕ and ψ, we have to show that $m(x, \phi) \geq m(x, \psi)$. $n(x, \phi) = n(x, \psi)$ implies that every variable y which is not larger (w.r.t. $>_{occ}$) than x and which is not almost solved in ϕ is not almost solved in ψ. In particular, if $y =_{occ} x$ and y is almost solved in ψ and occurs in ϕ, then it is also almost solved in ϕ. Then, there are two cases where the number of almost solved variables that are equal to x (w.r.t. $=_{occ}$) can be modified:

Using the Coalesce rule. However, in such a case, the total number of occurrences of variables that are equal (w.r.t. $=_{occ}$) to x remains unchanged.

Using the last merge rule. In such a case, the only additional problem is that some new first-order variable could be both almost solved and equal (w.r.t. $=_{occ}$) to x. This might only happen when some context variable Z_j collapses to an empty context in the target of the rule, yielding a new equality $x_i = y$, which possibly increases $m(y, \phi)$. However, in such a situation, $x \neq_{S,\phi} y$ because of the control on the merge rule (**Coalesce** does not apply). This implies that $x \neq_{occ} y$ by lemma 5.3 and since x is almost solved in ϕ. Hence $x_i \neq_{occ} x$ and $m(x, \phi)$ cannot increase.

□

Let us add some further remarks before checking the decreasingness:

(i) Every non logical symbol is larger than $\wedge$. Therefore, to prove the decreasingness, it is sufficient to prove that, for every atomic formula in the target of the rule, there is some strictly larger atomic formula in the source of the rule. Moreover, since $=_1$ is the largest symbol in the precedence, it is again sufficient to prove that any equation between first order terms in the target of the rule is strictly smaller than some first order equation in the source of the rule.

(ii) Finally, we have to consider in principle the last normalization rule. i.e. we have to consider that there is a membership constraint $X \in \neg\epsilon$ each time a new variable $X \in \mathcal{CX}$ is introduced. But it is never a problem as $X \in \neg\epsilon$ does not change the interpretation of the variables and its own interpretation is always smaller than one of the atomic formulas in the premisse which disappears during the transformation.

We give below, for each rule, the reason for decreasingness. Each time, we have to check whether the interpretation of variables has changed or not. In each case, we have then to use the definition of the associative path ordering to check the decreasingness.

Coalesce rule

Assume that $\phi \equiv \exists \vec{x}. x = y \wedge P \vdash \psi \equiv \exists \vec{x}. x = y \wedge P\{x \mapsto y\}$. According to lemma

5.5, we may assume that for all first-order variables z but x, $I_j(z,\phi) = I_j(z,\psi)$. Then $I_1(\psi,\psi) = I_1(P,\psi) < I_1(\phi,\phi)$ by embedding.

Decomposition rules

We will consider the last decomposition rule later on. Consider the other decomposition rules. Let $\phi \vdash \psi$ be one transformation (on existentially quantified conjunctions of atomic formulae) using this rule. Thanks to lemma 5.5, we have only to prove the decreasingness of the rules assuming that all first-order variables have an unchanged interpretation. In such a situation, for every atomic formula A, $I_1(A,\phi) = I_1(A,\psi)$ and will be simply written $I_1(A)$. Then we have $I_1(t_i = u_i) < I_1(f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n))$ for all i, $I_1(t_i \in q_i) < I_1(f(t_1,\ldots,t_n) \in q)$ for all i, $I_1(U \in C) < I_1(f(t_1,\ldots,t_n)[U]_i \in C)$, $I_1(t_j \in q_j) < I_1(f(t_1,\ldots,t_n)[U]_i \in C)$ for all $i \neq j$, $I_1(s \in q') < I_1(X(s) \in q)$, $I_1(u \in C_1) < I_1(X(u) \in C)$, $I_1(X \in C_0) < I_1(X(u) \in C)$ by the subterm property of the associative path ordering. It follows that the interpretation is decreasing for the 5 first decomposition rules.

The last rule of figure 2

The rule may change the interpretation of X, as well as the interpretation of other context variables. But only I_2 might change, which means that only the interpretation of some membership constraints may increase. Then, as remarked above, because $=_1$ is maximal, it is sufficient to prove that $f(s_1,\ldots,s_n) = X(s)$ is larger than any equation between first order terms in the target of the rule.

Moreover,X does not occur in s nor in $f(s_1,\ldots,s_n)$ which means that these terms do have the same (or a smaller) interpretation in the source and in the target formula. Finally, the interpretation of Y in the target formula should be smaller (or equal to) the interpretation of X in the source formula. Then it follows from the subterm property of the associative path ordering that the interpretation is strictly decreasing.

Clash and Occur-Check and the rules of figure 7

The decreasingness is straightforward.

Merge rules (except the last one)

The merge rules consist either in erasing some atomic formula, leading to a straightforward decreasingness, or else to a replacement of a variable with a non-variable term. The interpretations of the variables can be assumed to be unchanged thanks to lemma 5.5 (and we do not report below in which formula they are interpreted). Then the decreasingness follows the inequality between the interpretation of the variable and the interpretation of the term. Let us study this in detail.

First, let us recall that the merge rules are only applied on variables that are almost solved (thanks to the control) and hence, which will be always almost solved (or solved). This means that $x >_{occ} y$ whenever $x >_{\phi} y$, thanks to lemma 5.3.

If $\phi[x = s \wedge x \in q] \vdash \phi[x = s \wedge s \in q] \equiv \psi$, s is not a variable, hence, for every variable y of s, $x >_{occ} y$, which means that $I_j(x) > I_j(s)$ (for $j = 1,2$), thanks to the definition of I_j. Indeed, every functional symbol is smaller than h and $n(x,\phi)$ is strictly larger than $I_j(y)$, for every variable y in s since there is no positive occur-check. Now, $I_1(x = s \wedge x \in q) > I_1(x = s \wedge s \in q)$.

The situation is similar for the rules $x = f(s_1,\ldots,s_m) \wedge x = g(t_1,\ldots,t_n) \vdash x = f(s_1,\ldots,s_m) \wedge f(s_1,\ldots,s_m) = g(t_1,\ldots,t_n)$ and $x = f(s_1,\ldots,s_m) \wedge x = X(t) \vdash x = f(s_1,\ldots,s_m) \wedge f(s_1,\ldots,s_m) = X(t)$. $I(x) > I(f(s_1,\ldots,s_m))$, since, as above,

$I(x) > I(y)$ for every variable occurring in $s_1, \ldots, s_n$. (We use again the fact that x is maximal and that there is no positive occur-check).
It remains only to consider the third merge rule of figure 3, consisting in replacing an occurrence of X with a second order term $t[Y]_p$. Because of the properties of unification problems, there cannot be any positive occur-check on second order terms. Therefore, $N(Y, \phi)$ is strictly smaller than $N(X, \phi)$, both in the source and in the target formula. Now, since g_2 is larger than any symbol in F, $I_2(X) > I_2(t[Y]_p)$, which proves the decreasingness.

