Narrowing directed by a graph of terms

Jacques Chabin and Pierre Réty

LIFO & LRI, Dépt. math-info, Université d'Orléans
BP 6759, 45067 Orléans cedex 2, France
E-mail : {chabin, rety}@univ-orleans.fr

Abstract

Narrowing provides a complete procedure to solve equations modulo confluent and terminating rewriting systems. But it seldom terminates. This paper presents a method to improve the termination. The idea consists in using a finite graph of terms built from the rewriting system and the equation to be solved, which helps one to know the narrowing derivations possibly leading to solutions. Thus, the other derivations are not computed. This method is proved complete. An example is given and some improvements are proposed.

1 Introduction

Solving equations (unification) modulo term rewriting systems is an important problem, which is necessary to combine functional and logic programming, for example as in EQLOG langage [Goguen-Meseguer-86]. In this framework, completeness and termination of unification methods are desirable. Indeed, during a clause superposition attempt, it is interesting to detect the unsatisfiability of the equation to be solved, which implies that the superposition is not possible. Conditional completion procedures also need methods that solve the equations appearing in the conditions. If such an equation is unsatisfiable, the conditional rewrite rule containing it can be deleted, because this rule can't be applied. And to detect that, a complete and terminating procedure is necessary.

This paper presents a method that improves the termination of basic narrowing. Basic narrowing gives a complete set of solutions modulo confluent and terminating rewriting systems, by computing all the narrowing derivations issued from the equation to be solved [Hullot-80]. Actually, only the derivations that lead to a syntactically unifiable equation give solutions. Thus, the others are useless. Our idea consists in using this fact to prune useless paths from the search tree.

Let's look at a simple example. Consider the following confluent and terminating rewriting system :

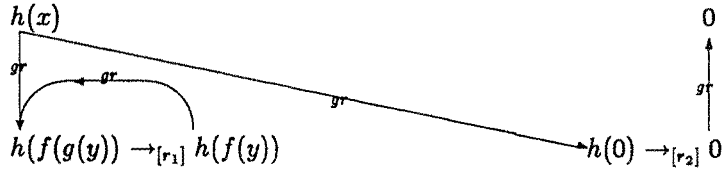
$$\begin{array}{lrcl} r_1: & h(f(g(y))) & \rightarrow & h(f(y)) \\ r_2: & h(0) & \rightarrow & 0 \end{array}$$

We want to solve the equation $h(x) \doteq_R 0$. By using basic narrowing, there are two derivations:

$$h(x) \doteq_R 0 \rightsquigarrow_{[r_1, x \mapsto f(g(x))]} h(f(x)) \doteq_R 0 \rightsquigarrow_{[r_1, x \mapsto g(x)]} h(f(x)) \doteq_R 0 \rightsquigarrow \ldots$$
$$\rightsquigarrow_{[r_2, x \mapsto 0]} 0 \doteq_R 0$$

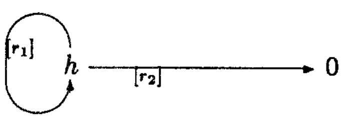
The first derivation does not terminate and gives no solution. The second gives the solution $(x \mapsto 0)$.

To improve the termination, we build a graph (called graph of terms) whose nodes are left-hand-sides and right-hand-sides of the rules, as well as the sides of the equation to be solved; and whose edges are the rewrite arrows $\rightarrow$ and arrows $\xrightarrow{gr}$ (gr means graph) between the syntactically unifiable terms (their variables are implicitly renamed to be disjoint):



Now we apply the only rewrite rules that occur along the paths going from $h(x)$ to 0. In this graph, there is only one (by using r_2). Then the method provides the solution $(x \mapsto 0)$ and terminates. In general, the graph of terms is more complicated because some arrows may appear between subterms, and the two sides of the equations can be narrowed.

Our method is an extension of [Sivakumar-Dershowitz-87] and [Dershowitz-Sivakumar-87], where a graph of top symbols is used. Their procedure applied on the above example does not terminate, because if we only consider the top symbols, the graph becomes:



and there are infinitely many paths going from h to 0. In order to get a finite graph, we will suppose the rewriting system is finite.

After introducing preliminary notions in section 2, we present the graph building algorithm as inference rules in section 3. A narrowing procedure is given in section 4 by inference rules and using equation systems. Its completeness is proved. Two variants of an example (about lists) are shown : one which terminates (in section 5) whereas it wouln't, if some known narrowing optimizations are used; the other which doesn't terminate (in section 6), and we show how the method can be improved.

2 Preliminary notions

We assume the reader is familiar with the standard definitions of one-sorted terms, substitutions, equations, rewriting systems, complete sets of unifiers (see [Dershowitz-Jouannaud-90]).

We denote constants by a, b; function symbols by $f, g, h, \ldots$; variables by $x, y, z, \ldots$; terms by $s, t, \ldots$; occurrences (also called positions) by $u, v, w, \ldots$, and ϵ denotes the top occurrence; substitutions by $\sigma, \theta, \ldots$. The set of variables of a term t is denoted by $V(t)$. We denote by $t|_u$ the subterm of t at occurrence u, and by $t[u \leftarrow s]$ the term obtained by replacing $t|_u$ by s in t. $D(t)$ is the set of the occurrences of t, and $O(t)$ is the set of non-variable occurrences. In the whole paper, we consider a rewriting system R which contains finitely many rewrite rules. The rewriting relation is denoted by $\rightarrow$.

