Electronic Notes in Theoretical Computer Science 71 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume71.html 20 pages

Outermost Ground Termination

Olivier Fissore [1], Isabelle Gnaedig [1], and Hélène Kirchner [1]

LORIA-INRIA & LORIA-CNRS,
54506 Vandoeuvre-lès-Nancy, BP 239 Cedex, France.

Abstract

We propose a semi-automatic inductive process to prove termination of outermost rewriting on ground term algebras. The method is based on an abstraction mechanism, schematizing normalisation of subterms, and on narrowing, schematizing reductions on ground terms. The induction ordering is not needed a priori, but is a solution of constraints set along the proof. Our method applies in particular to systems that are non-terminating for the standard strategy nor even for the lazy strategy.

Key words: Rewriting, termination, outermost, strategy, rule based languages, induction, narrowing, ordering constraints

1 Introducing the problem

Termination of rewriting is a crucial problem in automated deduction, for equational logic, as well as in programming, for rule based languages. The property is important in itself, but it is also required to decide of properties like confluence and sufficient completeness, or to allow proofs by consistency. Most of existing methods for proving termination of term rewriting systems (TRS in short) essentially tackle the universal termination problem: they work on free term algebras and prove termination for standard rewriting (rewriting without any strategy). Many are based on syntactic or semantic noetherian orderings containing the rewriting relation induced by the TRS [15,7,20,3,8]. Other methods consist of transforming the termination problem of a TRS $\mathcal{R}$ into the termination problem of another TRS $\mathcal{R}'$, provable with techniques of the previous category. Examples are semantic labelling [25], and the dependency pair method [2].

In this paper, like in [16], we address the termination problem in the context of proof environments for rule-based programming languages, such as ASF+SDF [17], Maude [5], Cafe-OBJ [13], or ELAN [4], where a program is

[1] Email: {fissore,gnaedig,Helene.Kirchner}@loria.fr

a term rewriting system and the evaluation of a query consists of rewriting a ground expression. In such a context, one needs more specific termination proof tools than those previously cited, allowing one to prove termination under specific reduction strategies. There are still few results in this domain, although the need is important. To our knowledge, methods have only been given for the innermost strategy [1,14], the context-sensitive rewriting including particular kinds of local strategies [21], and general local strategies [10].

The outermost strategy for evaluating expressions in the context of programming is essentially used when one knows that computations can be non-terminating. The intuition suggests that rewriting a term at the highest possible position gives more chance than with another strategy to lead to an irreducible form. Indeed, outermost rewriting may succeed when innermost fails, as illustrated by the expression $second(dec(1), 0)$, with the rewrite rules $second(x, y) \rightarrow y$ and $dec(x) \rightarrow dec(x - 1)$ on integers. Innermost rewriting fails to terminate, because it first evaluates $dec(1)$ into $dec(0)$, $dec(-1)$, and so on. Outermost rewriting, however, gives 0 in one rewriting step. Moreover, outermost derivations are often shorter : in our example, to reduce $second(u, v)$, one does not need to reduce u, which can lead to infinite computations or, at least, to a useless evaluation. This advantage makes the outermost strategy an interesting strategy for rule-based languages, by allowing the interpreters to be more efficient, as well as for theorem proving, by allowing the rewriting-based proofs to be shorter.

Outermost computations are of interest in particular for functional languages like Miranda, Haskell, or Clean, where interpreters or compilers generally involve a strategy for call by name. Often, lazy evaluation is used instead: labelling operators in terms as lazy or eager, it consists in reducing the eager subterms only when their reduction allows a reduction step higher in the term [23].

However, lazy evaluation may diverge while the outermost computation terminates, which gives an additional motivation for studying outermost termination. For instance, let us consider the evaluation of the expression $inf(0)$ with the following two rules : $cons(x, cons(y, z)) \rightarrow big$, $inf(x) \rightarrow cons(x, inf(s(x)))$. Evaluated in a lazy manner, $inf(0)$ is reduced to $cons(0, inf(s(0)))$, and then, since application of the first rule fails, the sub-expression $inf(s(0))$ has to be evaluated before considering the whole expression, which leads to an infinite evaluation. Evaluated in an outermost manner, $inf(0)$ is also reduced to $cons(0, inf(s(0)))$, but then $inf(s(0))$ is reduced to $cons(s(0), inf(s(s(0))))$, and then the whole expression is reduced to big.

