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

Completion of Rewrite Systems with Membership Constraints Part I: Deduction Rules†

Hubert Comon

CNRS and LRI,

Bat. 490, Université de Paris Sud,

91405 ORSAY cedex, France.

E-mail comon@lri.lri.fr

(Received April 92, revised June 94)

We consider a constrained equational logic where the constraints are membership conditions $t \in \underline{s}$ where $\underline{s}$ is interpreted as a regular tree language. Our logic includes a fragment of second order equational logic (without projections) where second order variables range over regular sets of contexts. The problem with constrained equational logics is the failure of the critical pair lemma. That is the reason why we propose new deduction rules for which the critical pair lemma is restored. Computing critical pairs requires, however, to solve some constraints in a second-order logic with membership constraints. In a second paper we give a terminating set of transformation rules for these formulas, which decides the existence of a solution, thus showing a new term scheme unification algorithm.

Since an order-sorted signature is nothing but a bottom-up tree automaton, order-sorted equational logic falls into the scope of our study; our results show how to perform order-sorted completion without regularity and without sort decreasingness. It also shows how to perform unification in the order-sorted case, with some higher-order variables (without any regularity assumption).

Introduction

Equational logic is ubiquitous in mathematics and the sciences. But it attracted more and more attention after D. Knuth and P. Bendix proposed their famous completion procedure (1970). Unfortunately, pure equational logic is lacking expressiveness for algebraic specification languages and therefore *sorts* and *order-sorted* signatures have been introduced by J. Goguen in OBJ (Futatsugi *et al.*, 1985). Of course, the problem of extending the Knuth-Bendix completion procedure arose. Despite many efforts, this has been (so far) achieved only under very strong restrictions, both on the signature (the so-called

† 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

regularity) and on the rules (the so-called (weak) *sort-decreasing* condition which is not stable by completion).

Viewing sorts as membership constraints, we show in this paper how to drop these restrictions. In addition, it turns out that our solution is also a partial answer to the divergence of Knuth-Bendix completion, by representing infinite sets of rules as a single constrained rule. As we show below on an example, the deduction rules need to introduce second order variables. However, the fragment of second-order (order-sorted) logic we need to consider is small enough so as to allow a terminating unification algorithm (this is shown in a second part of the work). As a consequence, we obtain an effective completion procedure in this fragment of second-order (order-sorted) equational logic, without any assumption of regularity or sort-decreasingness. Besides, our study is expected to be a first step in the design of constrained completion procedures. (Previous works (Kirchner *et al.*, 1990) assumed that every constraint can be turned into a finite set of substitutions.)

The main problem with adding constraints to equational logic is the failure of Huet's critical pair lemma. For example, consider the following rewrite system:

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(x : s_2) & \rightarrow & a \\ g(x : s_3) & \rightarrow & h(x, x) \end{array} \right.$$

defined on the (order-sorted) signature:

$$\left\{ \begin{array}{lrcl} a : & & \rightarrow & s_0 \\ f : & s_0 & \rightarrow & s_0 \\ & s_1 & \rightarrow & s_2 \\ & s_2 & \rightarrow & s_2 \\ & s_3 & \rightarrow & s_3 \\ g : & s_0 & \rightarrow & s_1 \\ & s_3 & \rightarrow & s_3 \\ h : & s_3 \times s_3 & \rightarrow & s_3 \end{array} \right. \qquad \left\{ \begin{array}{l} s_0 \leq s_3 \\ s_1 \leq s_3 \\ s_2 \leq s_3 \end{array} \right.$$

$\mathcal{R}$ is terminating and there is obviously no critical pair whereas the system is not convergent because

$$f(f(h(a,a))) \longleftarrow f(f(g(a))) \longrightarrow a$$

and $f(f(h(a,a)))$ and a are both irreducible. Considering overlaps at variable positions does not help. The reason of this phenomenon becomes apparent if we view sorts as constraints. An order-sorted signature is nothing but a bottom-up tree automaton and, if we replace each sort with the language it generates, we get the following formulation:

$$\mathcal{R} = \left\{ \begin{array}{lrcl} x \in f^+(g(f^*(a))) : & f(x) & \rightarrow & a \\ & g(x) & \rightarrow & h(x, x) \end{array} \right.$$

in which we can see an overlap between the left hand side of the second rule and the constraint of the first rule.

We give in this paper inference rules which define completion procedures in a constrained equational logic where the constraints are membership conditions. In order to handle superpositions between sort-expressions and terms we have to introduce some (very restricted) second order variables. For example, in the above rewrite system $\mathcal{R}$, the superposition of the (infinite) set of rules denoted by the first constrained rule and the left hand side of the second rule, leads to the set of equations

$$\{a = f^n(h(f^k(a), f^k(a)) \mid n \geq 2, k \geq 0\}$$

A variable $x = f^k(a)$ will allow us to express the multiple occurrences of k, yielding:

$$\{x \in f^*(a) : \; a = f^n(h(x,x)) \;\; | \; n \geq 2\}$$

On the other hand, we can express the context $f^n(\circ)$ by means of a second order variable X, resulting in a single constrained equation expressing the critical pair between the two original rules:

$$x \in f^*(a) \wedge X \in ff^+ : \;\; a = X(h(x,x))$$

In such second order terms, variables are constrained to belong to some regular languages of contexts. These contexts are similar to those in (Gramlich, 1988; Kirchner, 1989) and, actually, the constrained rules we consider may also be used for solving divergence problems in the Knuth-Bendix completion procedure. Our logic improves over above cited ones in many respects. In particular, we are able to give an effective completion procedure, which is not the case in (Gramlich, 1988; Kirchner, 1989) because unification of term schemes is undecidable. The problem is similar with Toyama's membership conditions (Toyama, 1988): because he gives no restriction on the sets occurring in the membership conditions, unification cannot be effective. Indeed, such an algorithm would have, at least, to compute the intersections and decide emptiness of sets in the class under consideration. Finally, in (Chen & Hsiang, 1991; Comon, 1995; Salzer, 1992), the authors propose to enrich the term structure by (roughly) adding integer exponents. However, they need to forbid nested integer exponents in order to keep the decidability of unification. We do not have such limitations. On the other hand we have other limitations, which shows that our results are complementary to those of (Chen & Hsiang, 1991; Comon, 1995; Salzer, 1992).

As shown above, since we do not assume any additional hypothesis on the rules, we have to add some new deduction rules. This is not the approach followed by e.g. (Kirchner *et al.*, 1988; Toyama, 1988; Smolka *et al.*, 1989; Schmidt-Schauß, 1988) where they try to keep the same deduction rules while enriching the syntax. The typical example is the "sort-decreasingness" (or "weak sort-decreasingness") condition on the rules which prevents the above example of non-commuting non-critical peaks. This property is assumed in order-sorted completion, but also a similar property is assumed in (Toyama, 1988). The work of A. Megrelis (1989) is interesting because, precisely, he avoids additional hypothesis by adding new deduction rules (that is exactly what we are going to do). Unfortunately, his logic is currently too rich for expecting a confluence criterion. This means that completion will always run forever in his framework.

Now, there are several recent works (which are going to be published in this special issue) dealing with non-sort-decreasing rewrite systems. For example, A. Werner (1993) use both extensions of the syntax and new deduction rules in order to guarantee the critical pair lemma. The new deduction rules include so-called "T-contacts" which compute some critical pairs at variable positions. It turns out that there are very simple examples of non-sort-decreasing systems where the completion does terminate using A. Werner's approach and for which our completion technique doesn't terminate. But, on the other hand, the converse should also be true: the approaches are (strictly speaking, because A. Werner's approach seems more useful for practical purposes) incomparable. However, there are other advantages of our sort-as-constraints approach. First of all, it is more expressive and we can schematize infinite sets of rules which cannot be captured otherwise. Then, despite context variables do not capture projections, it is some kind of higher-order completion and might be useful in other higher-order completion problems, as well as in other constrained completion problems.

The constraint system we consider is a fragment of a second order, order-sorted logic in which second order variables range over regular (one hole) contexts. W. Farmer (1988) has actually shown an effective algorithm for computing solved forms for monadic second order unification problems. This gives in particular a decision procedure for monadic second order order-sorted unification. W. Farmer works on terms which are built on an alphabet of (constant) function symbols whose arity is at most 1 (but the functional variables may have any arity). On the other hand, we are working on any (finite) alphabet of constant function symbols, but we have strong restrictions: for any two occurrences $t(p)$ and $u(q)$ of a second-order variable X, the corresponding subterms $t|_p$ and $u|_q$ must be identical. (This is a little bit more general than the linearity w.r.t. second order variables.) We also consider the extended framework of *order-sorted* logic. This means that our formulas also involve membership conditions (first order as well as second order conditions). Anyway, it has been shown that the general case of second-order unification is undecidable (Goldfarb, 1981) and we have therefore to assume some additional hypotheses. It turns out that our additional hypotheses are adequate with our application to order-sorted completion: they do not entail any additional restriction on the rewrite systems and they are stable by completion.

In this first part of the work, we present the logic (section 1) and we give the deduction rules (section 2) and prove a critical pair lemma. Finally, we show in section 3 that these deduction rules preserve a property of the constraint system which will justify the constraint system we consider in the second part of the work (presented in another paper).

1. The Constrained Logic

1.1. Syntax

1.1.1. Terms and Contexts

F is a fixed (finite) alphabet of *function symbols*. Each function symbol f is associated with a fixed non-negative integer: its *arity* $a(f)$. As usual such a graded alphabet defines a set of finite trees $T(F)$.

$\mathcal{X}$ is a fixed infinite alphabet of *variable symbols* (disjoint from F). $T(F, \mathcal{X})$ is the set of trees built on $F \cup \mathcal{X}$ where each symbol of $\mathcal{X}$ is considered as having arity 0. In general, variables will be denoted $x, x_1, x', y, z....$

As we explained in introduction, we also need here some further syntactic constructions involving context variables. $\mathcal{CX}$ is a fixed infinite alphabet of *context variable symbols* (supposed disjoint from $\mathcal{X} \cup F$). Variables belonging to $\mathcal{CX}$ are denoted using upper case letters: $X, X_1, X', Y, Z, ...$ $T(F, \mathcal{X}, \mathcal{CX})$ is the set of finite terms built on the alphabet $F \cup \mathcal{X} \cup \mathcal{CX}$ where each symbol of $\mathcal{X}$ is considered as having arity 0 and each symbol of $\mathcal{CX}$ is considered as having arity 1. † *Positions* are, as usual, strings of integers defining

† We do not need variables of arity larger than 1. We do not know actually whether our results carry over functional variables of arity larger or equal to 2

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 at position p.