The narrowing relations defined below don't normalize resulting terms, in contrast with [Fay-78] and [Réty-80]. The definitions and the theorem come from [Hullot-80].

Definition 1 We say t is narrowable into t' and we write $t \rightsquigarrow_{[u,l\rightarrow r,\sigma]} t'$ if $t|_u$ and l are unifiable by the most general unifier σ, with $u \in O(t)$, and $t' = \sigma(t[u \leftarrow r])$. The relation $\rightsquigarrow$ is called **narrowing.** $\diamond$

Definition 2 A narrowing derivation is said to be **basic** if at each step the narrowing occurrence does not belong to a part brought by a substitution in a previous step. Formally the derivation $t_1 \rightsquigarrow_{[u_1,l_1\rightarrow r_1,\sigma_1]}\rightsquigarrow \ldots \rightsquigarrow_{[u_n,l_n\rightarrow r_n,\sigma_n]} t_{n+1}$ is basic if there exist some sets $U_2, \ldots, U_{n+1}$ of occurrences of $t_2, \ldots, t_{n+1}$ respectively such that for all $i \in \{2, \ldots, n\}$, $u_i \in U_i$ and

$$U_{i+1} = (U_i - \{v \in U_i / \exists w, v = u_i.w\}) \cup \{u_i.v / v \in O(r_i)\}$$

$\diamond$

The equations to be solved modulo the rewriting system R are written in the form $t \doteq_R t'$, and are considered as new terms in the signature.

Theorem 1 *If the rewriting system R is confluent and terminating, the set of substitutions θ such that*

- *there exists a basic narrowing derivation issued from $t_0 \doteq_R t'_0$*

$$t_0 \doteq_R t'_0 \rightsquigarrow_{[\sigma_1]} t_1 \doteq_R t'_1 \rightsquigarrow \ldots \rightsquigarrow_{[\sigma_n]} t_n \doteq_R t'_n$$

  such that t_n and t'_n are unifiable by the most general unifier β

- *$\theta = \beta.\sigma_n \ldots \sigma_1$ and θ is normalized on $V(t_0) \cup V(t'_0)$,*

is a complete set of R-unifiers of t_0 and t'_0.

3 Building the graph of terms

3.1 Examples

In this section some graphs are shown to explain the method. They are not completely built, i.e. some arrows may be missing.

Consider a basic narrowing derivation issued from a term t_0. In each narrowing step $t_i \rightsquigarrow_{[u_i,l_i\rightarrow r_i,\sigma_i]} t_{i+1}$, the occurrence u_i does not belong to a part brought by a substitution.

Then it comes from t_0 or from a right-hand-side of a rewrite rule. Moreover $t_i|_{u_i}$ and the left-hand-side l_i are unifiable. Therefore, we can summarize this basic narrowing derivation by considering only t_0 and the rule sides. We see them as the nodes of a graph, whose edges are rewrite arrows and syntactical unification possibilities. The part brought by narrowing substitutions is omitted. This graph is finite since the set of rewrite rules is. For example, consider the rewrite rule :

$$r:\ h(x,g(y)) \to i(x,y)$$

Then the step :

$$h(f(x_1),y_1) \rightsquigarrow_{[\epsilon,r,(x\mapsto f(x_1),y_1\mapsto g(y))]} i(f(x_1),y)$$

is summarized by the graph :

$$h(f(x_1),y_1) \xrightarrow{gr} h(x,g(y)) \to_{[r]} i(x,y)$$

The arrow $\xrightarrow{gr}$ shows that two terms are unifiable (gr means graph). Since the narrowing substitution is not applied on the terms in the graph, the last term of the narrowing step $i(f(x_1),y)$ is an instance of the term $i(x,y)$ of the graph.

Conversely, if the graph contains $\xrightarrow{gr}$ whenever two terms are unifiable, then it associates a path in the graph with every basic narrowing derivation issued from t_0. Therefore, by considering all the paths that lead to a given term s of the graph, we know all the basic derivations issued from t_0, that may lead to an instance of s.

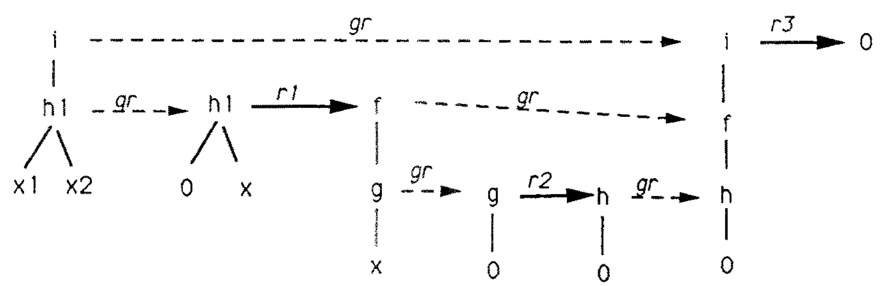
However, some narrowing steps may be applied on subterms. For example, consider the rewriting system :

$$\begin{array}{rrcl} r_1: & h_1(0,x) & \to & f(g(x)) \\ r_2: & g(0) & \to & h(0) \\ r_3: & i(f(h(0))) & \to & 0 \end{array}$$

and the basic narrowing derivation :

$$i(h_1(x_1,x_2)) \rightsquigarrow_{[1,r_1,(x_1\mapsto 0,x_2\mapsto x)]} i(f(g(x))) \rightsquigarrow_{[2,r_2,x\mapsto 0]} i(f(h(0))) \rightsquigarrow_{[\epsilon,r_3,Id]} 0$$