The last Merge Rule

Assume that $\phi \vdash \psi$ using this rule. If ψ is obtained on a branch corresponding to the first disjunction in the target of the rule, then no new first-order variable is introduced and the decreasingness is obtained in a way similar to the above one. Assume now that ψ is obtained on a branch stemming from the second disjunction in the target of the rule.
The maximality of x and the fact that the formula is irreducible w.r.t. all other rules ensure that all occurrences of x are either in the solved part of ϕ or as a member of an equation of the form $x = X_i(s_i)$. (This will be useful for completeness). Moreover, x is solved after applying the rule and all new first-order variables $x_1, \ldots, x_k$ are almost solved in the target formula. By lemma 5.5, $I_j(z, \phi) \geq I_j(z, \psi)$.
Now, concerning context variables, the introduction of new variables may change their interpretation, but, as already noticed, this doesn't matter as soon as the conjunction of equations between first-order terms in the premisse is strictly larger than the conjunction of equations between first-order terms in the conclusion.
Now, we prove that all equations $x_{i_{\Phi(j)}} = Z_j(s_j)$ have a strictly smaller interpretation than some $x = X_i(s_i)$. Note first that $n(x, \phi) \geq n(x_{i_{\Phi(j)}}, \psi)$ for all j since the x_i's are almost solved and $x >_{occ} x_i$.
We have to investigate a number of situations, depending on which of the variables Z_j's "collapse" to an empty context using the last normalization rule.

1 If, for some j such that $i_{\Phi(j)} = k$, Z_j collapses to an empty context and s_j becomes an almost solved variable. Then $n(x_k, \psi) < n(x, \phi)$ since s_j was not almost solved and not greater than x, hence counted in $n(x, \phi)$, but no longer in $n(x_k, \psi)$. Hence every equation $x_k = u$ is strictly smaller than any equation $x = X_j(s_j)$.

2 If Z_j collapses to an empty context and s_j is variable such that $s_j =_{S,\psi} z$ and z is a variable of ϕ which is almost solved. Then, again, $n(x, \phi) > n(x_{i_{\Phi(j)}}, \psi)$ for the same reason.
We assume in what follows that k is such that, for every j such that $i_{\Phi(j)} = k$, either Z_j does not collapse or s_j is not equal (modulo $=_{S,\psi}$) to an almost solved variable which occurs in ϕ.

3 If Z_j collapses to an empty context, then the number of occurrences of an almost solved variable which is equal (modulo $=_{occ}$) to x_k is bounded by the number of occurrences of some x_i in ψ (thanks to lemma 5.3). Then $m(x_k, \psi)$, is itself bounded by $n = m(x, \phi)$. Hence $I_1(x_{i_{\Phi(j)}}, \psi) \leq I_1(x, \phi)$ and $I_1(s_j, \psi) < I_1(X_j(s_j), \psi)$, hence $I_1(x_k = s_j, \psi) < I_1(x = X_j(s_j), \phi)$.

4 If, for any j such that $i_{\Phi(j)} = k$, Z_j does not collapse, then $m(x_k, \psi)$ is the number of occurrences of x_k itself which is bounded by $|\Phi^{-1}(j)|$, which is itself strictly smaller than $m(x, \phi)$ since m is assumed to be larger than (or equal

to) 2. It follows that $I_j(x_k,\psi) < I_j(x,\psi)$ and therefore $I_1(x_k = Z_j(s_j),\psi) < I_1(x = X_j(s_j),\phi)$.

It follows that any equation between first order terms in the target formula is strictly smaller than some equation between first order terms in the source formulas, which is sufficient to ensure the strict decreasingness, as noticed before. □

6. A complete example of unification problem solving

Let us show now an example of the unification process.

EXAMPLE 6.1. *We consider the unification problem of example 1.5*

$$\begin{aligned}\exists y, x_1, x_2, X_1, X_2, X_3.\ & x_1 = X(x) \wedge x_2 = Y(f(z,x)) \wedge f(x_1,x_2) = f(f(z,a),x_1) \\ & \wedge f(x,y) = x_2 \wedge X_1 = f(a,X_2) \wedge X_2 = f(x_1,X_3) \wedge X_1 \in C \wedge z \in q_i\end{aligned}$$

together with the interpretation given by the automaton $\mathcal{A}$ of example 1.3.

We should first non-deterministically guess which variables of $\mathcal{CX}$ are assigned to the empty context. However, for sake of simplicity, we will not commit ourself now and postpone this guess until it becomes necessary for an occur-check. This does not apply to introduced variables for which we will indeed guess whether they are empty or not.

Decomposing the equation $f(x_1,x_2) = f(f(z,a),x_1)$ we get $x_1 = f(z,a) \wedge x_2 = x_1$. Using the coalesce rule, we get now the problem

$$\begin{aligned}\exists y, x_1, x_2, X_1, X_2, X_3.\ & x_1 = X(x) \wedge x_1 = Y(f(z,x)) \wedge x_1 = x_2 \wedge x_1 = f(z,a) \\ & \wedge x_1 = f(x,y) \wedge X_1 = f(a,X_2) \wedge X_2 = f(x_1,X_3) \wedge X_1 \in C \wedge z \in q_i\end{aligned}$$

Merging $x_1 = f(z,a)$ and $x_1 = f(x,y)$, and decomposing the resulting equation, we get $x_1 = f(z,a) \wedge z = x \wedge a = y$. Replacing everywhere z with x we get

$$\begin{aligned}\exists y, x_1, x_2, X_1, X_2, X_3.\ & x_1 = X(x) \wedge x_1 = Y(f(x,x)) \wedge x_1 = x_2 \wedge x_1 = f(x,a) \wedge z = x \\ & \wedge y = a \wedge X_1 = f(a,X_2) \wedge X_2 = f(x_1,X_3) \wedge X_1 \in C \wedge x \in q_i\end{aligned}$$

Merging $x_1 = X(x)$ and $x_1 = f(x,a)$ we get $x_1 = f(x,a) \wedge X(x) = f(x,a)$, which in turn leads to $\exists Z.(X = f(x,\circ) \wedge x = a) \vee (X = f(\circ,a) \wedge x = x) \vee (X = f(x,Z) \wedge x = Z(a) \wedge Z \in \neg\epsilon) \vee (X = f(Z,a) \wedge x = Z(x) \wedge Z \in \neg\epsilon)$. This gives rise to 4 new unification problems. The last one reduces to $\perp$ because of the positive occur-check. The three other formulae lead (after some cleaning using the rules of figure 7) to respectively ϕ_0, ϕ_1 and ϕ_2:

$$\begin{aligned}\phi_0 \equiv \exists x_1, X_1, X_2, X_3.\ & X = f(x,\circ) \wedge x_1 = Y(f(x,x)) \wedge x_1 = f(x,a) \wedge x = a \wedge z = x \\ & \wedge X_1 = f(a,X_2) \wedge X_2 = f(x_1,X_3) \wedge X_1 \in C \wedge x \in q_i \\ \phi_1 \equiv \exists x_1, X_1, X_2, X_3.\ & X = f(\circ,a) \wedge x_1 = Y(f(x,x)) \wedge x_1 = f(x,a) \wedge z = x \\ & \wedge X_1 = f(a,X_2) \wedge X_2 = f(x_1,X_3) \wedge X_1 \in C \wedge x \in q_i \\ \phi_2 \equiv \exists x_1, X_1, X_2, X_3, Z.\ & X = f(x,Z) \wedge x = Z(a) \wedge x_1 = Y(f(x,x)) \wedge x_1 = f(x,a) \\ & \wedge z = x \wedge X_1 = f(a,X_2) \wedge X_2 = f(x_1,X_3) \wedge X_1 \in C \wedge x \in q_i \wedge Z \in \neg\epsilon\end{aligned}$$

In ϕ_0, $x = a \wedge x \in q_i$ reduces by merge to $x = a \wedge a \in q_i$ which reduces in turn to $x = a$ since there is a rule $a \to q_i$ in $\mathcal{A}$. Now, in the three formulas, we have again to merge and decompose the conjunction $x_1 = Y(f(x,x)) \wedge x_1 = f(x,a)$. All resulting problems lead to a positive occur-check, except when Y is an empty context, in which case we get $Y \in \epsilon \wedge x_1 = f(x,a) \wedge x = a$. Then merging in ϕ_2 the equations $x = a$ and $x = Z(a)$ we get a contradiction. Merging in ϕ_1 $x = a$ and $x \in q_i$, we get $x = a$ after decomposition.