Although not generally used with the rule-based evaluation process, the outermost strategy could be of interest for languages like Maude, OBJ or ELAN, for the previous efficiency reasons. A better knowledge of this strategy w.r.t. more usual ones in this context, like the innermost or local strategies, would be interesting, and could help to choose the good one when programming in these languages. Studying in particular the termination problem for these different strategies, we have pointed out the fact that it is distinct for the innermost, the outermost and the lazy strategies : termination for one of these strategies does not imply termination for any other of them [12].

Lazy termination of functional languages has already been studied (see for example [24]), but to our knowledge, no termination proof tool exists for specifically proving outermost termination of rewriting. We propose here a proof method for checking termination of outermost computations, even if rewriting without strategy or with other strategies diverges.

Our termination proof method for outermost rewriting on ground terms is based on an explicit induction mechanism on the termination property. The main idea is to use induction on the termination property in order to prove that any element t of a given set of terms T terminates, i.e. there is no infinite derivation chain starting from t. So this approach needs a noetherian ordering on terms used in the induction principle. Unlike classical induction proofs, where the ordering is given, we do not need to define it *a priori*. We only have to check its existence by ensuring satisfiability of ordering constraints incrementally set along the termination proof. Thanks to the power of induction, the generated constraints are often simple to solve.

The method is based on an abstraction mechanism, schematizing normalization of subterms, and on narrowing, schematizing reductions on ground terms. It is semi-automatic : it can stop with success, then establishing outermost termination of a given TRS ; it can stop with failure or diverge, in which case nothing can be concluded about termination. Constraints solving can be handled by the procedure itself, or can be left to the user or delegated to external automatic constraint solvers.

In section 2, the background is presented. Section 3 introduces the basic concepts of our inductive proof mechanism. Section 4 builds on these concepts to propose a proof technique for outermost termination. Proofs can be found in [12].

2 The background

We assume that the reader is familiar with the basic definitions and notations of term rewriting given for instance in [9]. $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the set of terms built from a given finite set $\mathcal{F}$ of function symbols f having arity $n \in \mathbb{N}$, and a set $\mathcal{X}$ of variables denoted $x, y \ldots$. $\mathcal{T}(\mathcal{F})$ is the set of ground terms (without variables). The terms composed by a symbol of arity 0 are called constants. Positions in a term are represented as sequences of integers. The empty sequence ϵ denotes the top position. Let p and p' be two positions. The position p is said to be (a strict) prefix of p' (and p' suffix of p) iff $p' = p\lambda$, where λ is a non-empty sequence of integers. Given a term t, $Var(t)$ is the set of variables of t, $\mathcal{O}(t)$ is the set of positions of t, inductively defined as follows: $\mathcal{O}(t) = \{\epsilon\}\ if\ t \in \mathcal{X}$, $\mathcal{O}(t) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq n \text{ and } p \in \mathcal{O}(t_i)\}\ if\ t =$

$f(t_1, \dots, t_n)$. We denote $\mathcal{O}_{\mathcal{V}}(t)$ the set of variable positions in t. The notation $t|_p$ stands for the subterm of t at position p. If $p \in \mathcal{O}(t)$, then $t[t']_p$ denotes the term obtained from t by replacing the subterm at position p by the term t'.

A substitution is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F}, \mathcal{X})$, written $\sigma = (x \mapsto t) \dots (y \mapsto u)$. It uniquely extends to an endomorphism of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We identify a substitution $\sigma = (x \mapsto t) \dots (y \mapsto u)$ with the finite set of equations $(x = t) \wedge \dots \wedge (y = u)$. The result of applying σ to a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is written $\sigma(t)$ or σt. The domain of σ, denoted $Dom(\sigma)$ is the finite subset of $\mathcal{X}$ such that $\sigma x \neq x$. The range of σ, denoted $Ran(\sigma)$, is defined by $Ran(\sigma) = \bigcup_{x \in Dom(\sigma)} Var(\sigma x)$. We have in addition $Dom(\sigma) \cap Ran(\sigma) = \emptyset$. A ground substitution or instantiation is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F})$. The composition of substitutions σ_1 followed by σ_2 is denoted $\sigma_2\sigma_1$. Given two substitutions σ_1 and σ_2, we write $\sigma_1 \leq \sigma_2[X]$ iff $\exists \theta$ such that $\sigma_2 = \theta\sigma_1[X]$ i.e. $\forall x \in X, \sigma_2 x = \theta\sigma_1 x$ (σ_1 is said more general than σ_2). Given a subset $\mathcal{X}_1$ of $\mathcal{X}$, we write $\sigma_{\mathcal{X}_1}$ for the restriction of σ to the variables of $\mathcal{X}_1$, i.e. the substitution such that $Dom(\sigma_{\mathcal{X}_1}) \subseteq \mathcal{X}_1$ and $\forall x \in Dom(\sigma_{\mathcal{X}_1}) : \sigma_{\mathcal{X}_1} x = \sigma x$.