The two first steps are applied on subterms. In order to get in the graph a path at the top from $i(h_1(x_1,x_2))$ to 0, first we add a $\xrightarrow{gr}$ corresponding to the second step, then a $\xrightarrow{gr}$ corresponding to the first step. Obviously, the arrow $\xrightarrow{gr}$ from $i(h_1(x_1,x_2))$ to $i(f(h(0)))$ does not mean these two terms are unifiable, it only means $i(h_1(x_1,x_2))$ can be narrowed only on subterms into an instance of $i(f(h(0)))$. Actually, the arrows $\xrightarrow{gr}$ have two meanings.

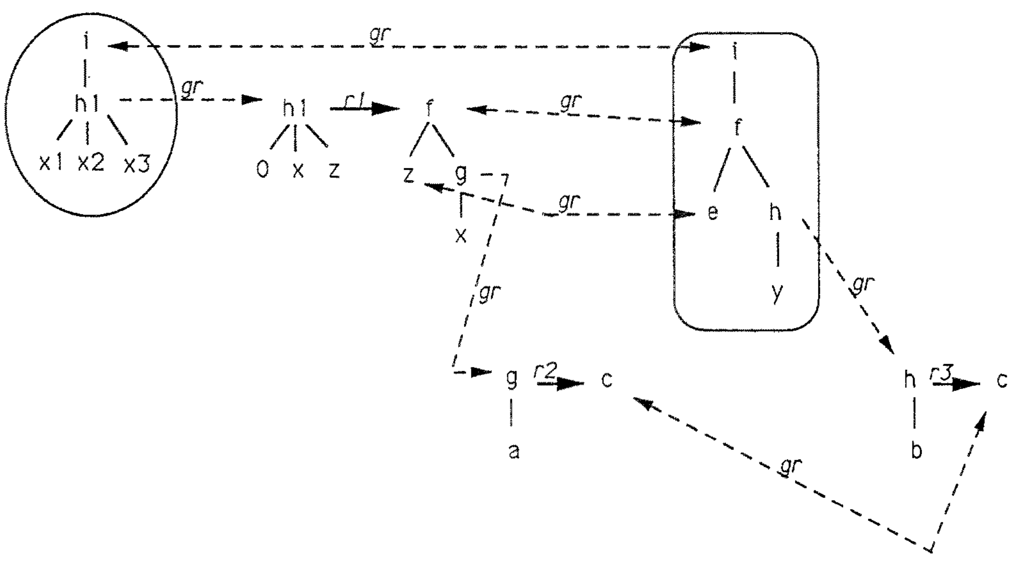


Observe that one of the $\xrightarrow{gr}$ arrows goes from a term to a subterm.

Now, let's try to solve an equation $t \doteq_R t'$. A solution is found whenever $t \doteq_R t'$ is narrowed into a syntactically unifiable equation. Then t and t' can be narrowed into unifiable terms t_2 and t'_2 :

$$t \rightsquigarrow^{*} \rightsquigarrow_{[\epsilon]} t_1 \rightsquigarrow^{*}_{[\neq\epsilon]} t_2$$
$$t' \rightsquigarrow^{*} \rightsquigarrow_{[\epsilon]} t'_1 \rightsquigarrow^{*}_{[\neq\epsilon]} t'_2$$

t_1, t'_1 are instances of right-hand-sides (or of t, t'), and we add the arrow $\overset{gr}{\longleftrightarrow}$ between them in the graph to show that they can be narrowed (possibly by 0 step) only on subterms into two unifiable terms. For example consider the following graph, where the terms to be solved are encercled. Remark that adding a $\overset{gr}{\longleftrightarrow}$ arrow may create another.



3.2 The algorithm

We present the algorithm in a simple form, via inference rules. It could be optimized to avoid useless arrows.

Notations : G denotes the current graph, l is a (sub)-left-hand-side, and r, r' are (sub)-right-hand-sides. Actually r, r', l are both pointers in the graph and terms. We assume r and r' do not point inside the same right-hand-side of the graph (except in section 6). $\longrightarrow^{*}$ denotes a path in the graph that may contain $\overset{gr}{\longrightarrow}$ arrows and rewriting arrows. $\overset{gr}{\longleftrightarrow}^{?}$ denotes zero or one step of $\overset{gr}{\longleftrightarrow}$.

(Add$\overset{gr}{\longrightarrow}$) $\quad \dfrac{G}{G \cup \{r \overset{gr}{\longrightarrow} l\}}$ $\quad$ if r is a variable
or l is a variable
or $r = f(r_1, \ldots, r_n)$, $l = f(l_1, \ldots, l_n)$ and
$\forall i \in \{1, \ldots, n\}, r_i \longrightarrow^{*} l_i \in G$

$$(\mathsf{Add}\overset{gr}{\longleftrightarrow})\quad \frac{G}{G \cup \{r \overset{gr}{\longleftrightarrow} r'\}} \quad \begin{array}{l}\text{if } r \text{ is a variable}\\ \text{or } r' \text{ is a variable}\\ \text{or } r = f(r_1, \dots, r_n),\, r' = f(r'_1, \dots, r'_n) \text{ and}\\ \quad \forall i \in \{1, \dots, n\}, \exists t_i, t'_i /\ r_i \longrightarrow^* t_i \overset{gr}{\longleftrightarrow}{}^? t'_i {}^* \longleftarrow r'_i \in G\end{array}$$

Let t and t' be the terms to be unified modulo the rewriting system R. The algorithm consists in two steps :

- initialization : $G \leftarrow R \cup \{t, t'$ considered as right-hand-sides$\}$
- adding arrows : apply the inferences rules as long as they add new arrows.