Finally from $\phi_0 \vee \phi_1 \vee \phi_2$ we only get two formulas which only differ in the value of X; let

$$\begin{aligned}\phi_3 \equiv \exists x_1, X_1, X_2, X_3.\, & Y \in \epsilon \wedge x = a \wedge x_1 = f(x, a) \wedge z = x \wedge X_1 = f(a, X_2)\\ & \wedge X_2 = f(x_1, X_3) \wedge X_1 \in C\end{aligned}$$

We may only reduce ϕ_3, keeping in mind that the $\phi_0 \vee \phi_1$ reduced to $\phi_3 \wedge (X = f(x, \circ) \vee X = f(\circ, a))$, up to distributivity and existential quantifiers rearrangement.

Now, we have to simplify the expression $X_1 = f(a, X_2) \wedge X_2 = f(x_1, X_3) \wedge X_1 \in C$. Let us first merge $X_1 = f(a, X_2)$ and $X_1 \in C$. We assume that C is given by the expression of 1.3, or more explicitly, by the state q_p^Ω in the automaton of example 2.3:

$$\begin{array}{rclrcl} f(q_i, q_i^\Omega) & \to & q_p^\Omega & f(q_i^\Omega, q_i) & \to & q_p^\Omega \\ f(q_p^\Omega, q_p) & \to & q_p^\Omega & f(q_p, q_p^\Omega) & \to & q_p^\Omega \\ f(q_i^\Omega, q_p) & \to & q_i^\Omega & f(q_p, q_i^\Omega) & \to & q_i^\Omega \\ f(q_p^\Omega, q_i) & \to & q_i^\Omega & f(q_i, q_p^\Omega) & \to & q_i^\Omega \\ \Omega & \to & q_p^\Omega & & & \end{array}$$

The result of the merge: $X_1 = f(a, X_2) \wedge f(a, X_2) \in C$ is followed by a decomposition of the membership constraint: $f(a, X_2) \in C$ becomes $(a \in q_i \wedge X_2 \in q_i^\Omega) \vee (a \in q_p \wedge X_2 \in q_p^\Omega)$. (As before, q_p^Ω and q_i^Ω could be replaced with context expressions, but we keep these notations for sake of simplicity).

Now, decomposing $a \in q_p$ we get $\perp$ (there is no rule $a \to q_p$ in the automaton $\mathcal{A}$) and decomposing $a \in q_i$, we get $\top$. Finally, $f(a, X_2) \in C$ is decomposed into $X_2 \in q_i^\Omega$. Again, this constraint is merged with $X_2 = f(x_1, X_3)$ and the membership constraint is decomposed, which gives finally: $(x_1 \in q_i \wedge X_3 \in q_p^\Omega) \vee (x_1 \in q_p \wedge X_3 \in q_i^\Omega)$. At this point we have to consider two problems corresponding to the two cases in the disjunction. We have now the two problems:

$$\begin{aligned}\phi_4 \equiv \exists x_1, X_1, X_2, X_3.\, & Y \in \epsilon \wedge x = a \wedge x_1 = f(x, a) \wedge z = x \wedge X_1 = f(a, X_2)\\ & \wedge X_2 = f(x_1, X_3) \wedge X_3 \in q_i^\Omega \wedge x_1 \in q_p\\ \phi_5 \equiv \exists x_1, X_1, X_2, X_3.\, & Y \in \epsilon \wedge x_1 = f(x, a) \wedge z = x \wedge X_1 = f(a, X_2)\\ & \wedge X_2 = f(x_1, X_3) \wedge X_3 \in q_p^\Omega \wedge x_1 \in q_i\end{aligned}$$

We have now to merge $x_1 = f(x, a)$ and $x_1 \in q_i$ (resp. $x_1 \in q_p$). This yields the single conjunction $x_1 = f(x, a) \wedge x \in q_p$ (resp. $x_1 = f(x, a) \wedge x \in q_i$) since $a \notin [\![q_p]\!]_{\mathcal{A}}$. Now, in both ϕ_4 and ϕ_5 we can merge the new membership constraint with $x = a$. For ϕ_4 we get $\top$ and ϕ_5 is reduced to $\perp$ again because $a \notin [\![q_p]\!]$. Hence, again, the disjunction boils down to a single problem:

$$\begin{aligned}\exists x_1, X_1, X_2, X_3.\, & Y \in \epsilon \wedge x = a \wedge x_1 = f(x, a) \wedge z = x \wedge X_1 = f(a, X_2)\\ & \wedge X_2 = f(x_1, X_3) \wedge X_3 \in q_i^\Omega\end{aligned}$$

Which is now irreducible. Which means that the original problem reduces to the two following problems:

$$\begin{aligned}\psi_1 \equiv \exists x_1, X_1, X_2, X_3.\, & X = f(x, \circ) \wedge Y \in \epsilon \wedge x = a \wedge x_1 = f(x, a) \wedge z = x \wedge X_1 = f(a, X_2)\\ & \wedge X_2 = f(x_1, X_3) \wedge X_3 \in q_i^\Omega\\ \psi_2 \equiv \exists x_1, X_1, X_2, X_3.\, & X = f(\circ, a) \wedge Y \in \epsilon \wedge x = a \wedge x_1 = f(x, a) \wedge z = x \wedge X_1 = f(a, X_2)\\ & \wedge X_2 = f(x_1, X_3) \wedge X_3 \in q_i^\Omega\end{aligned}$$

As we will see, both problems are solved forms of our unification problem and hence the original problem has a solution.

7. Completeness

We now have to investigate what are the irreducible formulas.

DEFINITION 7.1. *Given an admissible sequence S_0, a formula $\phi \in \mathcal{F}$ is an S_0*-DAG solved form *if it is either $\top$, $\perp$, or else it is a S_0-unification problem of the following form:*

$$\exists \vec{z}, \exists \vec{Z} : x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge y_1 \in q_1 \wedge \ldots \wedge y_m \in q_m \wedge Y_1 \in C_1 \wedge \ldots \wedge Y_k \in C_k$$

where

(i) $x_1, \ldots, x_n \in \mathcal{X} \cup \mathcal{CX}$ *and, for all i, x_i does not occur in* $x_{i+1} = t_{i+1} \wedge \ldots \wedge Y_k \in C_k$

(ii) $t_1, \ldots, t_n \in T(F, \mathcal{X}, \mathcal{CX}) \cup GC(F, \mathcal{X}, \mathcal{CX})$

(iii) $y_1, \ldots, y_m \in \mathcal{X}$ *are distinct variables*

(iv) $Y_1, \ldots, Y_k \in \mathcal{CX}$ *are distinct variables*

(v) $[\![q_1]\!]_{\mathcal{A}}, \ldots, [\![q_m]\!]_{\mathcal{A}}, [\![C_1]\!]_{\mathcal{A}}, \ldots, [\![C_k]\!]_{\mathcal{A}}$ *are not empty.*

A DAG solved form ϕ consists of a *equational part* ϕ_{eq} and a membership constraint ϕ_m. First, we show that this notion of solved forms makes sense: every solved form is solvable.