Given a set $\mathcal{R}$ of rewrite rules (a set of pairs of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, denoted $l \rightarrow r$, such that $Var(r) \subseteq Var(l)$) or term rewriting system on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, a function symbol in $\mathcal{F}$ is called a constructor iff it does not occur in $\mathcal{R}$ at the top position of the left-hand side of a rule, and is called a defined function symbol otherwise. The set of defined function symbols of $\mathcal{F}$ for $\mathcal{R}$ is denoted by $\mathcal{D}ef_{\mathcal{R}}$ ($\mathcal{R}$ is omitted when there is no ambiguity). The rewriting relation induced by $\mathcal{R}$ is denoted by $\rightarrow_{\mathcal{R}}$ ($\rightarrow$ if there is no ambiguity on $\mathcal{R}$), and defined by $s \rightarrow t$ iff there exists a substitution σ and a position p in s such that $s|_p = \sigma l$ for some rule $l \rightarrow r$ of $\mathcal{R}$, and $t = s[\sigma r]_p$. This is written $s \rightarrow_{p, l \rightarrow r, \sigma} t$ where either p or $l \rightarrow r$ or σ may be omitted; $s|_p$ is called a redex. The transitive (resp. reflexive transitive) closure of the rewriting relation induced by $\mathcal{R}$ is denoted by $\rightarrow^+_{\mathcal{R}}$ (resp. $\rightarrow^*_{\mathcal{R}}$). If $t \rightarrow^* t'$ and t' cannot be rewritten anymore, then t' is called a normal form of t and denoted by $t\downarrow$. Remark that given t, $t\downarrow$ may be not unique.

An ordering $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is said to be noetherian iff there is no infinite decreasing chain for this ordering. It is $\mathcal{F}$-stable iff for any pair of terms t, t' of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, for any context $f(\dots \; \dots)$, $t \succ t'$ implies $f(\dots t \dots) \succ f(\dots t' \dots)$. It has the subterm property iff for any t of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, $f(\dots t \dots) \succ t$. Observe that, for $\mathcal{F}$ and $\mathcal{X}$ finite, if $\succ$ is $\mathcal{F}$-stable and has the subterm property, then it is noetherian [19]. If, in addition, $\succ$ is stable by substitution (for any substitution σ, any pair of terms $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $t \succ t'$ implies $\sigma t \succ \sigma t'$), then it is called a simplification ordering. Let t be a term of $\mathcal{T}(\mathcal{F})$; let us recall that t terminates iff every rewriting derivation (or derivation chain) starting from t is finite.

We say that s outermost rewrites into t at position p, which is written $s \rightarrow^{out}_p t$ iff s rewrites into t at position p and there is no prefix position p' of p such that s rewrites at position p'. Let t be a term of $\mathcal{T}(\mathcal{F})$; like for the standard rewriting, we say that t outermost terminates iff every outermost rewriting derivation starting from t is finite. From now on, $t\downarrow$ denotes an outermost (rewriting) normal form of t.

Let us now recall the definition of narrowing. Let $\mathcal{R}$ be a TRS on $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A term t is *narrowed* into t', at the non-variable position p, using the rewrite rule $l \rightarrow r$ of $\mathcal{R}$ and the substitution σ, iff σ is a most general unifier of $t|_p$ and l, and $t' = \sigma(t[r]_p)$. This is denoted $t \leadsto_{R}^{p,l\rightarrow r,\sigma} t'$ where either p, or $l \rightarrow r$ or σ may be omitted. It is always assumed that there is no variable in common between the rule and the term, i.e. that $Var(l) \cap Var(t) = \emptyset$.