The termination is ensured since the graph can't contain infinitely many arrows (recall R is finite).

The graph depends on the terms to be unified. However, if one wants to unify several pairs of terms modulo the same rewriting system, one can avoid building the whole graph several times :

- Initialize the graph by R and add all the arrows.
- For each pair of terms t, t' to be unified, add t, t' into the graph and add all the arrows between t (respectively t') and others (sub)terms. Once having finished dealing with t and t', remove them out of the graph and delete the arrows that arrive at or start from them.

4 The narrowing procedure

We describe here how to use the graph of terms. For that, we present an equational formulation of narrowing, which is not identical to that of [Martelli-etal-87], because we introduce a further equation symbol $\overset{\rightarrow}{=}_R$ in order to avoid some redundancies. We use equational systems, called unificands, as in [Kirchner-84], except that we don't use any multiequations.

We define 5 kinds of equations, and the associated sets of solutions SOL. For terms t, t', consider :

- $t \doteq_R t'$ where $SOL(t \doteq_R t') = \{\theta \text{ normalized } | \theta(t) =_R \theta(t')\}$
- $t \overset{\rightarrow}{=}_R t'$ where $SOL(t \overset{\rightarrow}{=}_R t') = \{\theta$ normalized $|\theta(t) \rightarrow^*_R \theta(t')$ by a basic rewriting derivation$\}$
- $t \doteq t'$ where $SOL(t \doteq t') = \{\theta \text{ normalized } | \theta(t) = \theta(t')\}$
- T where $SOL(T) = \{\theta \text{ normalized}\}$
- F where $SOL(F) = \emptyset$

Let $\wedge, \vee$ be two new associative and commutative symbols of variable arity. The **unificands** are defined by :

- any equation $t \doteq t'$ or $t \overset{\rightarrow}{=}_R t'$ or $t \doteq_R t'$ or F or T is a unificand.
- if $S_1, \dots, S_n$ are unificands, then $\wedge S_1 \dots S_n$ and $\vee S_1 \dots S_n$ are unificands.

A unificand in the form $\wedge S_1 \ldots S_n$ is called conjunctive factor. SOL is extended in the natural way :

$$SOL(\wedge S_1 \ldots S_n) = \cap_{i=1}^{n} SOL(S_i)$$
$$SOL(\vee S_1 \ldots S_n) = \cup_{i=1}^{n} SOL(S_i)$$

The unificands are viewed as flattened terms in a new signature. It is assumed that considered unificands are completely flattened. $\wedge, \vee$ are prefix symbols, but they may be used as infix symbols.

The procedure is given by some inference rules. They are similar to those of [Chabin-90]. Each inference rule preserves the solutions, thus the choice of the equation to be reduced is "don't care". The following property is preserved by the inference rules, up to a variable renaming :

- In an equation $t \doteq_R t'$, the terms t and t' are (sub)-right-hand-sides of the graph.
- in an equation $t \stackrel{\rightarrow}{=}_R t'$, the term t is a (sub)-right-hand-side of the graph, and the term t' is a (sub)-left-hand-side of the graph.

In the inference rules, we distinguish variables and non variable terms. x and y denote variables, while s, t denote non-variable terms. $top(s)$ is the top-symbol of s and $s_1, \ldots, s_n$ are its subterms. $t_1, \ldots, t_n$ are subterms of t and $l \rightarrow r$ is a rewrite rule. Recall that $\stackrel{gr}{\longleftrightarrow}^{?}$ means zero or one step of $\stackrel{gr}{\longleftrightarrow}$. The big symbol $\bigvee$ means there are several conjunctive factors in the line if there exist several paths in the graph satisfying the condition. The symbol $\longrightarrow^*$ denotes a path in the graph which may contain $\stackrel{gr}{\longrightarrow}$ and rewrite arrows $\rightarrow$. If an inference rule creates an empty conjunctive factor, it has to be replaced by T, and if a rule creates an empty disjunctive factor (for none of the conditions is satisfied), it has to be replaced by F. When a rewriting rule is used, its variables must be renamed (if necessary) to avoid conflicts of variables. Note that the graph is only used for the equations whose terms are both non-variable.