LEMMA 7.2. *DAG solved forms which are distinct from $\perp$ are satisfiable.*

PROOF. Since $y_1, \ldots, y_m$, $Y_1, \ldots, Y_k$ are respectively distinct and since $[\![q_1]\!]_{\mathcal{A}}, \ldots [\![q_m]\!]_{\mathcal{A}}$, $[\![C_1]\!]_{\mathcal{A}}, \ldots [\![C_k]\!]_{\mathcal{A}}$ are not empty, there is an assignment τ which is a solution of ϕ_m. Now, $\phi_{eq}\tau$ is solved, by the other properties of DAG solved forms. It follows that $\sigma_{e\tau}$ (as defined in section 3.1) is a solution of the formula ϕ (where the existential quantifiers have been discarded). □

THEOREM 7.3. *Every formula, which is irreducible with respect to the rules of figures 2, 6, 3, 4 and 7 is a finite disjunction of DAG solved forms.*

PROOF. The decomposition rules investigate all possible situations of equations between non-variable terms or membership constraints with a non-variable member. Therefore, atomic formulas in an irreducible S_0-unification problem must only involve equations whose one member is a variable and membership constraints whose one member is a variable.

In an irreducible formula, there cannot be equations $x = y$ where both x and y occur in the unsolved part of the problem. Indeed, either x or y is not almost solved or both are almost solved. In both cases **Coalesce** can be applied, either replacing x with y or y with x.

Let us consider now one irreducible conjunction of atomic formulas ϕ. By construction, a variable x which occurs in an equation $x = s$ of ϕ and which is maximal w.r.t. the occur check relation must occur only once in ϕ, otherwise one of the merge rules can be applied. Indeed, if neither the 6 first merge rules cannot be applied, and if x is maximal, x can only occur as a member of an equation and thus, the last merge rule can be applied.

This means that every maximal variable is solved. Actually, this applies to the unsolved part of ϕ as well: every maximal variable of the unsolved part is solved. This shows that the unsolved part of ϕ should be empty: every irreducible conjunction is solved. This shows that ϕ is a DAG solved form since, if some $[\![q_i]\!]$ (resp. some $[\![C_i]\!]$) is empty, then one of the rules of figure 7 can be applied. □

COROLLARY 7.4. *The satisfiability of S_0-unification problems is decidable.*

Now, we conclude this section with a result relating the requirements of (Comon, 1993a) in the order-sorted completion with the above completeness result. The following lemma has been used in (Comon, 1993a) with a forward reference. Let us recall that a set $\{t_1, \ldots, t_n\}$ of first-order terms has the ST-property if, for every context variable X and any two occurrences $X(s)$ and $X(t)$ in $s_1, \ldots, s_n, t_1, \ldots, t_n$, we have $s \equiv t$.

LEMMA 7.5. *Let $\exists\vec{x}.\exists\vec{X}.s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge M$ be a general unification problem and $\exists\vec{x'}.\exists\vec{X}'.\phi_{eq} \wedge \phi_m$ be one of its solved forms. Then $\{s_1\sigma_{\phi_{eq}}, \ldots, s_n\sigma_{\phi_{eq}}, t_1\sigma_{\phi_{eq}}, \ldots, t_n\sigma_{\phi_{eq}}\}$ has the ST property.*

PROOF. Let $S_0 = (s_1, \ldots, s_n, t_1, \ldots, t_n)$. The ST-property implies that this sequence is admissible. By lemmas 2.6, 3.2, 4.2, the solved forms are S_0-unification problems.

Let $\phi_{eq} \equiv x_1 = u_1 \wedge \ldots \wedge x_n = u_n$. We prove the lemma by induction on n.

Assume $n = 1$. Then $\sigma_{\phi_{eq}} = \{x_1 \mapsto u_1\}$. By definition of S_0-unification problems, each second-order variable can only occur once in ϕ_{eq}. If x_1 is itself a context variable, then the property is obvious. If x_1 is a first-order variable, by property 5 of S_0-unification problems, for every solution θ of $x_1 = u_1$ and every context variable X occurring both in S_0 (with argument s) and in $x_1 = u_1$ (with argument t), $s\theta \equiv t\theta$. But, now, σ doesn't constrain X (nor the variables of t), which means that $s\sigma \equiv t$.†

Now, if $n \geq 2$, consider $\psi \equiv x_2 = u_2 \wedge \ldots \wedge x_n = u_n$. By induction hypothesis, $\{s_1\sigma_\psi, \ldots, s_n\sigma_\psi, t_1\sigma_\psi, \ldots, t_n\sigma_\psi\}$ has the ST property. Moreover $\sigma_{\phi_{eq}} = \{x_1 \mapsto u_1\sigma_\psi\}\sigma_\psi$.

If x_1 is not a context variable and if $u_1 \in T(F, \mathcal{X})$, the property is straightforward. If x_1 is not a context variable and $u_1 \equiv X(v_1)$, then $v_1 \in \mathrm{Arg}_\phi(X)$ and for all solutions θ of ϕ_{eq} and every subterm $X(t)$ of an element of S_0, $t\theta \equiv v_1\theta$ by property 6 of S_0-unification problems, which means again that $t\sigma_\psi \equiv v_1\sigma_\psi$, hence $t\sigma_{\phi_{eq}} \equiv v_1\sigma_{\phi_{eq}}$. Now, assume that x_1 is a context variable: $u_1 \equiv v_1[\circ]_p$. If u_1 does not contain any context variable, then the property is straightforward (from induction hypothesis). Similarly, if there is no context variable on the path to p, the property is again straightforward by the induction hypothesis. The only problem may occur when a context variable Y occur along the path p: $u_1 \equiv v_1[Y(w_1[\circ]_q)]_r$, in which case, if $x_1(s)$ and $Y(t)$ occur in S_0, we have to prove that $t\sigma_{\phi_{eq}}$ and $w_1[s]_q\sigma_{\phi_{eq}}$ are identical. This is again ensured by property 6 and the definition of Arg: $w_1[s]_q \in \mathrm{Arg}(Y)$. □

† Strictly speaking, we use the fact that $(\forall\sigma \text{ ground } .s\sigma \equiv t\sigma) \Leftrightarrow s \equiv t$, which holds as soon as $T(F)$ contains at least two elements (for first-order terms s, t) and $C(F)$ contains at least two elements (for context terms s, t). This is always true when F contains at least one constant and one non-constant symbol, which we assumed from the beginning.

8. Extensions

Many questions arise about possible extensions. We consider here four of them:

1 Can regular languages be replaced by some larger class of languages? Actually, it is possible to extend all our results and techniques to the broader class of languages that are recognizable by an *automaton with equality tests* (Bogaert & Tison, 1991). Such languages enjoy nice closure properties and decidability results and are a strict superclass of regular sets. We only have to extend the technical constructions of lemmas 2.4 and 2.2.

2 Is it possible to lift the results to arbitrary second order variables? Of course, we have to keep our restrictions on the kind of formulas we consider since second-order unification is undecidable (Goldfarb, 1981). I think that everything can be extended to the higher order case (this is only a conjecture). However, this extension would require a dramatic modification of the logic. On the other hand, this is useless for our completion of rewrite systems.

3 Is it possible to enrich the logic with higher-order equational axioms ? If we keep the restriction on the occurrences of second-order variables, it is of course possible to allow such kind of axioms without changing anything in our completion procedure. But we also would like to handle equational axioms such as those of (Jouannaud & Okada, 1991) for example.

4 Is is possible to define finer reduction orderings on second-order terms than the straightforward extension of the recursive path ordering ? We already mentioned that solving constraints combining higher-order ordering constraints and membership constraints would lead to such refined reduction orderings.

We also want to investigate extensions of the technique to other (symbolic) constraint languages (e.g. equational formulas or symbolic ordering constraints).

Acknowledgements

I thank A. Werner and two anonymous referees for their valuable comments on earlier versions of this paper.