The requirement of disjoint variables is easily fulfilled by an appropriate renaming of variables in the rules when narrowing is performed. Observe that for the most general unifier σ used in the above definition, $Dom(\sigma) \subseteq Var(l) \cup Var(t)$ and we can choose $Ran(\sigma) \cap (Var(l) \cup Var(t)) = \emptyset$, thus introducing in the range of σ only fresh variables.

3 Induction for termination

The main intuition is to observe the outermost rewriting derivation tree starting from a ground term t which is an instance of a term $g(x_1, \ldots, x_m)$, for some defined function symbol $g \in \mathcal{D}ef$, and variables $x_1, \ldots, x_m$. Proving termination on ground terms amounts to prove that all these outermost rewriting derivation trees have only finite branches.

For proving that every term t of $\mathcal{T}(\mathcal{F})$ outermost terminates, we proceed by induction on $\mathcal{T}(\mathcal{F})$ with a noetherian ordering $\succ$, assuming that for any t' such that $t \succ t'$, t' outermost terminates. We first prove that a basic set of minimal elements for $\succ$ outermost terminates. As we will see, it is natural to suppose termination of subterms to prove termination of the terms themselves. So the subterm property for $\succ$ is required, and then the set of minimal elements for $\succ$ is a subset of the set of constants of $\mathcal{F}$. We then simulate the rewriting derivation trees starting from any instance of $g(x_1, \ldots, x_m)$, for all $g \in \mathcal{D}ef$, for proving that all branches are finite.

Each derivation tree is simulated by a proof tree starting from $g(x_1, \ldots, x_m)$, and developed by alternatively using two main concepts, namely narrowing and abstraction. More precisely, narrowing schematizes, thanks to all possible narrowing unifiers, all outermost rewriting possibilities of the ground terms in the derivations. The abstraction process simulates the normalization of subterms in the derivations, according to the outermost strategy. This abstraction step is performed on subterms that can be assumed outermost terminating by induction hypothesis.

So the proof process amounts to develop abstract trees whose nodes are composed of a current term that may have variables, and a set of ground substitutions appropriately represented by a constraint. This constraint results from the conjunction of the successive unifiers used for narrowing, from $g(x_1, \ldots, x_m)$ to the current term. Each node schematizes a possibly empty set of ground terms, namely all instances of the current term given by substitutions that are ground solutions of the constraint.

Obviously, the induction ordering $\succ$ has to be the same along the proof : for all branches in each abstract tree with a root $g(x_1, \ldots, x_m), g \in \mathcal{D}ef$, and for all abstract trees. It is however important to point out the flexibility of the proof method that allows the combination with auxiliary termination proofs using a different technique: when the induction hypothesis cannot be applied on a term u, it is sometimes possible to prove outermost termination of every ground instance of u by another way. In the following, we will use a predicate $TERMIN(u)$ that is true iff every ground instance of u outermost terminates.

The termination proof procedure given in this paper is described by a set of inference rules applied with a special strategy S. To prove termination of $\mathcal{R}$ on every term $t \in \mathcal{T}(\mathcal{F})$ we proceed as follows: for each defined symbol $g \in \mathcal{D}ef$, we consider a so-called reference term $t_{ref} = g(x_1, \ldots, x_m)$ and a trivial set of constraints denoted $\top$. Applying the rules according to the strategy S to the initial state $(\{g(x_1, \ldots, x_m)\}, \top)$ builds a proof tree, whose nodes are the states produced by the inference rules. Branching is obtained by the different narrowing steps for all possible rewrite rules. Different kinds of constraints occur: equality, disequality and reduction constraints during the process and ordering constraints at the end of the process.

As said before, there are three cases for the behavior of the termination proof procedure. The good case is when the strategy terminates because the rules do not apply anymore and all terminal states of all proof trees have an empty set of terms. The strategy can also stop with failure, when the rules do not apply anymore on states having non empty sets of terms. Finally, it may not terminate if there is an infinite number of applications of rules on one of the reference terms. In the last two cases, we cannot conclude anything about termination.