The narrowing rules

(Narrow-$\doteq_R$)

$$s \doteq_R t \longrightarrow s_1 \doteq_R t_1 \wedge \ldots \wedge s_n \doteq_R t_n \quad \text{if } s \stackrel{gr}{\longleftrightarrow} t$$
$$\bigvee \; s_1 \stackrel{\rightarrow}{=}_R l_1 \wedge \ldots \wedge s_n \stackrel{\rightarrow}{=}_R l_n \wedge r \doteq_R t \quad \text{if } s \stackrel{gr}{\longrightarrow} l \rightarrow r \longrightarrow^* \stackrel{gr}{\longleftrightarrow}^{?} {}^* \longleftarrow t$$
$$\bigvee \; t_1 \stackrel{\rightarrow}{=}_R l_1 \wedge \ldots \wedge t_n \stackrel{\rightarrow}{=}_R l_n \wedge s \doteq_R r \quad \text{if } s \stackrel{gr}{\longleftrightarrow}^{?} {}^* \longleftarrow r \leftarrow l \stackrel{gr}{\longleftarrow} t$$

$$\begin{matrix} s \doteq_R y \\ y \doteq_R s \end{matrix} \longrightarrow y \doteq s$$
$$\bigvee \; s|_u \doteq l \wedge s[u \leftarrow r] \doteq_R y \quad \text{if } u \in O(s) \text{ and } s|_u \text{ and } l \text{ have the same top symbol}$$

$$x \doteq_R y \longrightarrow x \doteq y$$

(Narrow-$\overset{\rightarrow}{=}_R$)

$$
\begin{array}{lll}
s \overset{\rightarrow}{=}_R t \longrightarrow & s_1 \overset{\rightarrow}{=}_R t_1 \wedge \ldots \wedge s_n \overset{\rightarrow}{=}_R t_n & \text{if } s \xrightarrow{gr} t \\
\bigvee & s_1 \overset{\rightarrow}{=}_R l_1 \wedge \ldots \wedge s_n \overset{\rightarrow}{=}_R l_n \wedge r \overset{\rightarrow}{=}_R t & \text{if } s \xrightarrow{gr} l \rightarrow r \longrightarrow^* t
\end{array}
$$

$$
y \overset{\rightarrow}{=}_R t \longrightarrow \quad y \doteq t
$$

$$
\begin{array}{lll}
s \overset{\rightarrow}{=}_R x \longrightarrow & x \doteq s & \\
\bigvee & s|_u \doteq l \wedge s[u \leftarrow r] \overset{\rightarrow}{=}_R x & \begin{array}{l}\text{if } u \in O(s) \text{ and} \\ s|_u \text{ and } l \text{ have the same top symbol}\end{array}
\end{array}
$$

$$
y \overset{\rightarrow}{=}_R x \longrightarrow \quad y \doteq x
$$

Now, we need some rules to simplify the unificands.

Definition 3 A conjunctive factor C is said to be in **solved form** if $C = F$ or $C = T$ or $C = \wedge\, x_1 \doteq t_1 \ \ldots\ x_n \doteq t_n$ where $\forall i, j \ \ (i \neq j \Longrightarrow x_i \neq x_j)$ and there is no cycle of variable. $\diamondsuit$

The simplification rules

Transforming the conjunctive factors into solved form (well known problem of syntactical unification) :

(Solve-$\doteq$) $\quad \dfrac{\wedge t_1 \doteq t'_1 \ldots t_n \doteq t'_n}{S'}$ where $\left\{ \begin{array}{l} S' \text{ is a conjunctive factor in solved form} \\ \text{and } SOL(S') = SOL(\wedge t_1 \doteq t'_1 \ldots t_n \doteq t'_n) \end{array} \right.$

Deleting the equations that lead to non normalized solutions (they are redundant):

(Del-$\doteq$) $\quad \dfrac{x \doteq t}{F}$ if t is not in normal form

Deleting trivial equations :

(Del-$\doteq_R$) $\quad \dfrac{a \doteq_R a}{T}$ if a is a constant $\qquad$ (Del-$\overset{\rightarrow}{=}_R$) $\quad \dfrac{a \overset{\rightarrow}{=}_R a}{T}$ if a is a constant

Deleting the equations T and F :

(Del-$\vee$-T) $\quad \dfrac{\vee S_1 \ldots S_{i-1} T S_{i+1} \ldots S_n}{T}$ $\qquad$ (Del-$\vee$-F) $\quad \dfrac{\vee S_1 \ldots S_{i-1} F S_{i+1} \ldots S_n}{\vee S_1 \ldots S_n}$

(Del-$\wedge$-T) $\quad \dfrac{\wedge S_1 \ldots S_{i-1} T S_{i+1} \ldots S_n}{\wedge S_1 \ldots S_n}$ $\qquad$ (Del-$\wedge$-F) $\quad \dfrac{\wedge S_1 \ldots S_{i-1} F S_{i+1} \ldots S_n}{F}$

The distributivity rule

Transforming the unificands into disjunctive form :

(Distributivity) $\quad \dfrac{\wedge S_1 \ldots S_n (\vee Q_1 \ldots Q_p)}{\vee (\wedge S_1 \ldots S_n Q_1) \ldots (\wedge S_1 \ldots S_n Q_p)}$

Remark : The narrowing relation defined by these inference rules is basic since the part brought by substitutions is inside the $\doteq$ equations (after applying (Solve-$\doteq$)), which are not narrowed. It can even be left-to-right basic [Herold-86] (or right-to-left) because of the "don't care" choice of the equation to be narrowed.

We use the following strategy to apply the rules.

Definition 4 A derivation $S_0 \longrightarrow \ldots \longrightarrow S_n \longrightarrow \ldots$ is said **fair** if it is created by the following procedure :
while the current unificand contains $\doteq_R$ or $\overset{?}{=}_R$ equations do

- apply the rule (Narrow-$\doteq_R$) on all the $\doteq_R$ equations, and (Narrow-$\overset{?}{=}_R$) on all the $\overset{?}{=}_R$ equations,
- apply the simplifications rules.
- if you wish, transform into disjunctive form by applying (Distributivity), and then apply the simplification rules.