References

Bachmair, Leo, & Plaisted, David A. 1985. Termination orderings for Associative-Commutative rewriting systems. *Journal of Symbolic Computation*, **1**(4), 329–349.

Bogaert, B., & Tison, Sophie. 1991 (Feb.). *Automata with Equality Tests*. Tech. rept. IT 207. Laboratoire d'Informatique Fondamentale de Lille, USTL, Lille, France.

Chen, H., & Hsiang, Jieh. 1991. Logic Programming with recurrence domains. *In:* Albert, Javier Leach, Monien, Burkhard, & Artalejo, M. Rodriguez (eds), *18th International Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 510. Madrid, Spain: Springer-Verlag.

Comon, Hubert. 1990. Equational Formulas in Order-Sorted Algebras. *In: Proc. 17th Int. Coll. on Automata, Languages and Programming, Warwick, LNCS 443*. Springer-Verlag.

Comon, Hubert. 1993a (Aug.). *Completion of Rewrite systems with membership constraints. Part I: deduction rules.* This is a first part of a paper whose abstract appeared in Proc. ICALP 92, Vienna. It will appear in the Journal of Symbolic Computation.

Comon, Hubert. 1993b. Constraints in Term Algebras (Short Survey). *In:* M. Nivat, C. Rattray, T. Rus, & Scollo, G. (eds), *Proc. Conf. on Algebraic Methodology and Software Technology.* Workshop in Computing. Univ. of Twente: Springer-Verlag. Invited talk.

Comon, Hubert. 1995. On Unification of Terms with Integer Exponents. *Math. Systems Theory*, **28**, 67–88.

Comon, Hubert, & Delor, Catherine. 1994. Equational Formulae with Membership Constraints. *Information and Computation*, **112**(2), 167–216.

Dershowitz, Nachum, Jouannaud, Jean-Pierre, & Klop, J.W. 1991. Open Problems in Term Rewriting. *Pages 445–456 of:* Book, Ronald. V. (ed), *4th International Conference on Rewriting Techniques and Applications.* Lecture Notes in Computer Science, vol. 488. Como, Italy: Springer-Verlag.

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

Goldfarb, Warren D. 1981. Note on the undecidability of the second-order unification problem. *Theoretical Computer Science*, **13**, 225–230.

Huet, Gérard. 1976. *Résolution d'équations dans les langages d'ordre* $1, 2, \ldots \omega$. Thèse d'Etat, Univ. Paris 7.

Jouannaud, Jean-Pierre, & Kirchner, Claude. 1991. Solving equations in Abstract Algebras: A Rule-Based Survey of Unification. *In:* Lassez, Jean-Louis, & Plotkin, Gordon (eds), *Computational Logic: Essays in Honor of Alan Robinson.* MIT-Press.

Jouannaud, Jean-Pierre, & Okada, Mitsuhiro. 1991. Executable Higher-Order Algebraic Specification Languages. *Pages 350–361 of: Proc. 6th IEEE Symp. Logic in Computer Science, Amsterdam.*

Nivat, M., & Podelski, A. 1989. Tree monoids and recognizable sets of finite trees. *Pages 351–367 of:* Ait-Kaci, H., & Nivat, M. (eds), *Resolution of Equations in Algebraic Structures*, vol. 1. New York: AP.

R. Galbavý and M. Hermann. 1992. *Unification of infinite sets of terms schematized by primal grammars.* Research Report 92-R-220. Centre de Recherche en Informatique de Nancy, Nancy, France.

Salzer, Gernot. 1992. On Unification of Infinite Sets of Terms and its Applications. *Pages 409–421 of:* Voronkov (ed), *Proc. LPAR 92, LNCS 624.*

A. Proofs of the technical lemmas of section 2

Lemma 2.2 *For every $f \in F$, there is a function f^{-1} which associates to each $\underline{s} \in \mathcal{SE}$ a finite set of tuples of sort expressions, in such a way that*

$$f(t_1, \dots, t_n) \in [\![\underline{s}]\!]_{\mathcal{A}} \quad \Leftrightarrow \quad \exists(\underline{s}_1, \dots, \underline{s}_n) \in f^{-1}(\underline{s}), \forall i = 1, \dots, n, t_i \in [\![\underline{s}_i]\!]_{\mathcal{A}}$$

Similarly, for every $f \in F$, there is a function, again denoted f^{-1}, which associates to each $C \in \mathcal{CE}$ a finite set of tuples $(\underline{s}_1, \dots, \underline{s}_{i-1}, C_i, \underline{s}_{i+1}, \dots, \underline{s}_n)$ where each $\underline{s}_i$ is a sort expression and C_i is a context expression, in such a way that

$$\begin{array}{l} f(t_1, \dots, t_{i-1}, t_i[\cdot]_p, t_{i+1}, \dots, t_n) \in [\![C]\!]_{\mathcal{A}} \quad \Leftrightarrow \\ \qquad \exists(\underline{s}_1, \dots, \underline{s}_{i-1}, C_i, \underline{s}_{i+1}, \dots, \underline{s}_n) \in f^{-1}(C), \forall i \neq j, t_j \in [\![\underline{s}_j]\!]_{\mathcal{A}} \text{ and } t_i[\cdot]_p \in [\![C_i]\!]_{\mathcal{A}} \end{array}$$

The first part of the lemma can be found in (Comon & Delor, 1994). The second part of the lemma can be seen as a result of formal language theory: $GC(F, \mathcal{X}, \mathcal{CX})$ has a structure of free monoid over an infinite alphabet. Using remarks given in (Nivat & Podelski, 1989), we could possibly derive the second part of the lemma. However, for sake of completeness, we show here an explicit construction.

PROOF. f^{-1} will associate each context expression with a finite set of tuples $(\underline{s}_1, \dots, \underline{s}_n)[C]_i$ where $\underline{s}_1, \dots, \underline{s}_n$ are sort expressions, C is a context expression and $1 \leq i \leq n$. This notation means that the tuple should contain exactly one context expression and $n-1$ sort expressions. Then, we will write $(\underline{s}_1, \dots, \underline{s}_n)[C]_i \cdot C'$ and $(\underline{s}_1, \dots, \underline{s}_n)[C]_i \cdot \underline{s}$ in place of $(\underline{s}_1, \dots, \underline{s}_n)[C \cdot C']_i$ and $(\underline{s}_1, \dots, \underline{s}_{i-1}, C \cdot \underline{s}, \underline{s}_{i+1}, \dots, \underline{s}_n)$ respectively.

We assume that f is a non-constant function symbol. It is easy to construct $f^{-1}(\underline{s})$ and $f^{-1}(C)$ when they do not involve a negation :

(i) $f^{-1}(\bot_S) = \{(\bot_S, \dots, \bot_S)\}$

(ii) $f^{-1}(\top_S) = \{(\top_S, \dots, \top_S)\}$

(iii) If $\underline{s} \in Q$, then $f^{-1}(\underline{s}) = \{(\underline{s}_1, \dots, \underline{s}_n) \in S'^n \mid f(\underline{s}_1, \dots, \underline{s}_n) \rightarrow^+_{P'} \underline{s}\}$ ($\rightarrow^+$ 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)

(iv) $f^{-1}(g(\underline{s}_1, \dots, \underline{s}_m)) = \emptyset$ if $f \neq g$

(v) $f^{-1}(f(\underline{s}_1, \dots, \underline{s}_n)) = \{(\underline{s}_1, \dots, \underline{s}_n)\}$

(vi) $f^{-1}(\underline{s} \wedge \underline{s}') = \{(\underline{s}_1 \wedge \underline{s}'_1, \dots, \underline{s}_n \wedge \underline{s}'_n) \mid (\underline{s}_1, \dots, \underline{s}_n) \in f^{-1}(\underline{s}), (\underline{s}'_1, \dots, \underline{s}'_n) \in f^{-1}(\underline{s}')\}$

(vii) $f^{-1}(\underline{s} \vee \underline{s}') = f^{-1}(\underline{s}) \cup f^{-1}(\underline{s}')$

(viii) $f^{-1}(C \cdot \underline{s}) = f^{-1}(C) \cdot \underline{s}$ if $id_{T(F)} \notin [\![C]\!]_{\mathcal{A}}$ and $f^{-1}(C) \cdot \underline{s} \cup f^{-1}(\underline{s})$ otherwise.