4 Outermost termination

Let us reconsider the previous ideas in a more detailed way. Consider the following example, that is outermost terminating, but not terminating for the standard rewriting relation, nor for the innermost strategy, nor even for the lazy evaluation strategy.

$$\begin{aligned} f(g(a)) &\rightarrow a \\ f(f(x)) &\rightarrow b \\ g(x) &\rightarrow f(g(x)) \end{aligned}$$

Let us first prove "by hand" that $\mathcal{R}$ is outermost terminating on $\mathcal{T}(\mathcal{F})$ ($\mathcal{F} = \{f : 1, g : 1, a : 0, b : 0\}$) in the following way. First, consider the constants a and b: they are obviously outermost terminating. Let us then study the termination of the ground terms of the form $f(t)$. If $t = g(a)$, $f(t)$ outermost rewrites into a, which is in normal form. Similarly, if t is of the form $f(t')$, $f(t)$ rewrites into b. Otherwise, $f(t)$ is not reducible at the top position. Using an ordering $\succ$ with the subterm property as induction ordering, we get $f(t) \succ t$. So, by induction hypothesis, we assume that t outermost terminates. Now, either t is irreducible, and then, since $f(t)$ is not reducible at the top position, $f(t)$ is in normal form, or t normalizes into $t\downarrow$. Let t' be any intermediate term of the outermost reduction chain from t to $t\downarrow$. Either $f(t')$ is outermost reducible at the top position into a or b, or the outermost redex is in t'. In the second case, t' is reduced into another intermediate term, on which we can apply the same reasoning. As t is supposed to be terminating, the number of intermediate terms is finite, which allows to conclude that $f(t)$ outermost terminates. We finally study the termination of the ground terms of the form $g(t)$. Any ground term of the form $g(t)$ first rewrites into $f(g(t))$. Then, if $t = a$, $f(g(t))$ normalizes into a. If $t \neq a$, $f(g(t))$ outermost rewrites into $f(f(g(t)))$ at position 1 with the third rule. Finally, the obtained term rewrites into b with the second rule, which ends the proof.

Let us show how to formalize and automate such a reasoning with narrowing and abstraction.

4.1 Induction

As said previously, we observe the outermost rewriting derivation trees starting from $t_{ref} = g(x_1, \ldots, x_m)$, for every $g \in \mathcal{F}$, where $x_1, \ldots, x_m$ are variables that can be instantiated by any ground term. The outermost rewriting relation is simulated by narrowing and abstracting as follows:

- First, we observe outermost rewriting from $g(x_1, \ldots, x_m)$, following the possible values of $x_1, \ldots, x_m$, either on top directly or on top after reduction of subterms. Indeed, any ground instance of $g(x_1, \ldots, x_m)$ may be reducible on top directly or after outermost reductions of its subterm instances of variables $x_1, \ldots, x_m$.

  So to precisely modelize the outermost rewriting relation on ground terms, we first have to replace $x_1, \ldots, x_m$ by new and all different variables $x'_1, \ldots, x'_m$ defined as follows. Given any ground instance $\alpha g(x_1, \ldots, x_m)$ of $g(x_1, \ldots, x_m)$, the $x'_1, \ldots, x'_m$ represent the first reduced form of $\alpha x_1, \ldots, \alpha x_m$ generating an outermost reduction higher in the term (here, at the top), in any outermost rewriting chain starting from $\alpha g(x_1, \ldots, x_m)$.

  We memorize this replacement and we apply a step of outermost narrowing to $g(x'_1, \ldots, x'_m)$ in all possible ways, to get terms v. This is the narrowing step.

- Then the idea is to apply the induction hypothesis as efficiently as possible on each resulting narrowed term v. For that we try to apply this induction hypothesis to the "smallest" (w.r.t. the size) subterms v_i of v, replacing them by a variable y_i, representing any of its normal forms. Indeed, when the v_i are smaller, there is more chance to satisfy the ordering constraints $t_{ref} > v_i$.

According to the outermost strategy, this can be done only if during their normalization, the v_i do not introduce outermost redexes higher in the term v. More formally, the induction hypothesis is applied to the subterms $v|_{p_1}, \ldots, v|_{p_n}$ of the current terms v, provided $\alpha t \succ \alpha v|_{p_1}, \ldots, \alpha v|_{p_n}$ for every ground substitution α, for the induction ordering $\succ$ and provided $s = v[y_1]_{p_1} \ldots [y_n]_{p_n}$ is not narrowable for the outermost narrowing relation, defined below. This implies that every ground instance of the term s outermost terminates. This is the abstracting step, which is a final step on the branches of the proof tree it applies on. If v cannot be abstracted, we attempt again a narrowing step.

- Our mechanism also includes the case where the induction hypothesis can directly be applied on the current term v, when $\alpha t_{ref} \succ \alpha v$ for every ground substitution α, which ends the proof on the branch of v, since every derivation starting from v in the derivation tree of t is supposed to terminate by induction. In fact, this case is a particular case of the abstracting step.