EXAMPLE 1.1. *$f(X(x), Y(X(a))) \in T(F, \mathcal{X}, \mathcal{CX})$ if $Y, X \in \mathcal{CX}$, $x \in \mathcal{X}$ and $a, f \in F$ have respectively arity 0 and arity 2.*

If t is a term, $Var(t)$ is the set of variables (both large and small ones) occurring in t. $GC(F, \mathcal{X}, \mathcal{CX})$ is the set of all *general contexts* which are obtained from any $t \in T(F, \mathcal{X}, \mathcal{CX})$ by abstracting one of its subterms. Each element of $GC(F, \mathcal{X}, \mathcal{CX})$ can be written $t[\circ]_p$ where $t \in T(F, \mathcal{X}, \mathcal{CX})$ and $p \in Pos(t)$. $\circ$ is the "hole holder". Actually, $GC(F, \mathcal{X}, \mathcal{CX})$ is the subset of $T(F \cup \{\circ\}, \mathcal{X}, \mathcal{CX})$ consisting of all terms containing exactly one occurrence of $\circ$. A general context $t[\circ]_p$ can also be viewed as a mapping $\lambda x.t[x]_p$ from $T(F, \mathcal{X}, \mathcal{CX})$ into $T(F, \mathcal{X}, \mathcal{CX})$. $C(F)$ is the subset of $GC(F, \mathcal{X}, \mathcal{CX})$ of contexts which do not contain any element in $\mathcal{X} \cup \mathcal{CX}$. The concatenation $t[\circ]_p \cdot u$ of a general context and a term is defined as the term $t[u]_p$. Similarly, the concatenation of a general context $t[\circ]_p$ and a general context c is the general context $t[c]_p$.

$T(F, \mathcal{X}, \mathcal{CX})$ has a structure of F-algebra in a straightforward way. Moreover, it has a unique extension property: any pair of mappings σ_1 from $\mathcal{X}$ into $T(F, \mathcal{X}, \mathcal{CX})$ and σ_2 from $\mathcal{CX}$ into $GC(F, \mathcal{X}, \mathcal{CX})$ can be extended in a unique way to a pair of mappings $(\widetilde{\sigma}, \widehat{\sigma})$ from $T(F, \mathcal{X}, \mathcal{CX})$ into itself and from $GC(F, \mathcal{X}, \mathcal{CX})$ into itself respectively in such a way that

(i) $\widetilde{\sigma}|_{\mathcal{X}} = \sigma_1$ and $\widehat{\sigma}|_{\mathcal{CX}} = \sigma_2$

(ii) $f(t_1, \ldots, t_n)\widetilde{\sigma} = f(t_1\widetilde{\sigma}, \ldots, t_n\widetilde{\sigma})$ for all $t_1, \ldots, t_n \in T(F, \mathcal{X}, \mathcal{CX})$ and for all $f \in F$ of arity n

(iii) $f(t_1, \ldots, t_{i-1}, c, t_{i+1}, \ldots, t_n)\widehat{\sigma} = f(t_1\widetilde{\sigma}, \ldots, t_{i-1}\widetilde{\sigma}, c\widehat{\sigma}, t_{i+1}\widetilde{\sigma}, \ldots, t_n\widetilde{\sigma})$, for all t_1, $\ldots, t_n \in T(F, \mathcal{X}, \mathcal{CX})$, all $c \in GC(F, \mathcal{X}, \mathcal{CX})$, all $f \in F$ of arity n and all $1 \leq i \leq n$.

(iv) $c\widehat{\sigma} \cdot t\widetilde{\sigma} = (c \cdot t)\widetilde{\sigma}$ and $c_1\widehat{\sigma} \cdot c_2\widehat{\sigma} = (c_1 \cdot c_2)\widehat{\sigma}$ for all $c, c_1, c_2 \in GC(F, \mathcal{X}, \mathcal{CX})$ and $t \in T(F, \mathcal{X}, \mathcal{CX})$.

In the following, the pair (σ_1, σ_2) will be confused with $\widetilde{\sigma}$ and $\widehat{\sigma}$ and simply written σ. Such a σ will be called a *substitution* if it is the identity on $\mathcal{X} \cup \mathcal{CX}$ but on a finite set. When σ is a substitution, its *domain* $Dom(\sigma)$ is the union of $Dom_{\mathcal{X}}(\sigma) = \{x \in \mathcal{X} \mid x\sigma \neq x\}$ and $Dom_{\mathcal{CX}}(\sigma) = \{X \in \mathcal{CX} \mid X\sigma \neq X\}$. As usually, $\{x_1 \mapsto t_1; \ldots;\ x_n \mapsto t_n;\ X_1 \mapsto c_1;\ \ldots X_m \mapsto c_m\}$ where $t_1, \ldots, t_n \in T(F, \mathcal{X}, \mathcal{CX})$ and $c_1, \ldots c_m \in GC(F, \mathcal{X}, \mathcal{CX})$ denotes the substitution whose domain is $\{x_1, \ldots, x_n, X_1, \ldots, X_m\}$ and whose graph is extensively specified in the notation. Finally, the *range* of a substitution σ is the set of terms in $T(F, \mathcal{X}, \mathcal{CX}) \cup GC(F, \mathcal{X}, \mathcal{CX})$ which are images of some variable in the domain of σ.

Finally, given three terms $s, t, u \in T(F, \mathcal{X}, \mathcal{CX})$, $s\{t \mapsto u\}^*$ is the term obtained by replacing in s *all* occurrences of t with u.

1.1.2. Sort Expressions and Context Expressions

Sort expressions will involve basic sort symbols, application of a function symbol to sort expressions and Boolean operations on sort expressions. Similar constructions will be used for context expressions. In additions, we will use constructions which relate sort expressions and context expressions.

Let Q be a finite set of *sort symbols*, disjoint from $F \cup \mathcal{X} \cup \mathcal{CX}$. $\bot_S, \top_S, \epsilon, \top_C, \bot_C$ are also particular symbols, which do not belong to $Q \cup \mathcal{X} \cup \mathcal{CX} \cup F$. Then, the set $\mathcal{SE}$ of *sort expressions* and the set $\mathcal{CE}$ of *context expressions* are defined as the least sets which satisfy:

(i) $Q \cup \{\top_S, \bot_S\} \subseteq \mathcal{SE}$
(ii) $\{\epsilon, \top_C, \bot_C\} \subseteq \mathcal{CE}$.
(iii) $q \wedge q', q \vee q', \neg q \in \mathcal{SE}$ if $q, q' \in \mathcal{SE}$.
(iv) $C \wedge C'$, $C \vee C'$, $\neg C \in \mathcal{CE}$ if $C, C' \in \mathcal{CE}$
(v) $f(q_1, \ldots, q_n) \in \mathcal{SE}$ if $f \in F$ and $q_1, \ldots, q_n \in \mathcal{SE}$
(vi) $f(q_1, \ldots, q_n)[C]_i \in \mathcal{CE}$ if $1 \leq i \leq n$, $q_1, \ldots, q_n \in \mathcal{SE}$ and $C \in \mathcal{CE}$
(vii) $C \cdot q \in \mathcal{SE}$ if $C \in \mathcal{CE}$ and $q \in \mathcal{SE}$
(viii) $C \cdot C' \in \mathcal{CE}$ if $C, C' \in \mathcal{CE}$
(ix) $C^* \in \mathcal{CE}$ if $C \in \mathcal{CE}$.

EXAMPLE 1.2. *Assume that we have as basic sort symbols* $\mathsf{int}, \mathsf{nat}, \mathsf{even} \in Q$ *and function symbols* $s, +, \times, sq, 0$ *where* 0 *is a constant,* s *is unary and* $+, \times$ *are binary and used in infix notation. (The intuitive meaning of these symbols should be clear). We can construct the expression* $\mathsf{odd} \stackrel{\text{def}}{=} \mathsf{nat} \wedge \neg \mathsf{even}$. *We can also define the set of products of natural numbers containing one square:* $((\mathsf{nat} \times \mathsf{nat})[\epsilon]_2)^* \cdot (sq(\mathsf{nat}) \times ((\mathsf{nat} \times \mathsf{nat})[\epsilon]_2)^* \cdot \mathsf{nat})$, *which is depicted on figure 1.*

We can also abbreviate the former expression, replacing effectively $[\epsilon]$*:* $(\mathsf{nat} \times \epsilon)^* \cdot (sq(\mathsf{nat}) \times (\mathsf{nat} \times \epsilon)^* \cdot \mathsf{nat})$

1.1.3. Formulas

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

(i) "$t[X]_p \in C$" where $t[X]_p \in GC(F, \mathcal{X}, \mathcal{CX})$ 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) "$X = t[Y]_p$" where $X, Y \in \mathcal{CX}$ and $t[Y]_p \in GC(F, \mathcal{X}, \mathcal{CX})$.

and the logical connectives $\wedge, \vee, \exists x, \exists X$. In addition $\bot$ 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.

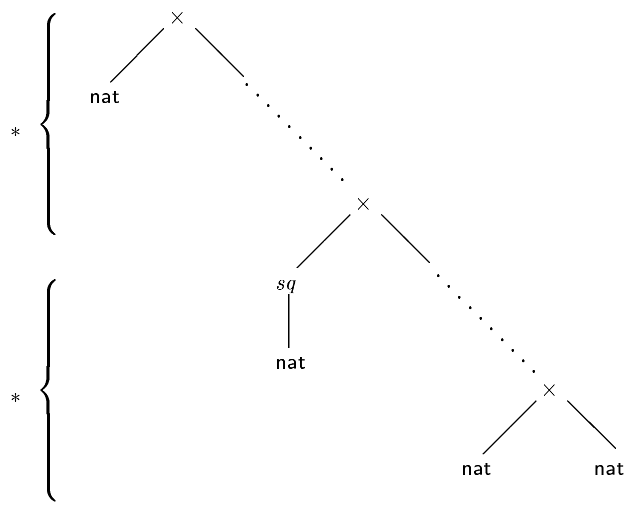

Figure 1. The sort expression of example 1.2

EXAMPLE 1.3. *Using example 1.2, we can express that x is a square: $x \in sq(\mathsf{nat})$. This expression is a membership constraint. A more general constraint can be*

$$s(0) + X \in (s(0) + \epsilon)^* \wedge X = s(Y) \wedge X(x) = s(Y(x))$$

1.1.4. Constrained Rules and Constrained Equations

An *equation* is a pair of terms $s, t \in T(F, \mathcal{X}, \mathcal{CX})$, which is denoted $s = t$. Note that s, t cannot be contexts in our logic. Note also that we make no difference between $s = t$ and $t = s$. A *constrained equation* is a pair $(\phi, s = t)$ where ϕ is a membership constraint and $s = t$ is an equation. Such a pair is written $\phi : \ s = t$. A *constrained rule* is a constrained equation in which the order on s, t is relevant. It is written $\phi : \ s \to t$. (We do not need to assume here that $Var(t) \subseteq Var(s)$.) As a consequence of the constraint solving results of the second part of this work, we may always assume that the constraints are membership solved forms.