(ix) $f^{-1}(\epsilon) = \emptyset = f^{-1}(\bot_C)$

(x) $f^{-1}(\top_C) = \{(\top_S, \dots, \top_S)[\top_C]_i \mid 1 \leq i \leq n\}$

(xi) $f^{-1}(C \wedge C') = \{(\underline{s}_1 \wedge \underline{s}'_1, \dots, \underline{s}_n \wedge \underline{s}'_n)[C_0 \wedge C'_0]_i \mid (\underline{s}_1, \dots, \underline{s}_n)[C_0]_i \in f^{-1}(C), (\underline{s}'_1, \dots, \underline{s}'_n)[C'_0]_i \in f^{-1}(C')\}$

(xii) $f^{-1}(C \vee C') = f^{-1}(C) \cup f^{-1}(C')$

(xiii) $f^{-1}(g(q_1, \dots, q_n)[C]_i) = \emptyset$ if $f \neq g$

(xiv) $f^{-1}(f(q_1, \dots, q_n)[C]_i) = \{(q_1, \dots, q_n)[C]_i\}$

(xv) $f^{-1}(C \cdot C') = f^{-1}(C) \cdot C'$ if $id_{T(F)} \notin [\![C]\!]_{\mathcal{A}}$ and $f^{-1}(C) \cdot C' \cup f^{-1}(C')$ otherwise.

(xvi) $f^{-1}(C^*) = f^{-1}(C) \cdot C^*$

In order to define f^{-1} on expressions involving negations, we only consider normal forms of expressions w.r.t. the following rules:

$$
\begin{array}{rcl}
\neg(q_1 \wedge q_2) & \rightarrow & \neg q_1 \vee \neg q_2 \\
\neg(q_1 \vee q_2) & \rightarrow & \neg q_1 \wedge \neg q_2 \\
\neg\neg q & \rightarrow & q \\
\neg \top_S & \rightarrow & \bot_S \\
\neg \bot_S & \rightarrow & \top_S \\
\neg \top_C & \rightarrow & \bot_C \\
\neg \bot_C & \rightarrow & \top_C \\
\neg(C \wedge C') & \rightarrow & \neg C \vee \neg C' \\
\neg(C \vee C') & \rightarrow & \neg C \wedge \neg C' \\
\neg\neg C & \rightarrow & C \\
\neg f(q_1, \ldots, q_n) & \rightarrow & \bigvee_{g \neq f} g(\top_S, \ldots, \top_S) \vee \bigvee_{i=1}^{n} f(\top_S, \ldots, \top_S)[\neg q_i]_i \\
\neg f(q_1, \ldots, q_n)[C]_i & \rightarrow & \bigvee_{g \neq f} \bigvee_{j=1}^{a(g)} g(\top_S, \ldots, \top_S)[\top_C]_j \\
 & & \vee \bigvee_{j \neq i} f(\top_S, \ldots, \top_S)[\top_C]_i[\neg q_j]_j \vee f(\top_S, \ldots, \top_S)[\neg C]_i
\end{array}
$$

Now, it remains only to define f^{-1} on the negation of basic sorts $q \in Q$ and on the expressions $\neg C \cdot q$ and $\neg C \cdot C'$. For the two last constructions, we define :

$$
\begin{array}{l}
f^{-1}(\neg(C \cdot q)) = \{(\bigwedge_{U \in f^{-1}(C)} \underline{s}_1^U, \ldots, \bigwedge_{U \in f^{-1}(C)} \underline{s}_n^U) \mid \\
\quad \forall U = (q_1, \ldots, q_n)[C_0]_i \in f^{-1}(C), \\
\quad (\underline{s}_1^U, \ldots, \underline{s}_n^U) \in \{(\top_S, \ldots, \top_S)[\neg q_j]_j \mid j \neq i\} \\
\qquad \cup \{(\top_S, \ldots, \top_S)[\neg(C_0 \cdot q)]_i\}\}
\end{array}
$$

and

$$
\begin{array}{l}
f^{-1}(\neg(C \cdot C')) = \{(\bigwedge_{U \in f^{-1}(C)} \underline{s}_1^U, \ldots, \bigwedge_{U \in f^{-1}(C)} \underline{s}_n^U) \mid \\
\quad \forall U = (q_1, \ldots, q_n)[C_0]_i \in f^{-1}(C), \\
\quad (\underline{s}_1^U, \ldots, \underline{s}_n^U)[C_1]_k \in \{(\top_S, \ldots, \top_S)[\neg q_j]_j \mid j \neq i\} \\
\qquad \cup \{(\top_S, \ldots, \top_S)[\neg(C_0 \cdot C')]_i\}\}
\end{array}
$$

if $id_{T(F)} \notin [\![C]\!]_{\mathcal{A}}$. Otherwise, the definition is a little bit more complicated:

$$
\begin{array}{l}
f^{-1}(\neg(C \cdot q)) = \{(q_1' \wedge \bigwedge_{U \in f^{-1}(C)} \underline{s}_1^U, \ldots, q_n' \wedge \bigwedge_{U \in f^{-1}(C)} \underline{s}_n^U) \mid \\
\quad \forall U = (q_1, \ldots, q_n)[C_0]_i \in f^{-1}(C), \\
\quad (\underline{s}_1^U, \ldots, \underline{s}_n^U) \in \{(\top_S, \ldots, \top_S)[\neg q_j]_j \mid j \neq i\} \\
\qquad \cup \{(\top_S, \ldots, \top_S)[\neg(C_0 \cdot q)]_i\}, (q_1', \ldots, q_n') \in f^{-1}(\neg q)\}
\end{array}
$$

and similarly for $f^{-1}(C \cdot C')$.

Now it only remains to define $f^{-1}(\neg\underline{s})$ for a basic sort $\underline{s}$. 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 $SQ \cup \{\perp_S, \top_S\}$, 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 $(\underline{s}_1, \ldots, \underline{s}_n)$ such that $f(\underline{s}_1, \ldots, \underline{s}_n) \to^+_{P_d} q \in S_d - S_{fd}$.

Now, we have defined f^{-1}: we have to prove the correctness of this construction.
We prove by induction on n that, for every sort expression $\underline{s}$ of size smaller than n and for every context expression C of size smaller than n (the size is e.g. the number of symbols needed to print the expression),

(i) for every $f \in F$, for every ground terms $t_1, \ldots, t_n$, $f(t_1, \ldots, t_n) \in [\![\underline{s}]\!]_{\mathcal{A}}$ iff there is a tuple $(\underline{s}_1, \ldots, \underline{s}_n) \in f^{-1}(\underline{s})$ such that $t_1 \in [\![\underline{s}_1]\!]_{\mathcal{A}} \wedge \ldots \wedge t_n \in [\![\underline{s}_n]\!]_{\mathcal{A}}$.