$\diamond$

Transforming the unificand into disjunctive form is necessary to compute the possibly solutions. The loop must not be run infinitely many times without transforming the unificand into disjunctive form.

The above procedure is complete, i.e. any solution is found in a finite time.

Theorem 2 *Let S_0 be a unificand, $\theta_0 \in SOL(S_0)$, and $S_0 \longrightarrow * \ldots \longrightarrow *S_n \longrightarrow \ldots$ a fair derivation issued from S_0, where $S_1, \ldots, S_n, \ldots$ are the unificands in disjunctive form obtained at the end of some loop steps in the procedure.*

Then there exists a unificand S_i of this derivation and a substitution θ_i such that

- *θ_i is solution of a conjunctive factor of S_i that contains no $\doteq_R$ and $\overset{?}{=}_R$ equation*
- *$\theta_i = \theta_0\ [V(S_0)]$*

Proof : See [Chabin-Réty-91]. $\square$

Intuitively, we see that if a term can be narrowed infinitely many times, then there is a cycle in the graph of terms.

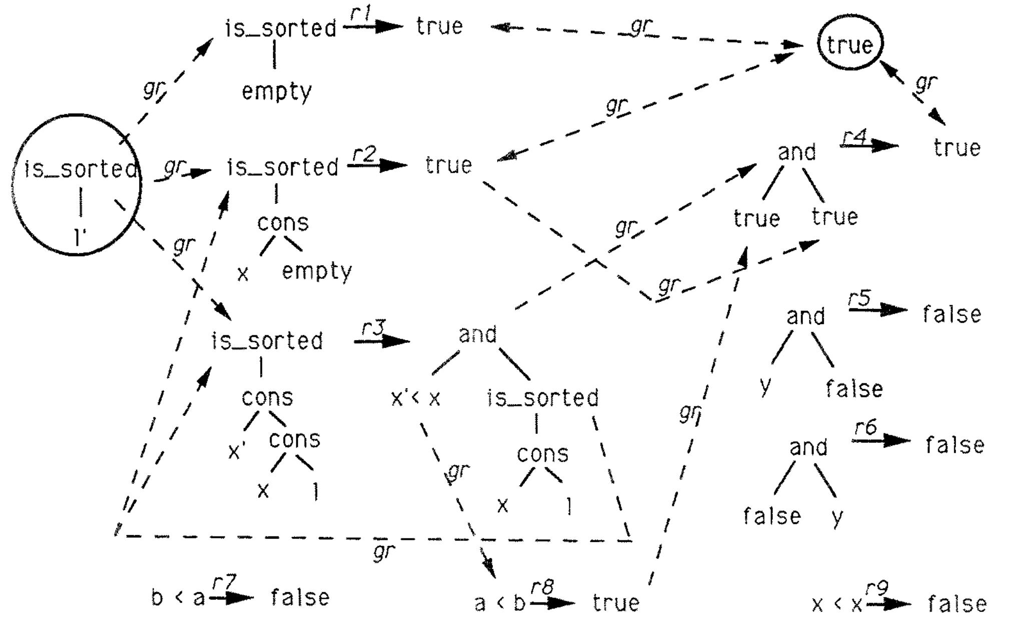
Definition 5 The graph of terms is said to contain a **cycle by subterm** if there are some right-hand-sides $t_1, \ldots, t_n$ and some occurrences $u_1 \in O(t_1), \ldots, u_n \in O(t_n)$, such that there exist paths in the graph of the form $t_1|_{u_1} \longrightarrow^* t_2$, $t_2|_{u_2} \longrightarrow^* t_3$, ..., $t_n|u_n \longrightarrow^* t_1$. $\diamond$

Conjecture 1 *(termination criterion)*
If the graph of terms doesn't have any cycles by subterm, then the above procedure terminates.

Unfortunately, there is a cycle by subterm if a function symbol is recursively defined.

5 Example

Consider a set $\{a, b\}$ (which could be extended) of constants, strictly ordered by $<$, and the lists with the symbols $\{empty, cons\}$, the booleans with $\{true, false, and\}$, and the function *is_sorted* that says whether a list is strictly sorted. The graph is below. Some arrows are missing, only the usefull arrows are indicated. The terms to be unified are encercled.



We solve $is\_sorted(l') \doteq_R true$. We obtain :