EXAMPLE 1.4. *Using the example given in introduction,*

$$x \in f((f(\epsilon))^* \cdot g((f(\epsilon))^* \cdot a)) : \ \ f(x) \to a$$

is a constrained rule.

1.2. Semantics

As explained in the introduction, we are going to interpret sort expressions as regular sets of trees and context expressions as regular set of trees "with one hole". Since regular languages have all desired closure properties, it is actually sufficient to assume such an interpretation for the sort symbols.

1.2.1. Sort Expressions and Context Expressions

Definition 1.5. *A (bottom-up)* tree automaton *is a tuple* (F, Q, Q_f, P) *where* F *is a set of function symbols (together with their arity),* Q *is a finite set of* states, Q_f *is a subset of* Q*, called* final states *and* P *is a set of* rules *of the form*

$$f(q_1, \ldots, q_n) \to q$$

or

$$q \to q'$$

where $q_1, \ldots, q_n, q, q' \in Q$ *and* f *has arity* n.

A term t *is* accepted *(or* recognized*) by the automaton, if it can be rewritten, using the set of rewrite rules* P *to a final state.*

The language *accepted (or recognized) by an automaton* $\mathcal{A}$ *is the set of terms that are accepted by* $\mathcal{A}$.

There is correspondence between what is called an *order-sorted signature* and tree automata. A rule $q \to q'$ indeed corresponds to the sort inclusion $q \leq q'$ and a rule $f(q_1, \ldots, q_n) \to q$ corresponds to a declaration $f : q_1 \times \ldots \times q_n \to q$. Then what is called "a term of sort q" is a term accepted by the automaton in state q.

Tree automata only accept sets of ground terms. But this is not a restriction as it is always possible to add finitely many rules and function symbols in such a way that infinitely many expressions (called variables) are accepted in each state. Or, in order to stay within usual notations, it is always possible to assume rules of the form $x \to q$ stating that the variable x is of sort q. See section 1.2.4 for more discussion about this point.

We assume given a tree automaton $\mathcal{A}$ whose set of states contains the set of sort symbols Q. Then, we define the semantics mappings $[\![\_]\!]_{\mathcal{A}}$ from $\mathcal{SE}$ into $2^{T(F)}$ (subsets of $T(F)$) and from $\mathcal{CE}$ into $2^{C(F)} \subseteq 2^{T(F)^{T(F)}}$ (subsets of mappings from $T(F)$ to $T(F)$) as follows:

(i) $[\![q]\!]_{\mathcal{A}}$ is the language recognized by $\mathcal{A}$ in the state q, if $q \in Q$. †

(ii) $[\![\bot_S]\!]_{\mathcal{A}} = \emptyset$, $[\![\top_S]\!]_{\mathcal{A}} = T(F)$, $[\![\top_C]\!]_{\mathcal{A}} = C(F)$, $[\![\bot_C]\!]_{\mathcal{A}} = \emptyset$, $[\![\epsilon]\!]_{\mathcal{A}} = \{id_{T(F)}\}$ where $id_{T(F)}$ is the identity function on $T(F)$.

(iii) $[\![\_]\!]_{\mathcal{A}}$ is a morphism from $(\mathcal{SE}, \wedge, \vee, \neg)$ into $(2^{T(F)}, \cap, \cup, \overline{})$ where $\overline{A}$ is the complement of A in $T(F)$. This means, for example, that $[\![q \wedge q']\!]_{\mathcal{A}} = [\![q]\!]_{\mathcal{A}} \cap [\![q']\!]_{\mathcal{A}}$.

(iv) $[\![\_]\!]_{\mathcal{A}}$ is a morphism from $(\mathcal{CE}, \wedge, \vee, \neg)$ into $(2^{C(F)}, \cap, \cup, \overline{})$

(v) $[\![f(q_1, \ldots, q_n)]\!]_{\mathcal{A}} = \{f(t_1, \ldots, t_n) \in T(F) \mid t_1 \in [\![q_1]\!]_{\mathcal{A}}, \ldots, t_n \in [\![q_n]\!]_{\mathcal{A}}\}$

(vi) $[\![f(q_1, \ldots, q_n)[C]_i]\!]_{\mathcal{A}} = \{\lambda x.f(t_1, \ldots, t_n)[u[x]_p]_i \in C(F) \mid \lambda x.u[x]_p \in [\![C]\!]_{\mathcal{A}}, \forall i, t_i \in [\![q_i]\!]_{\mathcal{A}}\}$

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

(viii) $[\![C \cdot C']\!]_{\mathcal{A}} = \{\lambda x.t[u[x]_p]_q \in C(F) \mid t[\circ]_q \in [\![C]\!]_{\mathcal{A}}, u[\circ]_p \in [\![C']\!]_{\mathcal{A}}\}$

† Then $[\![q]\!]_{\mathcal{A}}$ is a regular tree language: that is why we use sometimes a regular expression instead of a sort symbol in the examples.

(ix) $[\![C^*]\!]_{\mathcal{A}} = \bigcup_{n\geq 0} [\![C^n]\!]_{\mathcal{A}}$ where $[\![C^0]\!]_{\mathcal{A}} = \{id_{T(F)}\}$ and $C^n = C \cdot C^{n-1}$ for $n \geq 1$.

EXAMPLE 1.6. *Let $F = \{s, p, 0, +\}$ where s, p are unary, 0 is a constant and + is binary. Let $Q = \{\mathsf{nat}, \mathsf{int}, \mathsf{even}, \mathsf{odd}\}$. The rules of the automaton $\mathcal{A}$ are:*

$$\begin{array}{rclcrclcrcl}
0 & \rightarrow & \mathsf{even} & & s(\mathsf{even}) & \rightarrow & \mathsf{odd} & & s(\mathsf{odd}) & \rightarrow & \mathsf{even} \\
p(\mathsf{nat}) & \rightarrow & \mathsf{int} & & s(\mathsf{nat}) & \rightarrow & \mathsf{nat} & & p(\mathsf{int}) & \rightarrow & \mathsf{int} \\
s(\mathsf{int}) & \rightarrow & \mathsf{int} & & \mathsf{nat}+\mathsf{nat} & \rightarrow & \mathsf{nat} & & \mathsf{int}+\mathsf{int} & \rightarrow & \mathsf{int} \\
\mathsf{nat} & \rightarrow & \mathsf{int} & & & & & & & &
\end{array}$$

In the interpretation defined by $\mathcal{A}$, we have for example $[\![\mathsf{odd}]\!]_{\mathcal{A}} \cup [\![\mathsf{even}]\!]_{\mathcal{A}} = [\![(s(\epsilon))^* \cdot 0]\!]_{\mathcal{A}}$, the set of all terms built using s and 0. We also have

$$[\![\mathsf{nat}]\!]_{\mathcal{A}} \cap [\![\neg\mathsf{odd}]\!]_{\mathcal{A}} = [\![\mathsf{even} \vee \Big((s(\epsilon) \vee (\mathsf{nat}+\epsilon) \vee (\epsilon+\mathsf{nat}))^* \cdot ((\mathsf{nat}+\epsilon) \vee (\epsilon+\mathsf{nat})) \cdot \mathsf{nat}\Big)]\!]_{\mathcal{A}}$$

i.e. the nat *terms which are not odd are either even or contain a + symbol.*

1.2.2. Solutions of Formulas

A *ground assignment* is a mapping σ from $\mathcal{X} \cup \mathcal{CX}$ to $T(F) \cup C(F)$ such that $\sigma|_{\mathcal{X}}$ is a mapping from $\mathcal{X}$ to $T(F)$ and $\sigma|_{\mathcal{CX}}$ is a mapping from $\mathcal{CX}$ to $C(F)$. Such an assignment is extended into an F-algebra homomorphism from $T(F, \mathcal{X}, \mathcal{CX})$ to $T(F)$. (This can be done in only one way as already mentioned.)

A *solution* of an equation $s = t$ is a ground assignment σ such that $s\sigma \equiv t\sigma$ ($\equiv$ is the identity on $T(F)$). A *solution* of a membership constraint $t \in q$ (resp. $X \in C$) w.r.t. the automaton $\mathcal{A}$ is a ground assignment σ such that $t\sigma \in [\![q]\!]_{\mathcal{A}}$ (resp. $X\sigma \in [\![C]\!]_{\mathcal{A}}$). These definitions of solutions are lifted to formulas in $\mathcal{F}$ in a straightforward way. $[\![\phi]\!]_{\mathcal{A}}$ denotes the set of all solutions of $\phi \in \mathcal{F}$ with respect to the automaton $\mathcal{A}$. Two formulas ϕ and ψ are said to be *equivalent* (w.r.t. $\mathcal{A}$) if $[\![\phi]\!]_{\mathcal{A}} = [\![\psi]\!]_{\mathcal{A}}$.

EXAMPLE 1.7. *Consider example 1.3 and the interpretation given by the automaton of example 1.6. The solutions of $s(0) + X \in (s(0) + \epsilon)^*$ are the assignments $X \mapsto c$ where $c \in [\![(s(0) + \epsilon)^*]\!]_{\mathcal{A}}$. The solution of $X(x) = s(Y(x))$ are the same as the solutions of $X = s(Y)$ (in both cases x is assigned to any term) and there is no solution to the conjunction of these atomic constraints since there is no context $c \in [\![(s(0)+\epsilon)^*]\!]_{\mathcal{A}}$ whose root is an s symbol.*

1.2.3. Constrained Equations

We consider here a "term generated semantics" of our constrained logic (as in (Kirchner *et al.*, 1990)): a constrained equation (resp. a constrained rule) is the representation of a possibly infinite set of equations (resp. a possibly infinite set of rules).

Let $\mathcal{A}$ be an automaton with states Q, then

$$[\![\phi :\ s = t]\!]_{\mathcal{A}} = \{s\sigma = t\sigma \mid \sigma \in [\![\phi]\!]_{\mathcal{A}}\}.$$

A similar definition holds for rewrite rules.

All definitions of reduction relation, equality step, ... for rewrite systems (see (Dershowitz & Jouannaud, 1990)) carry over the constrained case, replacing constrained equations (or rules) by the corresponding (infinite) sets of equations (or rules) they denote. Thus, we will use the notations of (Dershowitz & Jouannaud, 1990). For example, $\underset{E,\mathcal{A}}{\longleftrightarrow}$ (in general, we will omit these subscripts) is the relation $\underset{[\![E]\!]_{\mathcal{A}}}{\longleftrightarrow}$.