(ii) for every $f \in F$, for every ground terms $t_1, \ldots, t_n$, for every ground context C_0, for every non root position $i \cdot p$ $f(t_1, \ldots, t_n)[\cdot]_{i\cdot p} \in [\![C]\!]_{\mathcal{A}}$ iff there is a tuple $(\underline{s}_1, \ldots, \underline{s}_n)[C_0]_i \in f^{-1}(C)$ such that $t_j \in [\![\underline{s}_j]\!]_{\mathcal{A}}$ for every $j \neq i$ and $t_i[\cdot]_p \in [\![C_0]\!]_{\mathcal{A}}$.

We have to investigate a number of cases:

(i) $\underline{s} = \perp_S$. The result follows from the fact that $[\![\perp_S]\!]_{\mathcal{A}}$ is empty.

(ii) $\underline{s} = \top_S$. The result follows from the fact that $t \in [\![\top_S]\!]_{\mathcal{A}}$ is satisfied for every ground term.

(iii) $\underline{s} \in S$. $f(t_1, \ldots, t_n) \in [\![\underline{s}]\!]_{\mathcal{A}}$ iff $f(t_1, \ldots, t_n) \to^*_{P'} \underline{s}$, 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): $f(t_1, \ldots, t_n) \to^*_{P'} f(\underline{s}_1, \ldots \underline{s}_n) \to_{P'} \underline{s}' \to^*_{P'} \underline{s}$. By definition, $(\underline{s}_1, \ldots, \underline{s}_n) \in f^{-1}(\underline{s})$ and, for all i, $t_i \to^*_{P'} \underline{s}_i$.

(iv) $\underline{s} = g(\underline{s}_1, \ldots, \underline{s}_m)$ with $f \neq g$. For all $f \neq g$, $f(t_1, \ldots, t_n) \notin [\![g(\underline{s}_1, \ldots, \underline{s}_m)]\!]_{\mathcal{A}}$.

(v) $\underline{s} = f(\underline{s}_1, \ldots, \underline{s}_m)$. By definition, $f(t_1, \ldots, t_n) \in [\![f(\underline{s}_1, \ldots, \underline{s}_n)]\!]_{\mathcal{A}}$ iff for all i, $t_i \in [\![\underline{s}_i]\!]_{\mathcal{A}}$.

(vi) $\underline{s} = \underline{s}_1 \wedge \underline{s}_2$.

$$\begin{array}{rcl} f(t_1, \ldots, t_n) \in [\![\underline{s}]\!]_{\mathcal{A}} & \Leftrightarrow & f(t_1, \ldots, t_n) \in [\![\underline{s}_1]\!]_{\mathcal{A}} \wedge f(t_1, \ldots, t_n) \in [\![\underline{s}_2]\!]_{\mathcal{A}} \\ & \Leftrightarrow & \exists(\underline{s}_{1,1}, \ldots, \underline{s}_{1,n}) \in f^{-1}(\underline{s}_1),\ \forall j, t_j \in [\![\underline{s}_{1,j}]\!]_{\mathcal{A}} \\ & & \exists(\underline{s}_{2,1}, \ldots, \underline{s}_{2,n}) \in f^{-1}(\underline{s}_2), \forall j, t_j \in [\![\underline{s}_{1,j}]\!]_{\mathcal{A}} \\ & & \qquad\qquad \text{by induction hypothesis} \\ & \Leftrightarrow & \exists(\underline{s}_{1,1}, \ldots, \underline{s}_{1,n}) \in f^{-1}(\underline{s}_1), \exists(\underline{s}_{2,1}, \ldots, \underline{s}_{2,n}) \in f^{-1}(\underline{s}_2), \\ & & \forall j, t_j \in [\![\underline{s}_{1,j} \wedge \underline{s}_{2,j}]\!]_{\mathcal{A}} \end{array}$$

(vii) $\underline{s} = \underline{s}_1 \vee \underline{s}_2$. 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, $f^{-1}(S_1 \cup S_2) = f^{-1}(S_1) \cup f^{-1}(S_2)$.

(viii) $\underline{s} = \neg(\underline{s}_1 \wedge \underline{s}_2)$ or $\underline{s} = \neg(\underline{s}_1 \vee \underline{s}_2)$. We are back to the above case after distributing the negation.

(ix) $\underline{s} = \neg g(\underline{s}_1, \ldots, \underline{s}_m)$ where $g \neq f$. This is straightforward

(x) $\underline{s} = \neg f(\underline{s}_1, \ldots, \underline{s}_n)$. Then

$$\begin{array}{rcl} f(t_1, \ldots, t_n) \in [\![\underline{s}]\!]_{\mathcal{A}} & \Leftrightarrow & f(t_1, \ldots, t_n) \notin [\![f(\underline{s}_1, \ldots, \underline{s}_n)]\!]_{\mathcal{A}} \\ & \Leftrightarrow & \neg(t_1 \in [\![\underline{s}_1]\!]_{\mathcal{A}} \wedge \ldots \wedge t_n \in [\![\underline{s}_n]\!]_{\mathcal{A}}) \\ & \Leftrightarrow & t_1 \notin [\![\underline{s}_1]\!]_{\mathcal{A}} \vee \ldots \vee t_n \notin [\![\underline{s}_n]\!]_{\mathcal{A}} \end{array}$$

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

(xii) $\underline{s} = C \cdot \underline{s}'$. If $id_{T(F)} \notin [\![C]\!]_{\mathcal{A}}$, then the top position of $t \in [\![C \cdot \underline{s}']\!]_{\mathcal{A}}$ must be in $[\![C]\!]_{\mathcal{A}}$. More precisely, $f(t_1, \ldots, t_n) \in [\![C \cdot \underline{s}']\!]_{\mathcal{A}}$, iff there is an index i and a position p such that $f(t_1, \ldots, t_{i-1}, t_i[\cdot]_p, t_{i+1}, \ldots, t_n) \in [\![C]\!]_{\mathcal{A}}$ and $t_i|_p \in [\![\underline{s}']\!]_{\mathcal{A}}$. Then, by induction hypothesis, $\exists(\underline{s}_1, \ldots, \underline{s}_{i-1}, C_i, \underline{s}_{i+1}, \ldots, \underline{s}_n) \in f^{-1}(C)$ such that $\forall i \neq j, t_i \in [\![\underline{s}_i]\!]_{\mathcal{A}}$ and $t_i[\cdot]_p \in [\![C_i]\!]_{\mathcal{A}}$. Which means that $t_i \in [\![C_i \cdot q]\!]$.
The additional case when $id_{T(F)} \in [\![C]\!]_{\mathcal{A}}$ is straightforward.

(xiii) $\underline{s} = \neg(C \cdot \underline{s}')$. As above, assume that $id_{T(F)} \notin [\![C]\!]_{\mathcal{A}}$. By induction hypothesis, $f(t_1, \ldots, t_n) \notin [\![C \cdot q]\!]_{\mathcal{A}}$ iff, for every tuple $(\underline{s}_1, \ldots, \underline{s}_n) \in f^{-1}(C \cdot q)$, there is an index i such that $t_i \notin [\![\underline{s}_i]\!]_{\mathcal{A}}$. On the other hand, $f^{-1}(C \cdot q) = f^{-1}(C) \cdot q$. Thus, $f(t_1, \ldots, t_n) \notin [\![C \cdot q]\!]_{\mathcal{A}}$ iff there is an index j such that, for all $U = (q_1, \ldots, q_n)[C_0]_i \in f^{-1}(C)$, either $i \neq j$ and $t_j \notin [\![q_j]\!]_{\mathcal{A}}$ or $i = j$ and $t_i \notin [\![C_0 \cdot q]\!]_{\mathcal{A}}$. Hence, the definition of $f^{-1}(C \cdot q)$ is correct.