1. After narrowing
$(((l' \stackrel{\rightarrow}{=}_R empty) \wedge (true \doteq_R true)) \vee ((l' \stackrel{\rightarrow}{=}_R cons(x_1, empty)) \wedge (true \doteq_R true))$
$\vee((l' \stackrel{\rightarrow}{=}_R cons(x'_2, cons(x_2, l_2))) \wedge ((x'_2 < x_2) \text{ and } is\_sorted(cons(x_2, l_2))) \doteq_R true)))$
2. After simplifying
$((l' \stackrel{\rightarrow}{=}_R empty) \vee (l' \stackrel{\rightarrow}{=}_R cons(x_1, empty)) \vee ((l' \stackrel{\rightarrow}{=}_R cons(x'_2, cons(x_2, l_2))) \wedge$
$((x'_2 < x_2) \text{ and } (is\_sorted(cons(x_2, l_2))) \doteq_R true)))$
3. After narrowing
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge ((x'_2 < x_2) \stackrel{\rightarrow}{=}_R true) \wedge$
$(is\_sorted(cons(x_2, l_2)) \stackrel{\rightarrow}{=}_R true) \wedge (true \doteq_R true)))$
4. After simplifying
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge ((x'_2 < x_2) \stackrel{\rightarrow}{=}_R true) \wedge$
$(is\_sorted(cons(x_2, l_2)) \stackrel{\rightarrow}{=}_R true)))$
5. After narrowing
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \stackrel{\rightarrow}{=}_R a) \wedge (x_2 \stackrel{\rightarrow}{=}_R b) \wedge$
$(true \stackrel{\rightarrow}{=}_R true) \wedge (((cons(x_2, l_2) \stackrel{\rightarrow}{=}_R cons(x_3, empty)) \wedge (true \stackrel{\rightarrow}{=}_R true)) \vee$
$((cons(x_2, l_2) \stackrel{\rightarrow}{=}_R cons(x'_4, cons(x_4, l_4)) \wedge ((x'_4 < x_4) \wedge is\_sorted(cons(x_4, l_4))) \stackrel{\rightarrow}{=}_R true))))$

6. After simplifying and transforming into normal form
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \stackrel{\rightarrow}{=}_R a) \wedge (x_2 \stackrel{\rightarrow}{=}_R b) \wedge$
$(cons(x_2, l_2) \stackrel{\rightarrow}{=}_R cons(x_3, empty))) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \stackrel{\rightarrow}{=}_R a) \wedge (x_2 \stackrel{\rightarrow}{=}_R b) \wedge$
$((cons(x_2, l_2) \stackrel{\rightarrow}{=}_R cons(x'_4, cons(x_4, l_4))) \wedge ((x'_4 < x_4)\ and is\_sorted(cons(x_4, l_4))) \stackrel{\rightarrow}{=}_R true))))$
7. After narrowing
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge$
$(x_2 \stackrel{\rightarrow}{=}_R x_3) \wedge (l_2 \stackrel{\rightarrow}{=}_R empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge (x_2 \stackrel{\rightarrow}{=}_R x'_4) \wedge$
$(l_2 \stackrel{\rightarrow}{=}_R cons(x_4, l_4)) \wedge (x'_4 < x_4 \stackrel{\rightarrow}{=}_R true) \wedge (is\_sorted(cons(x_4, l_4))) \stackrel{\rightarrow}{=}_R true) \wedge (true \stackrel{\rightarrow}{=}_R true)))$
8. After simplifying
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge$
$(x_2 \stackrel{\rightarrow}{=}_R x_3) \wedge (l_2 \stackrel{\rightarrow}{=}_R empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge (x_2 \stackrel{\rightarrow}{=}_R x'_4) \wedge$
$(l_2 \stackrel{\rightarrow}{=}_R cons(x_4, l_4)) \wedge (x'_4 < x_4 \stackrel{\rightarrow}{=}_R true) \wedge (is\_sorted(cons(x_4, l_4))) \stackrel{\rightarrow}{=}_R true)))$
9. After narrowing
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge$
$(x_2 \doteq x_3) \wedge (l_2 \doteq empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge (x_2 \doteq x'_4) \wedge$
$(l_2 \doteq cons(x_4, l_4)) \wedge (x'_4 \stackrel{\rightarrow}{=}_R a) \wedge (x_4 \stackrel{\rightarrow}{=}_R b) \wedge (true \stackrel{\rightarrow}{=}_R true) \wedge (((cons(x_4, l_4) \stackrel{\rightarrow}{=}_R cons(x_5, empty)) \wedge$
$(true \stackrel{\rightarrow}{=}_R true)) \vee ((cons(x_4, l_4) \stackrel{\rightarrow}{=}_R cons(x'_6, cons(x_6, l_6))) \wedge$
$((x'_6 < x_6)\ and\ is\_sorted(cons(x_6, l_6)) \stackrel{\rightarrow}{=}_R true)))$
10. After simplifying
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge$
$(x_3 \doteq b) \wedge (l_2 \doteq empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge (x'_4 \doteq b) \wedge$
$(l_2 \doteq cons(x_4, l_4)) \wedge (x'_4 \stackrel{\rightarrow}{=}_R a) \wedge (x_4 \stackrel{\rightarrow}{=}_R b) \wedge (((cons(x_4, l_4) \stackrel{\rightarrow}{=}_R cons(x_5, empty))) \vee$
$((cons(x_4, l_4) \stackrel{\rightarrow}{=}_R cons(x'_6, cons(x_6, l_6))) \wedge ((x'_6 < x_6)\ and\ is\_sorted(cons(x_6, l_6)) \stackrel{\rightarrow}{=}_R true)))$
11. After narrowing
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge$
$(x_3 \doteq b) \wedge (l_2 \doteq empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge (x'_4 \doteq b) \wedge$
$(l_2 \doteq cons(x_4, l_4)) \wedge (x'_4 \doteq a) \wedge (x_4 \doteq b) \wedge (((x_4 \stackrel{\rightarrow}{=}_R x_5) \wedge (l_4 \stackrel{\rightarrow}{=}_R empty)) \vee ((x_4 \stackrel{\rightarrow}{=}_R x'_6) \wedge$
$(l_4 \stackrel{\rightarrow}{=}_R cons(x_6, l_6)) \wedge (x'_6 < x_6 \stackrel{\rightarrow}{=}_R true) \wedge (is\_sorted(cons(x_6, l_6)) \stackrel{\rightarrow}{=}_R true) \wedge (true \stackrel{\rightarrow}{=}_R true)))$
12. After simplifying
$((l' \doteq empty) \vee (l' \doteq cons(x_1, empty)) \vee ((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge (x'_2 \doteq a) \wedge (x_2 \doteq b) \wedge$
$(x_3 \doteq b) \wedge (l_2 \doteq empty)))$

The procedure terminates, although there is a cycle by subterm in the graph, and we obtain 3 solutions, from which 4 closed solutions can be deduced.