EXAMPLE 1.8. *Consider the example 1.4. The constrained rule actually denotes the set of rules*

$$\{f(f^{n+1}(g(f^m(a)))) \to a \mid m, n \geq 0\}$$

This does not depend on the automaton in this example since there is no sort symbol in the constraint.

1.2.4. Discussion

Our sets of constrained equations and constrained rules only denote infinite *ground* systems. Nevertheless, this is by no means a restriction:

(i) the aim of our proof system is to prove some properties of a constrained equational specification language. The declarative semantics of such programs use the set of ground instances of the axioms: we do not need any extra construction.

(ii) to prove refutationally that $E \models \forall \vec{x} : \ s = t$, we only need to consider all instances of E belonging to $T(F, \vec{x})$: it only needs ground instances of the axioms, over a (finite) extended signature.

(iii) we can include finitely many *sort declarations* in the definition of the automaton. In other words, declarations such as "x has type s" can be seen as additional rules $x \to s$ in the automaton $\mathcal{A}$. Then, the interpretation is modified accordingly: assignments to terms containing x become valid.

(iv) It is even possible to have infinite sets of variables as usual (although it is useless in our opinion). Indeed, it is sufficient to add a new (unary) constructor $x_{\underline{s}}$ for each sort $\underline{s}$ with $0 :\to \mathsf{nat}$, $s : \mathsf{nat} \to \mathsf{nat}$, $x_{\underline{s}} : \mathsf{nat} \to \underline{s}$. Actually, the variables which occur in rewriting terms are considered as (new) constants, that is why only ground terms are needed. They have to be distinguished with *logical variables* which are used in the rules and for which *no declaration is required* since the membership to some sort should be expressed in the constraint of the rule.

2. Deduction on constrained Equations

This section is devoted to the (Knuth-Bendix) completion of infinite sets of (ground) equations, which are represented by constrained equations. As shown in the introduction, the classical rules (Dershowitz & Jouannaud, 1990) are not complete, i.e. it may be the case that none of the classical rules can be applied and there is some valid (ground) equation which does not have any rewrite proof. Therefore, we have to design new deduction rules corresponding to new critical pairs lemmas.

2.1. Critical pair lemmas and Deduction Rules

The deduction rules are summarized in figure 2.

Deduce 1

$$\frac{\phi : l \to r \qquad \psi : g \to d}{\phi' : l[d]_p\sigma = r\sigma}$$

If

$$\begin{cases} 1a.\ l|_p \text{ is not a variable} \\ 1b.\ \phi \wedge \psi \wedge l|_p = g \mapsto^* \phi' \wedge \sigma \end{cases}$$

Deduce 2

$$\frac{x \in q \wedge \phi : l \to r \qquad \psi : g \to d}{\phi' : r\{x \mapsto X_1(d)\} = l\{x \mapsto X_1(d)\}}$$

If

$$\begin{cases} 2a.\ X_1(g) \in q \wedge \psi \wedge \phi \wedge X_1(d) \notin q \mapsto^* \phi' \\ 2b.\ \phi' \text{ is satisfiable} \end{cases}$$

Deduce 3

$$\frac{X_1 \in C \wedge \phi : l \to r \qquad \psi : g \to d}{\phi' : l\{X_1(t) \mapsto X_3(d)\}^*\sigma = (r\{X_1(t) \mapsto X_3(d)\}^*\sigma}$$

If

$$\begin{cases} 3a.\ \exists X_2\ X_3 \cdot X_2 \in C \wedge X_2(t) = g \wedge \phi \wedge \psi \mapsto^* \phi' \wedge \sigma \\ 3b.\ l|_p \equiv X_1(t) \end{cases}$$

Deduce 4

$$\frac{X_1 \in C \wedge \phi : l \to r \qquad \psi : g \to d}{\phi' : l\theta = r\theta}$$

If, for some $f \in F$ (with $a(f) = n$), for some $i, j \in \{1, \dots n\}$, $j \neq i$

$$\begin{cases} 4a.\ \phi \wedge \psi \wedge X_2(f(x_1, \dots, x_n))[X_4(g)]_{1\cdot j}[X_3]_{1\cdot i} \in C \\ \qquad\qquad \wedge X_2(f(x_1, \dots, x_n))[X_3]_{1\cdot i}[X_4(d)]_{1\cdot j} \notin C \ \mapsto^* \ \phi' \\ 4b.\ \phi' \text{ is satisfiable} \\ 4c.\ \theta = \{X_1 \mapsto X_2(f(x_1, \dots, x_n))[X_4(d)]_{1\cdot j}[X_3]_{1\cdot i}\} \end{cases}$$

Figure 2. Deduction Rules

Deduce 1

$$\frac{\phi : l \to r \qquad \psi : g \to d}{\phi' : l[d]_p\sigma = r\sigma}$$

If

$$\begin{cases} 1a.\ l|_p \text{ is not a variable (neither first-order nor second-order)} \\ 1b.\ \phi \wedge \psi \wedge l|_p = g \mapsto^* \phi' \wedge \sigma \end{cases}$$

It is the classical deduction rule. We adopt the convention that a substitution may also be viewed as a solved set of equations. Therefore, a substitution may appear as part of a constraint. The constraint $\phi' \wedge \sigma$ is moreover assumed to be in *solved form*. This means that ϕ' is a membership solved form. Finally, $\mapsto^*$ is the reduction of a constraint to one of its solved forms. (See Part II for more information about constraint solving)

Deduce 2

$$\frac{x \in q \wedge \phi : \; l \to r \qquad \psi : \; g \to d}{\phi' : \; r\{x \mapsto X_1(d)\} = l\{x \mapsto X_1(d)\}}$$

If

$$\begin{cases} 2a.\; X_1(g) \in q \wedge \psi \wedge \phi \wedge X_1(d) \notin q \mapsto^* \phi' \\ 2b.\; \phi' \text{ is satisfiable} \end{cases}$$

This corresponds intuitively to the superposition between a left hand side and a sort expression: x is assigned to $X_1(g)$, which has to belong to q. The condition $X_1(d) \notin q$ states that there is not already a trivial rewrite proof of the conclusion; we require that $x \in q \wedge \phi : l \to r$ cannot be applied on l if we replace x with $X_1(d)$, thus commuting the rewrite steps. This condition is not necessary, but it is very useful in pruning unnecessary deductions.

Deduce 3

$$\frac{X_1 \in C \wedge \phi : \; l \to r \qquad \psi : \; g \to d}{\phi' : \; l\{X_1(t) \mapsto X_3(d)\}^* \sigma = (r\{X_1(t) \mapsto X_3(d)\}^* \sigma}$$

If

$$\begin{cases} 3a.\; \exists X_2\, X_3 \cdot X_2 \in C \wedge X_2(t) = g \wedge \phi \wedge \psi \mapsto^* \phi' \wedge \sigma \\ 3b.\; l|_p \equiv X_1(t) \end{cases}$$

This corresponds intuitively to superpositions "inside" a second-order variable X_1: X_1 is assumed to be split into the two second-order variables X_3 and X_2 ($X_1 = X_3 \cdot X_2$). Then, the superposition takes place at X_2. This is the meaning of condition $3a$ where we have to solve $g = X_2(w)$. Then we get the two members of the new equation by simplification of the extrema of the critical peak

$$r\sigma \leftarrow l[X_3(X_2(w))]_p \sigma \to l[X_3(d)]_p \sigma$$

We indeed know that $X_2(w)\sigma$ can be reduced by $\psi : \; g \to d$: we apply this rule as long as possible on the extrema of the peak. We get the members of the equation.

Deduce 4

$$\frac{X_1 \in C \wedge \phi : \; l \to r \qquad \psi : \; g \to d}{\phi' : \; l\theta = r\theta}$$

If, for some $f \in F$ (with $a(f) = n$), for some $i, j \in \{1, \dots n\}$, $j \neq i$

$$\begin{cases} 4a.\; \phi \wedge \psi \wedge X_2(f(x_1, \dots, x_n))[X_4(g)]_{1 \cdot j}[X_3]_{1 \cdot i} \in C \\ \qquad\qquad \wedge X_2(f(x_1, \dots, x_n))[X_3]_{1 \cdot i}[X_4(d)]_{1 \cdot j} \notin C \;\; \mapsto^* \;\; \phi' \\ 4b.\; \phi' \text{ is satisfiable} \\ 4c.\; \theta = \{X_1 \mapsto X_2(f(x_1, \dots, x_n))[X_4(d)]_{1 \cdot j}[X_3]_{1 \cdot i}\} \end{cases}$$

This is similar to **Deduce 2**; it corresponds to an overlap in a second-order variable, at a position disjoint from the position of the hole. Let us give a typical example. Let R be the rewrite system consisting of the two rules:

$$\begin{cases} X \in f(\circ, b) : & g(X(a)) & \to & a \\ & b & \to & a \end{cases}$$

Consider the proof $a \leftarrow g(f(a,b)) \to g(f(a,a))$. $g(f(a,a))$ is irreducible because the constraint of the first rule is no longer satisfied. Therefore, as in the case of

Deduce 2, we have to add the hidden critical pairs which correspond to rewrite steps which prevent the satisfaction of the constraint. The condition looks complicated because we have to guess what is the shared path between the position of the hole and the position at which the second rule is applied. This shared path corresponds to the context X_2. Then we guess the function symbol below X_2 (this is f) and, assuming a maximal sharing, the position of the hole and the redex start below f with distinct sons i and j. Finally X_3 is the remaining part of X_1 towards the hole and X_4 is the remaining part of X_2 towards the redex. The condition $X_2(f(x_1, \dots, x_n))[X_3]_{1 \cdot i}[X_4(d)]_{1 \cdot j} \notin C$ ensures that the "peak" is not "commuting". In other words, it ensure that we are not adding too many consequences. See also figure 6.

The critical pair lemma states that, for "non-commuting peaks" there is a deduction rule leading to a simpler proof:

LEMMA 2.1. *Let $s, t, u \in T(F)$ and R be a set of constrained rules where the constraints are membership solved forms. If $s \xleftarrow[\mathcal{R}]{} t \xrightarrow[\mathcal{R}]{} u$, then either there is a rewrite proof (in $\mathcal{R}$) of $s \xleftrightarrow[\mathcal{R}]{*} u$ (i.e. a proof $s \xrightarrow[\mathcal{R}]{*} \xleftarrow[\mathcal{R}]{*} u$) or there is a deduction rule D such that $\mathcal{R} \vdash_D \phi :\ s' = t'$ and a proof $s \xrightarrow[\mathcal{R}]{*} \xleftrightarrow[\phi:\ s'=t']{} \xleftarrow[\mathcal{R}]{*} u$.*

PROOF. The proof is similar to the proof of Huet's critical pair lemma (Huet, 1980): we have to investigate the possible relative positions of the rewrite steps.