The correctness for context expression is proved in a similar way. We skip it. □

Lemma 2.4 *For every context expression C, there is a function C^{-1} mapping every sort expression $\underline{s}$ to a finite set of pairs $(C', \underline{s}')$ of a context expression and a sort expression, in such a way that*

$$(t[s]_p \in [\![\underline{s}]\!]_{\mathcal{A}} \wedge t[\circ]_p \in [\![C]\!]_{\mathcal{A}}) \;\Leftrightarrow\; \exists(C', \underline{s}') \in C^{-1}(\underline{s}), t[\circ]_p \in [\![C']\!]_{\mathcal{A}} \wedge s \in [\![\underline{s}']\!]_{\mathcal{A}}$$

For every context expression C, there is a finite set $\mathrm{split}(C)$ *of context expressions such that $t[u[\circ]_r]_q \in [\![C]\!]_{\mathcal{A}}$ iff*

$$\exists(C_0, C_1) \in \mathrm{split}(C) : \; t[\circ]_q \in [\![C_0]\!]_{\mathcal{A}} \wedge u[\circ]_r \in [\![C_1]\!]_{\mathcal{A}}$$

Proof. Assuming that we have defined $\top_C^{-1}$, it is easy to define C^{-1}:

$$C^{-1}(\underline{s}) = \{(C \wedge C', \underline{s}') \mid (C', \underline{s}') \in \top_C^{-1}(\underline{s})\}$$

which is obviously correct. Now, it only remains to split sort and context expressions.

We are going to define $\top_C^{-1}(\underline{s})$, i.e. to construct the automata which recognizes C' and $\underline{s}'$ (resp. C_0, C_1). Then, it is sufficient to construct (using standard methods) the regular expression corresponding to an automaton (Gécseg & Steinby, 1984). Indeed, every regular expression is a sort expression (or a context expression if there is one hole) We didn't consider this point of view before since the meaning may be lost while translating each basic sort into a regular expression. Also, we loose some freedom if we impose to normalize the sort expressions at each step of transformation. Finally, if we use regular expressions, we cannot expect a straightforward generalization to a more general class of languages than regular ones. We can neither restrict the language of context and sort expressions to some subclass which do not contain regular expressions. For example, we cannot remove the star operation which was not needed in the first part of this work.

For the purpose of this proof we are going to use the symbol Ω instead of $\cdot$ (for typographic reasons). It is possible to assume the following properties for automata recognizing context expressions:

> The set of states is split into Q^0 and Q^Ω in such a way that
>
> (i) the languages recognized in some state of Q^0 are subsets of $T(F)$.
>
> (ii) every term recognized in a state of Q^Ω contains exactly one occurrence of Ω
>
> (iii) for every production rule $f(q_1,\ldots,q_n) \to q$, if $q \in Q^\Omega$, then exactly one of the q_i's is in Q^Ω, or else $f = \Omega$.

Assume that $\mathcal{A} = (F, Q, \{\underline{s}\}, P)$ is an automaton recognizing $[\![\underline{s}]\!]$, then, for each state $q \in Q$, we construct the two automata $\mathcal{A}_q^1$ and $\mathcal{A}_q^2$ as follows:

(i) $\mathcal{A}_q^1 = (F, Q_1, \{\underline{s}^\Omega\}, P_1)$ is a context automaton where Q_1 is actually split into Q_1^0 and Q_1^Ω as above. $Q_1^0 = Q$ and Q_1^Ω is a duplicated version of Q (we denote by q^Ω the state in Q_1^Ω corresponding to $q \in Q$). $P_1 = P \cup \{\Omega \to q^\Omega\} \cup P^\Omega$ where

$$P^\Omega = \{f(q_1,\ldots,q_n)[q_i^\Omega]_i \to q^\Omega \mid f(q_1,\ldots,q_n) \to q \in P\}$$

(ii) $\mathcal{A}_q^2 = (F, Q, \{q\}, P)$

We are going to show that

$$t[s]_p \in [\![\underline{s}]\!]_\mathcal{A} \quad \Leftrightarrow \quad \exists q \in Q, t[\cdot]_p \in [\![\underline{s}^\Omega]\!]_{\mathcal{A}_q^1} \wedge s \in [\![q]\!]_{\mathcal{A}_q^2}$$

Indeed, considering a reduction $t[s]_p \to_P^* \underline{s}$, we choose q in such a way that $s \to_P^* q$, then the last part of the reduction $t[q]_p \to_P^* \underline{s}$ can be translated into a reduction $t[\Omega]_p \to_{P_1}^* \underline{s}^\Omega$. Let us prove by induction on $|p_0|$ that, if $t'[q]_{p_0} \to_P^* q_0$, then $t'[\Omega]_{p_0} \to_{P_1}^* q_0^\Omega$.

(i) If $p_0 = \Lambda$, then $q = q_0$ and $\Omega \to q_0^\Omega$ by construction

(ii) If $p_0 = i \cdot p'$ and $t'[\Omega]_{p_0} \equiv f(t_1,\ldots,t_{i-1},t_i[\Omega]_{p'},t_{i+1},\ldots,t_n)$, then, for every $j \neq i$, $t_j \to_P^* q_j$, $t_i[q]_{p'} \to_P^* q_i$ and there is a rule $f(q_1,\ldots,q_n) \to q_0$ in P. Now, for $j \neq i$, $t_j \to_{P_1}^* q_j$ as well since $q_j \in Q_1^0$. Moreover, by induction hypothesis, $t_i[\Omega]_{p'} \to_{P_1}^* q_i^\Omega$. On the other hand, $f(q_1,\ldots,q_{i-1},q_i^\Omega,q_{i+1},\ldots,q_n) \to q_0^\Omega$ is a rule of P_1 by construction. Therefore, $t'[\Omega]_{p_0} \to_{P_1}^* q_0^\Omega$

Conversely, if $t[\cdot]_p \in [\![\underline{s}^\Omega]\!]_{\mathcal{A}_q^1}$ and $s \in [\![q]\!]_{\mathcal{A}_q^2}$, then $t[s]_p \in [\![\underline{s}]\!]$.

We proceed in a similar way for split. Note that there is a main difference between $\top_C^{-1}$ and split: split just consists of finding the prefixes of a word regular expression, because the position r of a hole in a context of C_0 should be a prefix of the corresponding position

p of a context in C. This is not the case for $\top_C^{-1}$. For example, let $[\![\underline{s}]\!]_{\mathcal{A}} = f(a,\cdot)^* \cdot a$. Then $\top_C^{-1}(\underline{s})$ not only consists of $(f(a,\cdot)^*, f(a,\cdot)^* a)$ (as it would be for words); we also have a decomposition along uncomparable paths: $(f(a,\cdot)^* f(\cdot, f(a,\cdot)^* a), a)$.

Assume that $\mathcal{A} = (F, Q, C, P)$ is an automaton recognizing the context language $[\![C]\!]$. Then, for each state $q \in Q^\Omega$ we construct, as above, the two context automata:

(i) $\mathcal{A}_q^1 = (F, Q_1, C, P_1)$ with $Q_1^0 = Q^0$, $Q_1^\Omega = Q^\Omega$ and P_1 contain all rules of P which do not involve Ω and the extra rule $\Omega \to q$.

(ii) $\mathcal{A}_q^2 = (F, Q, q, P)$

With this definition, we have

$$t[\cdot]_p \in [\![C]\!] \text{ iff } \exists u, v, q, p_1, p_2, t[\cdot]_p \equiv u[v[\cdot]_{p_2}]_{p_1} \wedge u[\cdot]_{p_1} \in [\![C]\!]_{\mathcal{A}_q^1} \wedge v[\cdot]_{p_2} \in [\![q]\!]_{\mathcal{A}_q^2}$$

□