Consider the infinite derivation :

$$is\_sorted(l') \rightsquigarrow_{[r_3]} (x' < x)\ and\ is\_sorted(cons(x, l))$$
$$\rightsquigarrow (x' < x)\ and\ (x'_1 < x_1\ and\ is\_sorted(cons(x_1, l_1))) \rightsquigarrow \ldots$$

It is left-to-right basic, and every term is in normal form. Therefore none among the ordinary, basic, left-to-right basic, normalizing, basic normalizing narrowings terminates.

6 Improvements and conclusion

The previous example works well because most rewriting rules have closed terms as right-hand-side. Unfortunately, if we replace the rule r_4 by $(true\ and\ y) \rightarrow y$, the unificand obtained at step 3 contains the conjunctive factor :

$$((l' \doteq cons(x'_2, cons(x_2, l_2))) \wedge ((x'_2 < x_2) \stackrel{\rightarrow}{=}_R true) \wedge (is\_sorted(cons(x_2, l_2)) \stackrel{\rightarrow}{=}_R y) \wedge (y \doteq_R true)))$$

Then $y \doteq_R true$ is transformed into $y \doteq true$, and $is\_sorted(cons(x_2, l_2)) \stackrel{\rightarrow}{=}_R y$ can be narrowed infinitely many times. Therefore our procedure does not terminate.

Actually, we need to take into account the fact that y is equal to $true$, i.e. by solving $is\_sorted(cons(x_2, l_2)) \stackrel{\rightarrow}{=}_R true$, which terminates. For that, we propose to introduce merging rules, like

$$s \stackrel{\rightarrow}{=}_R x \wedge x \doteq t \longrightarrow s \stackrel{\rightarrow}{=}_R t \wedge x \doteq t$$
$$s \doteq_R x \wedge x \doteq t \longrightarrow s \doteq_R t \wedge x \doteq t$$
$$s \stackrel{\rightarrow}{=}_R x \wedge x \doteq_R t \longrightarrow s \doteq_R t \wedge x \doteq_R t$$

and so on. Thus, we don't have to narrow equations in the form $s \stackrel{\rightarrow}{=}_R x$ or $x \doteq_R t$ if $x \notin s$. So, we hope the method will often terminate.

Building the graph of terms obviously spends a long time. By using a graph of top symbols, as in [Dershowitz-Sivakumar-87], it would spend less time. But it would terminate less often.

7 References

[Chabin-90] Jacques Chabin. Surréduction dirigée par un graphe d'opérateurs. Formulation équitable par des systèmes d'équations et implantation. Rapport de DEA, université d'Orléans, Septembre 1990. (In french).

[Chabin-Réty-91] J. Chabin and P. Réty. Narrowing directed by a graph of terms (including proofs). Internal report 91-1, LIFO, Université d'Orléans. 1991.

[Dershowitz-Sivakumar-87] N. Dershowitz and G. Sivakumar. Solving goals in equational langages. Proceedings of the first workshop CTRS. Springer-Verlag, vol 308. 1987.

[Dershowitz-Jouannaud-90] N. Dershowitz and J.P. Jouannaud. Rewrite system. Handbook of Theoretical Computer Science, Vol B, North-Holland, 1990.

[Fay-78] M. Fay. First-Order Unification in an Equational Theory. Master Thesis 78-5-002. University of Santa Cruz. 1978.

[Goguen-meseguer-86] J.A. Goguen and J. Meseguer. EQLOG : Equality, Types and generic Modules for logic Programming. In logic programming: functions, relations and equations. D. Degroot and G. Lindstrom, eds. Prentice-Hall, Englewood Cliffs, NJ, pp 295-363, 1986.

[Herold-86] A. Herold. Narrowing Techniques applied to idempotent Unification. Internal SEKI Report SR-86-16. August 1986.

[Hullot-80] J.M. Hullot. Canonical forms and unification. Proceedings of the fifth Conference on Automated Deduction, Springer-Verlag, vol 87, July 1980.

[Kirchner-84] C. Kirchner. A new equational unification method : a generalization of Martelli-Montanari's algorithm. Proceedings of the 7th CADE. Springer-Verlag, vol 170. 1984.

[Martelli-etal-87] A. Martelli, C. Moiso, and G-F. Rossi. Lazy unification algorithms for canonical rewrite systems. Proceedings of CREAS, Austin, Texas. May 1987.

[Réty-87] P. Réty. Improving basic Narrowing Techniques. Proceedings of the second conference on Rewriting Techniques and Applications, Springer-Verlag, vol 256. May 1987.

[Sivakumar-Dershowitz-87] G. Sivakumar and N. Dershowitz. Goal directed equation solving. Technical report. University of Illinois, USA. 1987.