Assume that $t \xrightarrow[\phi: l \to r, p]{\sigma} s$ and $t \xrightarrow[\psi: g \to d, q]{\theta} u$. If p and q are not comparable (w.r.t. the prefix ordering) then the two rewrite steps commute (as in (Huet, 1980)).

Assume now that (e.g.) p is a prefix of q: $q = p \cdot \rho$. Then there are three cases† which correspond respectively to:

(i) the reduction by $g \to d$ takes place below a first-order variable position of l

(ii) the reduction by $g \to d$ takes place at a non variable position of l

(iii) the reduction by $g \to d$ takes place at a position introduced an instance of a second order variable

In the first and second cases, ρ may be written $\rho = \rho_1 \cdot q_1 \cdot \dots \cdot \rho_k$ where $\rho_1, \rho_1 \cdot 1 \cdot \rho_2, \dots, \rho_1 \cdot \dots \cdot 1 \cdot \rho_{k-1}$ are respectively positions of second-order variables $Y_1, \dots Y_{k-1}$ in l and, for each Y_i, $Y_i\sigma = u_i[\circ]_{q_i}$ for some first order term u_i. We may moreover assume that k is maximal: let $l' = l|_{\rho_1 \cdot \dots \cdot \rho_{k-1} \cdot 1}$. We assume that, for any prefix α of ρ_k which is a position of l', $l'(\alpha) \notin \mathcal{CX}$.

Similarly, in the third case, ρ may be written $\rho = \rho_1 \cdot q_1 \cdot \dots \cdot \rho_k \cdot q_k$ where $l(\rho_1 \cdot \dots \cdot \rho_{k-1} \cdot 1 \cdot \rho_k)$ is a second-order variable Y_k and $Y_k\sigma = u_k[\circ]_{q'_k}$ where q'_k is not a prefix of q_k (either q_k is a strict prefix of q'_k or q_k and q'_k are incomparable).

Now, we investigate the three cases:

† Note that we must be careful: we may have $f = t(p)$ and $f \neq t\sigma(p)$. Indeed, there may be some second order variable at a position q which is a prefix of p. Thus, the positions of constant functions symbols may change when applying a substitution, which is a main difference with the first order case.

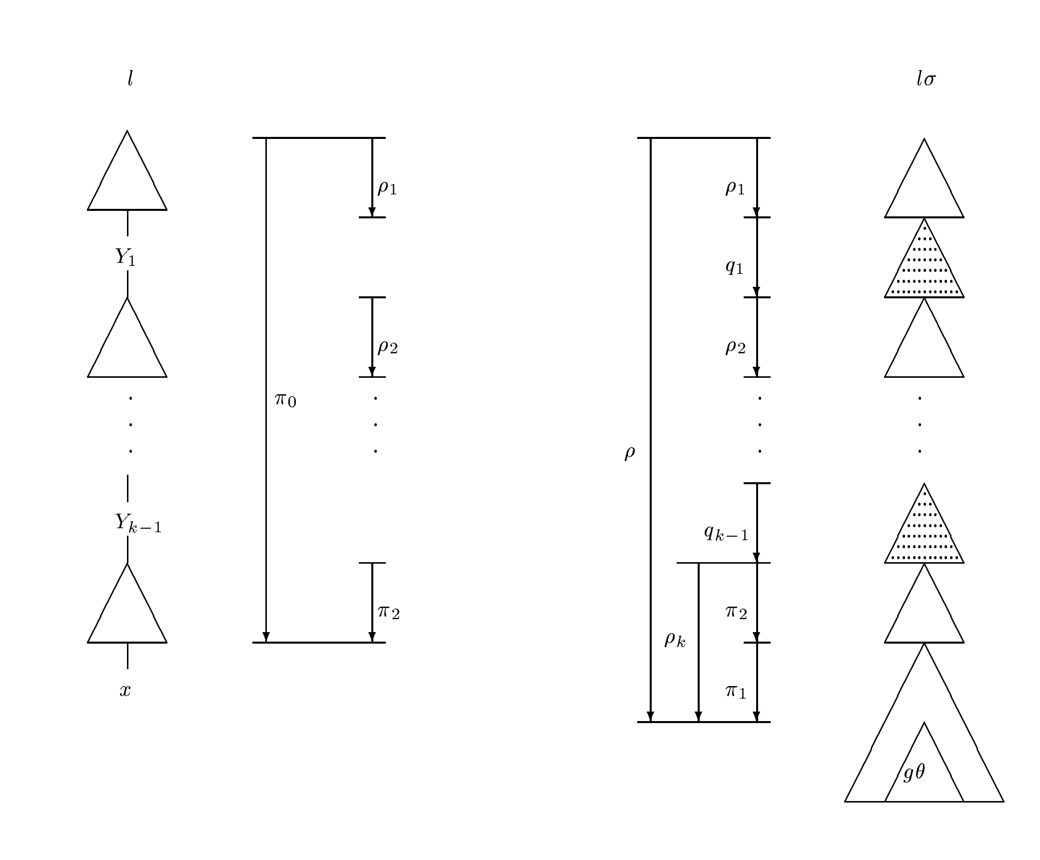

Figure 3. $l\sigma$ and $g\theta$ overlap below a first order variable

First case: there is some first order variable $x = l|_{\pi_0}$ such that π_0 is a prefix of $\rho_1 \cdot 1 \cdot \ldots \cdot 1 \cdot \rho_k$.
This case corresponds to the non-critical overlap in the standard completion. There are two subcases: either we are back to the classical critical pair lemma if the constraints do not prevent the application of the rules, or else we use **Deduce 2**. Let $\pi_0 \cdot \pi_1 = \rho_1 \cdot \ldots \cdot \rho_k$. Then π_0 can be split: $\pi_0 = \rho_1 \cdot 1 \cdot \ldots \cdot 1 \cdot \rho_{k-1} \cdot 1 \cdot \pi_2$. and $\rho_k = \pi_2 \cdot \pi_1$. (see figure 3). Now, let σ' be the substitution which is identical to σ, except on x where $x\sigma' = x\sigma[d\theta]_{\pi_1}$.

Case 1.1: $\sigma' \models \phi$. In such a case, we are back to the classical lemma: it is sufficient to apply the rule $g \to d$ at position π_1 in $x\sigma$, for each occurrence of x in l on one hand, and for each occurrence of x in r on the other hand. Then, apply $\phi : l \to r$ on the former term. We get a rewrite proof of $s = u$

Case 1.2: $\sigma' \not\models \phi$. Since σ and σ' only differ on x, there should be a membership constraint $x \in q$ in ϕ which is satisfied by σ and which is not satisfied by σ'. Let now α be the substitution defined on l by $l\sigma = l\alpha$, on g by $g\theta = g\alpha$ and on an extra second-order variable Y by $Y\alpha = x\sigma[\circ]_{\pi_1}$. Then, by hypotheses, α

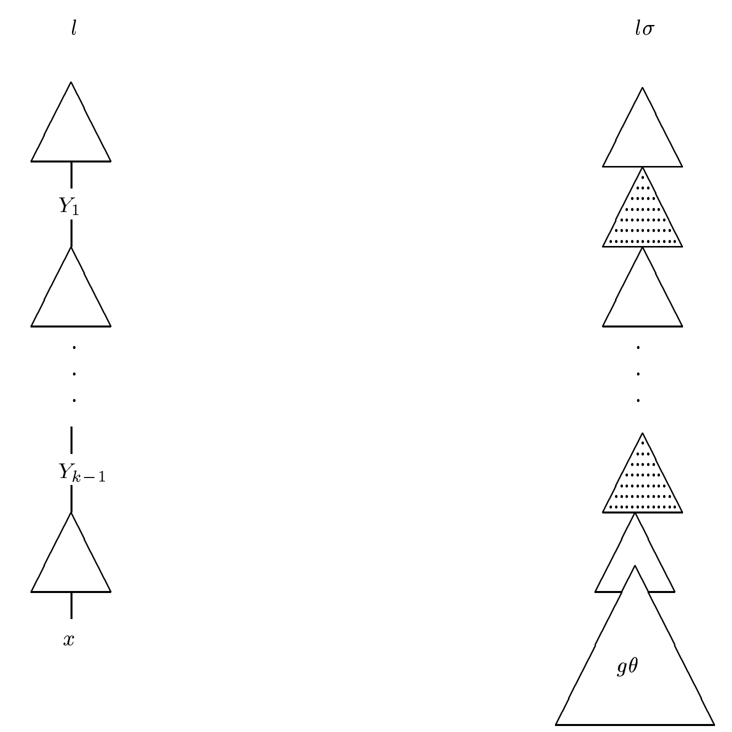

Figure 4. Overlap at a functional Position of l

is a solution of $Y(g) \in q \wedge \phi \wedge \psi$. On the other hand, α is also a solution of $Y(d) \notin q$ since $\sigma' \not\models \phi$. Therefore, it is possible to apply **Deduce 2**. α is now a solution of ϕ'. Thus, $r\sigma' = l\sigma'$ is an instance of the new equation, which shows the desired property.

Second case: $\rho_1 \ldots \rho_k$ is a non-variable position of l

This situation (depicted on figure 4) corresponds to the usual superposition case. It is solved with **Deduce 1**: by definition of $\xleftrightarrow[\phi' : v = w, p]{}$, we have $s \xleftrightarrow[\phi' : l[d]_{\rho_1 \cdot \ldots \cdot \rho_k} \lambda = r\lambda]{} u$ where $\lambda \wedge \phi$ is some (appropriately chosen) solved form of $\phi \wedge \psi \wedge l|_p = g$.

Third case: $l'(\rho_k)$ is a second-order variable Y_k

There are still two subcases depending on whether q_k is a strict prefix of q'_k or else q_k and q'_k are uncomparable.

Case 3.1: q_k is a strict prefix of q'_k . Let p_1 be the position $\rho_1 \cdot 1 \cdot \ldots \cdot 1 \cdot \rho_k$; $l|_{p_1}$ is a term $Y_k(w)$ where Y_k is a second order variable. Let p_2 be the position $\rho_1 \cdot q_1 \ldots \cdot \rho_k$. This situation is depicted on figure 5.

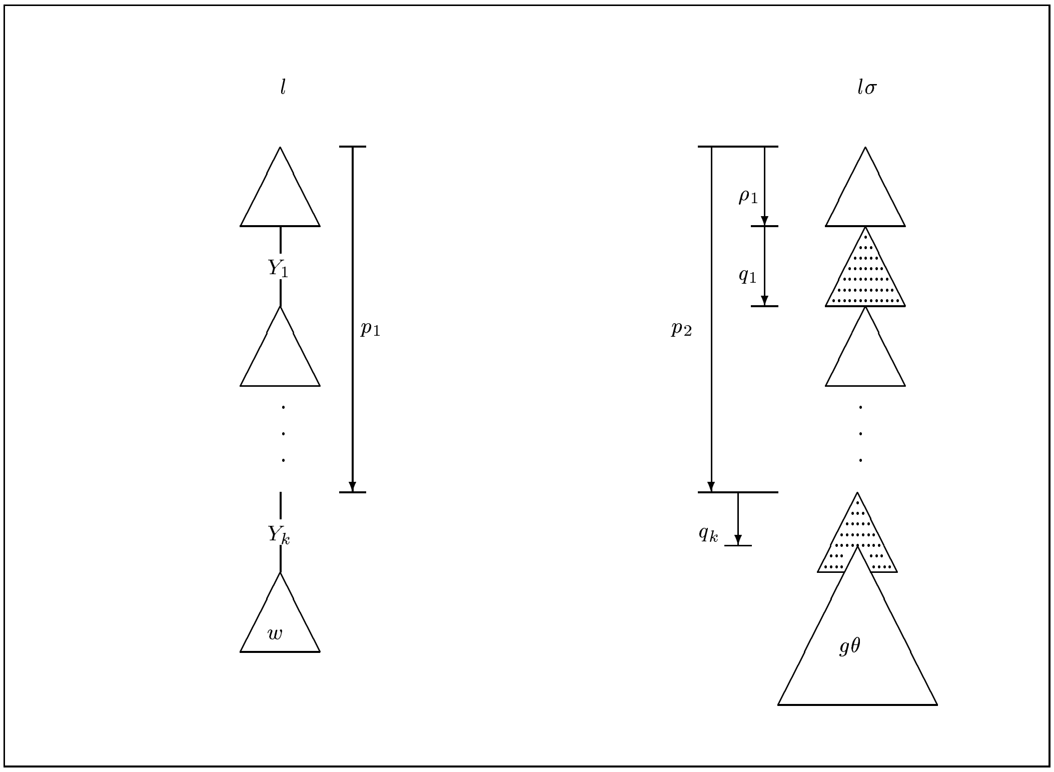

Figure 5. Overlap inside a second-order variable. The ancestor case.

Now, let τ be the substitution $\sigma\theta\{Y \mapsto l\sigma|_{p_2}[\circ]_{q_k}\}$. $Y_k(w)\sigma = Y\tau(g\theta)$ and thus $l[Y(g)]_{p_1}\tau = l\sigma$, $s = t[r\sigma]_p$, $u = t[d\theta]_q = t[Y(d)\tau]_{p\cdot p_2}$. If l' is the term obtained by replacing *all* occurrences of $Y_k(w)$ with $Y(d)$ in l, then

$$t \xrightarrow[\psi:g\to d]{p\cdot\rho} t[Y(d)\tau]_{p\cdot p_2} \xrightarrow[\psi:g\to d]{*} t[l'\tau]_p$$

Moreover, if r' is obtained by replacing all occurrences of $Y_k(w)$ with $Y(d)$ in r,

$$t[l'\tau]_p \xleftrightarrow[\phi':l'\lambda=r'\lambda]{} t[r'\tau]_p \xleftarrow[\psi:g\to d]{*} s$$

where $Y\cdot Z \in C \wedge Z(w) = g \wedge \phi \wedge \psi \mapsto^* \lambda \wedge \phi'$ is an (appropriately chosen) solved form. Indeed, there is a substitution α on Z such that $(Y\cdot Z)\tau\alpha \in [\![C]\!]_{\mathcal{A}}$ and $Z(w)\tau\alpha = g\tau$ and $\tau \models \phi \wedge \psi$. Finally, note that

$$(Y_k \in C \wedge \phi : l \to r),\ (\psi : g \to d)\ \vdash_{\textbf{Deduce 3}}\ \phi' : l'\lambda = r'\lambda.$$

Case 3.2: q_k and q'_k are uncomparable

This situation is depicted on figure 6.

We use in part the notations of case 3.1. But there are some additional ones. Let π_0 be the maximal common prefix of q_k and q'_k: $q'_k = \pi_0\cdot i\cdot\pi_1$ and $q_k = \pi_0\cdot j\cdot\pi_2$ with $i \neq j$ distinct integers. Let again $f = l\sigma(p_2\cdot\pi_0)$. Let σ' be the substitution which is identical to σ, except on Y_k where $Y_k\sigma' = Y_k\sigma[r\theta]_{q_k}$. As in the first

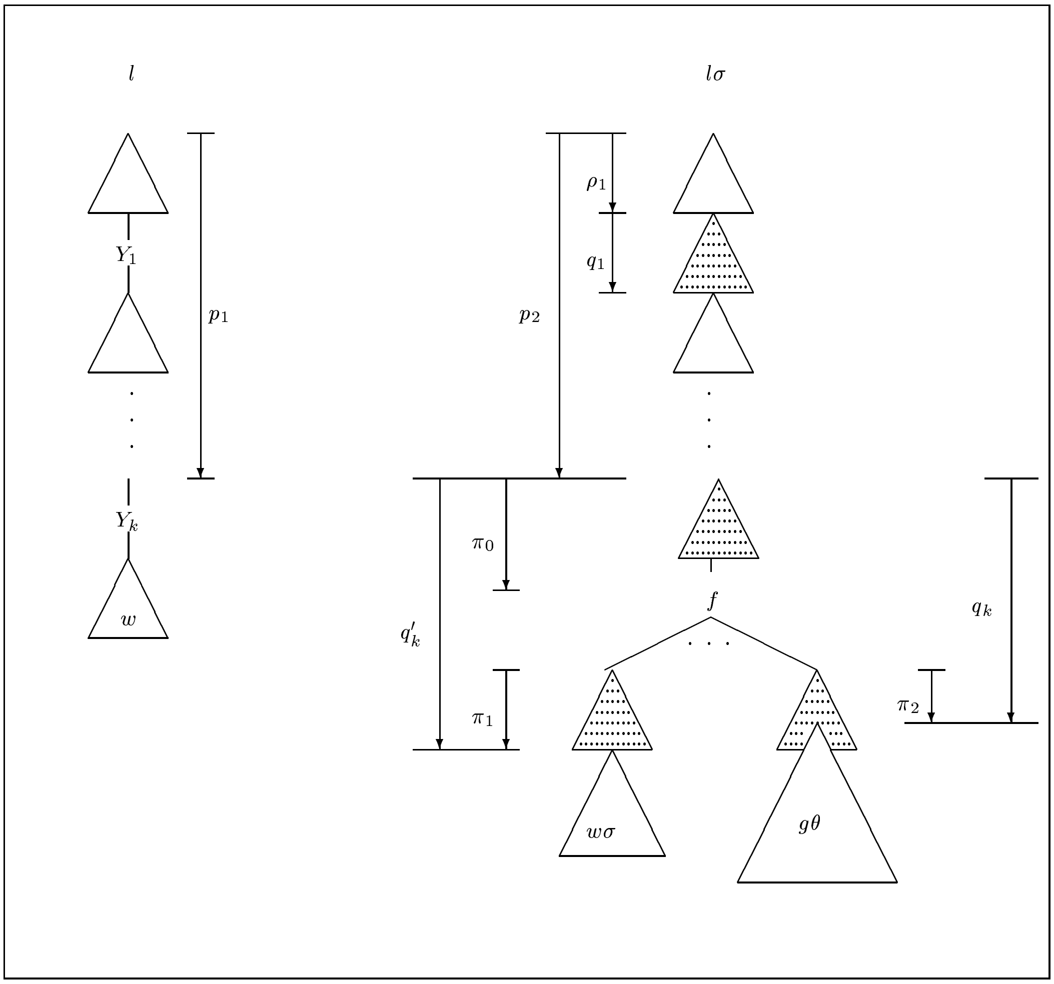

Figure 6. Overlap inside a second-order variable. The disjoint case.

case, there are again two possible situations: either $\sigma' \models \phi$ and, as in the case 1.1, there is a rewrite proof of $s = u$. Or else $\sigma' \not\models \phi$ and **Deduce 4** applies. Indeed, in order to show the satisfiability of the formula of condition $4a.$, choose for X_2 the context $l\sigma|_{p_2}[\circ]_{\pi_0}$, for X_3 the context $l\sigma|_{p_2 \cdot \pi_0 \cdot i}[\circ]_{\pi_1}$ and for X_4 the context $l\sigma|_{p_2 \cdot \pi_0 \cdot j}[\circ]_{\pi_2}$.

$\square$

Note that some simplification steps have been incorporated in the deduction rules. This is because we need to keep a property of the terms which will be very important in the last section.

2.2. Simplification Rules

Up to now, the rewrite relation is only defined on ground terms. If we want to design simplification rules, we have to lift the rewrite relation to non-ground terms. There are a lot of ways for this lifting process (Kirchner *et al.*, 1990). We do not intend to contribute on this problem. So, we choose here a simple way for defining simplifications:

Definition 2.2. *A* constrained term *is a pair $\phi : s$ where ϕ is a solved membership constraint and s is a term.*

Definition 2.3. *Given a set $\mathcal{R}$ of constrained rules, the rewrite relation $\rightarrow_{\mathcal{R}}$ is defined on constrained terms by:*

$$\phi : s \xrightarrow[\psi:l\rightarrow r]{p,\theta} \phi : s' \quad \textit{iff} \quad s|_p = l\theta,\ s' = s[r\theta]_p \textit{ and } \theta \wedge \phi \models \psi$$

Note that, with such a definition, a constrained term can only be simplified in a term associated with the same constraint. This definition is consistent with the classical definition (if ϕ is empty we find again the classical notion of rewriting). It is also consistent with classical notions of order-sorted rewriting. Finally, this rewrite relation is monotonic and stable by all ground substitutions which satisfy the constraint.

A possible generalization would be to allow simplifications where θ depends on the solution of ϕ. More precisely, we could say that $\phi : s$ can be simplified at the position p by $\psi : l \rightarrow r$ if

$$\text{For all } \sigma, \qquad \sigma \models \phi \qquad \Rightarrow \qquad \sigma \models \exists Var(l) :\ s|_p = l \wedge \psi.$$

However, with such a definition, a constrained term would not be (in general) simplified in a single constrained term, but in a finite set of constrained terms. Indeed, assuming that $\phi : s$ can be simplified at position p by $\psi : l \rightarrow r$, the result of the simplification would be the terms $\phi_i : s[r]_p\tau_i$ where the constraints $\phi_i \wedge \tau_i$ are a complete set of solved forms for $\phi \wedge l = s|_p$.

Even, this last definition could be generalized so as to allow *partial simplifications* i.e. to split the constraint ϕ into two parts, only one of which can be simplified... We chose a solution in which a simplification step does not complicate the problem.

Following this discussion, we give the simplification rules in figure 7. The conditions of the simplifications, e.g. $\theta \wedge \phi \models \psi$ are decidable, as an easy consequence of the results of the next section. Indeed, for example, deciding the last condition amounts to decide the validity of $\neg\phi \vee \psi\theta$. We use an ordering on constrained terms in the condition of **Collapse**; this will be explained in the next paragraph.

2.3. Orientation

We also use the classical **Orient** rule. We do not address here the problem of orderings on second order terms. In our framework, we could use, e.g. a recursive path ordering where any second order variable is not comparable with any other symbol. It is possible to improve such an ordering and extend it to constrained terms, defining $\phi : s > \phi : t$ iff, for every solution σ of ϕ, $s\sigma > t\sigma$. However, we do not know yet whether such conditions are decidable. (For inequations alone this has been proved recently decidable when $>$ is a recursive path ordering (Comon, 1990; Jouannaud & Okada, 1991).)

Delete

$$\frac{E \cup \{\phi : s = t\}, \mathcal{R}}{E, \mathcal{R}}$$

If $\phi \wedge s \neq t$ is not satisfiable.

Simplify

$$\frac{E \cup \{\phi : s = t\}, \mathcal{R}}{E \cup \{\phi : s' = t\}, \mathcal{R}}$$

If $\phi : s \to_{\mathcal{R}} \phi : s'$

Compose

$$\frac{E, \mathcal{R} \cup \{\phi : l \to r\}}{E, \mathcal{R} \cup \{\phi : l \to r'\}}$$

If $\phi : r \to_{\mathcal{R}} \phi : r'$

Collapse

$$\frac{E, \mathcal{R} \cup \{\phi : l \to r\}}{E \cup \{\phi : s = r\}, \mathcal{R}}$$

If $\phi : l \xrightarrow[\psi : g \to d]{p, \theta} \phi : s$ where $\psi : g \to d \in \mathcal{R}$ and, either $p \neq \Lambda$ or else θ is not a renaming or else $\phi : r > \phi : d\theta$.

Figure 7. Simplification Rules

2.4. Refutational Completeness

Finally, we rely on (Dershowitz & Jouannaud, 1990) for the notions of *fairness*, *completion procedure*,... Because of our critical pair lemma, we get results similar to those of (Dershowitz & Jouannaud, 1990), in particular the refutation completeness theorem for completion procedures that do not fail; the proof is carried out using the same proof ordering (discarding the constraints in the interpretation). The only difference is the critical pair lemma which states that critical peaks can be replaced with proofs of the form $\xrightarrow{*} \underset{\phi : s = t}{\longleftrightarrow} \xleftarrow{*}$ where $\phi : s = t$ is a consequence of a deduction rule, instead of stating that critical peaks can be replaced with $\underset{s = t}{\longleftrightarrow}$ where $s = t$ is a critical pair. But, in any case, the new proof is simpler than the critical peak. We do not give details here because there is no new idea and this development is straightforward.

It is also possible to design an unfailing completion procedure with similar rules, and to get a refutation completeness theorem. We do not give more details, because our main contribution is the critical pair lemma: replacing classical deduction rules with the rules of figure 2 leads to results similar to the classical ones.

Example 2.4. *Let us show how our completion procedure works on the example given in introduction.*

$$\mathcal{R} = \left\{ \begin{array}{lccc} x \in f^+(g(f^*(a))) : & f(x) & \to & a \\ & g(x) & \to & h(x, x) \end{array} \right.$$

$$\mathcal{R} \vdash_{\textbf{Deduce 2}} \; x \in f^*(a) \wedge X \in ff^+ : \; a = X(h(x, x))$$

but it is also possible to apply **Deduce 2**, *overlapping the first rule in itself, which leads to the equation*

$$Y \in f^+ : \ Y(a) = a$$

Orienting the two last equations into rules, we get a convergent system because the only deduction rules that are applicable lead to renamings of the last rules.

2.5. Termination issues

As pointed out by A. Werner, our completion rules do not terminate in some simple situations such as

Example 2.5. *The signature is given by:*

$$\left\{ \begin{array}{rrcl} & \mathsf{nat} & \to & \mathsf{int} \\ & \mathsf{int} & \to & \mathsf{real} \\ 0 : & & \to & \mathsf{nat} \\ f : & \mathsf{nat} & \to & \mathsf{nat} \\ sq : & \mathsf{int} & \to & \mathsf{nat} \\ \star : & \mathsf{int} \times \mathsf{int} & \to & \mathsf{int} \end{array} \right.$$

And there is a single rule

$$x \in \mathsf{int} : \ \ sq(x) \to x \star x$$

Overlapping the rule into itself, we can apply **Deduce 2** *and get*

$$X(sq(x)) \in \mathsf{int} \wedge x \in \mathsf{int} \wedge X(x \star x) \notin \mathsf{int} : \ \ sq(X(x \star x)) = X(x \star x) \star X(x \star x)$$

And the constraint is satisfiable. For, consider $\{X \mapsto f(\_)\}$*:* $f(x \star x) \notin \mathsf{int}$ *since* $x \star x \notin \mathsf{nat}$.

Next, more overlaps can be considered, leading to a non-terminating completion.

This kind of divergent behaviour cannot be avoided in our approach since the interpretation domain is fixed, hence we cannot infer properties such as $x \in \mathsf{int} \Rightarrow x \star x \in \mathsf{nat}$. This might be considered as a drawback w.r.t. A. Werner's approach (Werner, 1993) or Kirchner's approach (Hintermeier *et al.*, 1994).

Consider however that our completion method is an extension of classical completion methods: if the system is sort-decreasing (or unsorted) and if the completion terminates (in any of the versions of completion), then our completion will terminate yielding the same canonical system. In the above example, the rewrite system is of course not sort decreasing.

There are also examples of the converse situation where, thanks to our point of view, completion terminates in an obvious way whereas it doesn't necessary with other approaches, precisely because there are inferences on the sort structure.

Example 2.6. *Let the signature be given by*

$$\left\{ \begin{array}{rccl} f : & q & \to & q \\ f : & q' & \to & q' \\ a : & & \to & q \\ b : & & \to & q' \end{array} \right.$$

and the rewrite system contains a single rule

$$a \to b$$

Then there is no applicable deduction rule: the system is convergent. The methods of (Werner, 1993; Hintermeier et al.*, 1994) however require to deduce informations on the sort structure which may lead to a divergent behaviour. (On this particular example, the method of (Hintermeier* et al.*, 1994) doesn't diverge, but it should not be too complicated to build an appropriate example).*

3. Restricting the constraints that have to be considered

The terms occurring while running the completion procedure enjoy the following property: if $X \in \mathcal{CX}$ occurs twice in t at positions p and q, then $t|_p \equiv t|_q$. Roughly, we want to show here that, if this is an invariant of the constraint solving, then it is also an invariant of the completion.

DEFINITION 3.1. *A set of terms* $\{s_1, \dots, s_n\}$ *has the* ST-property *if*

$$\forall i, j \in \{1, \dots, n\}, \forall X \in \mathcal{CX}, \forall p \in Pos(s_i), \forall q \in Pos(s_j), \forall t, u \in T(F, \mathcal{X}, \mathcal{CX}),$$
$$(s_i|_p \equiv X(t) \wedge s_j|_q \equiv X(u)) \Rightarrow t \equiv u$$

Let $WT(F, \mathcal{X}, \mathcal{CX})$ be the set of terms which satisfy the above property. They may be viewed as DAGS which are linear w.r.t. the second-order variables. An equation (or a rule, or a set of rules) has the ST property if the sequence of terms occurring in it has the ST property.

Assuming that, when a simplification rule is applied at an inner position of $X(w)$ where $X \in \mathcal{CX}$, then it is also applied at the same position for every other occurrence of $X(w)$ (which corresponds to a fair control), then the ST property is kept during completion as shown by the following lemma. In order to be precise, we denote $\vdash$ the relation on pairs $(E, \mathcal{R})$ defined as follows: $E, \mathcal{R} \vdash E', \mathcal{R}'$ if one of the following holds

(i) $\mathcal{R} = \mathcal{R}'$ and $E' = E \cup \{\phi' : l[d]_p\sigma \downarrow = r\sigma \downarrow\}$ where $\phi' : l[d]_p\sigma = r\sigma$ is obtained from $\mathcal{R}$ using **Deduce 1** and $\downarrow$ is the normal form w.r.t. $\psi : g \to d$. More precisely, E' is obtained by applying first **Deduce 1**. Then, if there is a variable $X \in \mathcal{CX}$ along the path p which also occur in r, then, for all occurrences of a term $X(s)$ in l, r, $X(s)\sigma$ is reduced using $\psi : g\sigma \to d\sigma$. Hence $E', \mathcal{R}'$ is obtained from $E, \mathcal{R}$ by applying one deduction rule and then k times **Simplify**.

(ii) or $E', \mathcal{R}'$ is obtained from $E, \mathcal{R}$ by applying one of the deduction rules **Deduce 2**, **Deduce 3**, **Deduce 4**.

(iii) or $E', \mathcal{R}'$ is obtained from $E, \mathcal{R}$ by applying **Delete** or **Orient** or one of the simplification rules (**Simplify**, **Collapse**, **Compose**) at a position p of s (resp. l) such that there is no prefix q of p such that $s(q) \in \mathcal{CX}$ (resp. $l(q) \in \mathcal{CX}$).

(iv) or $E = E_0 \cup \{\phi : s = t\}$, $E' = E_0 \cup \{\phi : u = v\}$, $\mathcal{R} = \mathcal{R}'$. $s = t \in ST(F, \mathcal{X}, \mathcal{CX})$ and $\phi : s \xrightarrow[\psi:g\to d]{p_1 \cdot q, \sigma} \dots \xrightarrow[\psi:g\to d]{p_n \cdot q, \sigma} \phi : u\ ,\ \phi : t \xrightarrow[\psi:g\to d]{q_1 \cdot q, \sigma} \dots \xrightarrow[\psi:g\to d]{q_m \cdot q, \sigma} \phi : v\ ,\ (n \geq 1,\ m \geq 0)$ where $p_1, \dots, p_n$ are all positions in s of a second order variable X and $q_1, \dots, q_m$ are all positions of X in t. $\psi : g \to d$ is a rule of $\mathcal{R}$. It is also assumed that there is no occurrence of a second order variable along the path q. (This corresponds to $n+m$ applications of **Simplify**).

(v) or $\mathcal{R} = \mathcal{R}_0 \cup \{\phi : l \to r\}$, $\mathcal{R}' = \mathcal{R}_0 \cup \{\phi : l \to u\}$, $E = E'$, $r \in ST(F, \mathcal{X}, \mathcal{CX})$ and $\phi : r \xrightarrow[\psi:g\to d]{p_1\cdot q,\sigma} \dots \xrightarrow[\psi:g\to d]{p_n\cdot q,\sigma} \phi : u$ $(n \geq 1)$ where $p_1 \cdot q, \dots, p_n \cdot q$ are all positions of some second-order variable X in r which does not occur in l. $\psi : g \to d$ is a rule of $\mathcal{R}$. It is also assumed that there is no occurrence of a second order variable along the path q. (This corresponds to n applications of **Compose**.)

(vi) or $\mathcal{R} = \mathcal{R}_0 \cup \{\phi : l \to r\}$, $\mathcal{R}' = \mathcal{R}_0$, $E' = E \cup \{\phi : u = v\}$, $l \to r \in ST(F, \mathcal{X}, \mathcal{CX})$ and $\phi : l \xrightarrow[\psi:g\to d]{p_1\cdot q,\sigma} \dots \xrightarrow[\psi:g\to d]{p_n\cdot q,\sigma} \phi : u$, $\phi : r \xrightarrow[\psi:g\to d]{q_1\cdot q,\sigma} \dots \xrightarrow[\psi:g\to d]{q_m\cdot q,\sigma} \phi : u$, $(n \geq 1$, $m \geq 0)$ where $p_1, \dots, p_n$ are all positions of some second-order variable X in l and $q_1, \dots, q_m$ are all positions of X in r and either $p_1 \neq \Lambda$ or else σ is not a renaming or else $\phi : r > \phi : d\theta$. $\psi : g \to d$ is a rule of $\mathcal{R}$. It is also assumed that there is no occurrence of a second order variable along the path q. (This corresponds to an application of **Collapse** followed by $m+n-1$ applications of **Simplify**.)

The following lemma shows that, with some assumptions on constraint solving which actually holds true with the procedure described in the second part, the ST property is an invariant of the completion.

LEMMA 3.2. *Assume that for each equation $s = t \in E$ and each rule $l \to r \in \mathcal{R}$, $\{s, t\}$ and $\{l, r\}$ have the ST property. Assume moreover that for every constraint $\exists \vec{x} \exists \vec{X}. u_1 = v_1 \wedge \dots \wedge u_n = v_n \wedge m$ where $u_1, \dots, u_n, v_1, \dots, v_n$ are terms in $T(\mathcal{F}, \mathcal{X}, \mathcal{CX})$ such that $\{u_1, \dots, u_n, v_1, \dots, v_n\}$ have the ST property and m is a membership constraint, its solved form $\exists \vec{x'} \exists \vec{X'}. \sigma \wedge m'$ is such that $\{u_1\sigma, \dots, u_n\sigma, v_1\sigma, \dots, v_n\sigma\}$ has the ST property. Then, $E, \mathcal{R} \vdash E', \mathcal{R}'$ implies that each equation of R' and each rule of $\mathcal{R}'$ has the ST property.*

PROOF. Let us investigate all possible ways of constructing $E', \mathcal{R}'$.

(i) Assume first that $E', \mathcal{R}'$ is obtained by applying **Deduce 1** and a number of **Simplify**. (See the above definition of $\vdash$). $\mathcal{R}' = \mathcal{R}$ and $E' = E \cup \{\phi' : l[d]_p\sigma \downarrow = r\sigma \downarrow\}$. $l|_p = g$ has the ST property, hence $l|_p\sigma = g\sigma$ by hypothesis. Now consider a variable $X \in \mathcal{CX}$ occurring in $l[d]_p\sigma = r\sigma$. First note that $l[d]_p\sigma \equiv l\sigma[d\sigma]_p$ since a variable occuring in l along the path p cannot occur in $l|_p$ because of the ST-property. First, we show that $\{l\sigma, r\sigma, d\sigma, g\sigma\}$ has the ST property. We assume that $l \to r$ on one hand and $g \to d$ on the other hand do not share any variable. If X is not a variable of $l|_p, g$ but occurs in $l|_p\sigma, g\sigma$, then we may assume that X does not occur in l, g, d, r and the ST-property of $l|_p\sigma, g\sigma$ implies that $\{l\sigma, g\sigma, d\sigma, r\sigma\}$ has the ST property. Now, if X occurs both in $l|_p, g$ and in $l|_p\sigma, g\sigma$, then every occurrence of X in l, r, g, d is in a term $X(s)$ and the hypothesis on $l|_p\sigma, g\sigma$ implies that, if $x\sigma \equiv t[X(s')]$ (resp $Y\sigma \equiv C[X]$), then $s' \equiv s\sigma$ (resp. Y only occurs in $l|_p, g$ with argument s). It follows again that $\{l\sigma, g\sigma, r\sigma, d\sigma\}$ has the ST property. Finally, if X does neither occur in $l|_p, g$ nor in $l|_p\sigma, g\sigma$, then all occurrences of X in $l\sigma, g\sigma, r\sigma, d\sigma$ are in a term $X(s')$ such that $s' \equiv s\sigma$ and $X(s)$ is a subterm of l, g, r, d.
Now, consider the equation $l[d]_p\sigma \downarrow = r\sigma \downarrow$. If X does not occur along the path p, the ST property of $\{l[d]_p\sigma, r\sigma\}$ follows from the ST-property of $\{l\sigma, g\sigma, r\sigma, d\sigma\}$. Then replacing *all* occurrences of $g\sigma$ with $d\sigma$ in $l[d]_p\sigma$ and $r\sigma$ does not dfestroy the

property. Finally, if X occurs along the path p with an argument s, then $s \equiv C[d]_q$ and all occurrences of X in $l[d]_p\sigma \downarrow = r\sigma \downarrow$ are in terms $X(C[d\sigma]_q)$, hence $l[d]_p\sigma \downarrow = r\sigma \downarrow$ has again the ST property.

(ii) If $E', \mathcal{R}'$ is obtained from $E, \mathcal{R}$ by applying **Deduce 2** or **Deduce 4**, then the property is straightforward from the hypotheses: there is only a substitution and no replacement; substitions with terms having the ST-property preserves this property.

(iii) If $E', \mathcal{R}'$ is obtained from $E, \mathcal{R}$, by applying **Deduce 3**, then the proof is similar to the **Deduce 1** case. The main difference is that simplification is here part of the rule since we know in advance that there is indeed a variable of $\mathcal{CX}$ above the application of $g \to d$.

(iv) If $E', \mathcal{R}'$ is obtained by **Orient** or **Delete**, then the property is obvious.

(v) If $E', \mathcal{R}'$ is obtained by applying a simplification rule, then it follows from the above definition of $\vdash$: if $X \in \mathcal{CX}$ occurs twice in an equation (or a rule), say in the term t_1 at position p and in the term t_2 at position q, then any simplification at a suffix position of p will will be followed by a simplification at the corresponding suffix position of q, resulting in identical subterms. □

The constraint solving rules and their termination proof will be considered in a second part of the paper.

Ackowledgments

I thank Andreas Werner who found an important oversight (the rule **Deduce 4**) and a mistake (the WT-property was a property of terms instead of equations) in an earlier version of this paper. I also thank two anonymous referees for their valuable comments.

References

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

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

Chen, H., & Hsiang, Jieh. 1991. Logic Programming with recurrence domains. *In:* (Albert *et al.*, 1991).

Comon, Hubert. 1990. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, **1**(4), 387–411.

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

Dershowitz, Nachum, & Jouannaud, Jean-Pierre. 1990. Rewrite Systems. *Pages 243–309 of:* van Leeuwen, J. (ed), *Handbook of Theoretical Computer Science*, vol. B. North-Holland.

Farmer, W. 1988. A unification algorithm for second order monadic terms. *Annals of Pure and Applied Logic*, **39**, 131–174.

Futatsugi, Kokichi, Goguen, Joseph, Jouannaud, Jean-Pierre, & Meseguer, Jose. 1985. Principles of OBJ2. *In: Proc. 12th ACM Symp. on Principles of Programming Languages, New Orleans.*

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

Gramlich, B. 1988. *Unification of Term Schemes. Theory and Applications.* Research Report SR-88-18. SEKI, Univ. Kaiserslautern, Germany.

Hintermeier, C., Kirchner, Hélène, & Kirchner, Claude. 1994. Dynamically Typed Computations for Order-Sorted Equational Presentations. *Pages 450–461 of:* Abiteboul, Serge, & Shamir, Eli (eds), *21th International Colloquium on Automata, Languages and Programming.* Lecture Notes in Computer Science, vol. 820. Jerusalem, Israel: Springer-Verlag.

Huet, Gérard. 1980. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the ACM*, **27**(4), 797–821.

Jouannaud, Jean-Pierre, & Okada, Mitsuhiro. 1991. Satisfiability of systems of ordinal notations with the subterm property is decidable. *In:* (Albert *et al.*, 1991).

Kirchner, Claude, Kirchner, Hélène, & Meseguer, José. 1988. Operational Semantics of OBJ-3. *In: Proc. 15th Int. Coll. on Automata, Languages and Programming, LNCS 317.* Tampere: Springer-Verlag.

Kirchner, Claude, Kirchner, Helene, & Rusinowitch, Michaël. 1990. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, **4**(3), 9–52. Special issue on automatic deduction.

Kirchner, Hélène. 1989. Schematization of infinite sets of rewrite rules generated by divergent completion process. *Theoretical Computer Science*, **67**(2–3), 303–332.

Knuth, Donald E., & Bendix, Peter B. 1970. Simple word problems in universal algebras. *Pages 263–297 of:* Leech, J. (ed), *Computational Problems in Abstract Algebra.* Pergamon Press.

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

Schmidt-Schauß, Manfred. 1988 (Apr.). *Computational Aspects of an Order-Sorted Logic with Term Declarations.* Ph.D. thesis, Univ. Kaiserslautern.

Smolka, Gert, Nutt, W., Goguen, J. A., & Meseguer, José. 1989. Order-Sorted Equational Computation. *Chap. 10, pages 297–367 of:* Aït-Kaci, Hassan, & Nivat, Maurice (eds), *Resolution of Equations in Algebraic Structures, Volume 2, Rewriting Techniques.* New York, N.Y.: Academic Press.

Toyama, Y. 1988. Confluent Term Rewriting Systems with membership Conditions. *In: Proc. 1st Int. Workshop on Conditional Term Rewriting Systems, Orsay, LNCS 308.* Springer-Verlag.

Werner, Andreas. 1993. A semantic approach to order-sorted rewriting. *In:* Kirchner, C. (ed), *Proc. 5th Rewriting Techniques and Applications, Montréal, LNCS 690.*