4.2 Outermost narrowing

We first formally define the variable replacement performed before a narrowing step.

Definition 4.1 Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ be a term whose variable occurrences from left to right in t are $x_1, \ldots, x_m$. The reduction renaming of t consists in replacing the x_i by new and all different variables x'_i in t, and is denoted by the so called reduction formula

$$R(t) = (x_1 \twoheadrightarrow^* x'_1, \ldots, x_m \twoheadrightarrow^* x'_m)[t].$$

The result of the reduction renaming applied to t is denoted $Ren(t)$.

Satisfaction of the reduction formulas is given in such a way that we simulate the outermost reduction relation by alternatively using reduction renaming and narrowing steps.

Definition 4.2 Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ be a term whose variable occurrences from left to right are $x_1, \ldots, x_m$, at positions $p_1, \ldots, p_m$ respectively. A ground substitution θ satisfies the reduction formula $R(t) = (x_1 \twoheadrightarrow^* x'_1, \ldots, x_m \twoheadrightarrow^* x'_m)[t]$ iff there exists an outermost rewriting chain starting from θt, such that either $t[\theta x'_1]_{p_1} \ldots [\theta x'_m]_{p_m}$ is the first reduced form of $\theta t = t[\theta x_1]_{p_1} \ldots [\theta x_m]_{p_m}$ on this chain having an outermost rewriting position at a non variable position of t, if this position exists, or $\theta x'_1 = (\theta x_1 \downarrow), \ldots, \theta x'_m = (\theta x_m \downarrow)$ if there is no such position.

Before going on, a few remarks on this definition can be made. In the second case of satisfiability, $t[\theta x_1 \downarrow]_1 \ldots [\theta x_m \downarrow]_m$ is in normal form. Moreover, $R(t)$ is always satisfiable : it is sufficient to take a ground substitution θ such that $t[\theta x_1]_{p_1} \ldots [\theta x_m]_{p_m}$ has an outermost rewriting position at a non variable position of t, and then to extend its domain $\{x_1, \ldots, x_m\}$ to $\{x_1, \ldots, x_m, x'_1, \ldots, x'_m\}$ by choosing for each $i \in \{1, ..., m\}$, $\theta x'_i = \theta x_i$. If such a substitution does not exist, then every ground instance of t has no outermost rewriting position at a non variable position of t, and it is sufficient to take a ground substitution θ such that $\theta x_1 = \ldots = \theta x_m = \theta x'_1 = \ldots = \theta x'_m = u$, with u any ground term in normal form.

However, there may exist several substitutions solution of such constraints. Let us consider for instance the rewrite system $R = \{f(a) \to f(c), b \to a\}$ and the reduction formula $R(f(x)) = (x \twoheadrightarrow^* x')[f(x)]$. The substitution $\theta_1(x) = \theta_1(x') = a$ and $\theta_2(x) = b, \theta_2(x') = a$ are two distinct solutions. With the substitution θ_2, $f(a)$ is the first reduced form of $f(b)$ having an outermost rewriting position at a non variable position of $f(x)$ (here at top).

Finally, quite often, the reduction formula is reducible to a simple renaming. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ be a term whose variables are $x_1, \ldots, x_m$, at positions $p_1, \ldots, p_m$ respectively. Let us consider $R(t) = (x_1 \twoheadrightarrow^* x'_1, \ldots, x_m \twoheadrightarrow^* x'_m)[t]$ and let us denote $I_t = \{i \in [1..m] | t$ outermost rewrites at a prefix position p'_i of $p_i\}$. Then $R(t)$ is equivalent (in the sense that the same set of ground substitutions satisfies both formulas) to $R(t) \bigwedge_{i \in I_t} x'_i = x_i$. Indeed, for any θ satisfying $R(t)$, for any $i \in I_t$, $t[\theta x_1]_{p_1} \ldots [\theta x_i]_{p_i} \ldots [\theta x_m]_{p_m}$ has at least one outermost rewriting position at a non variable position of t : the position p'_i or a prefix position of p'_i. Then we have $\theta x'_i = \theta x_i$.

To illustrate this, let us consider the system $\{g(x) \to x, f(x,x) \to x\}$ (the right-hand sides of the rules are not important here), and the reduction formula $R(f(x, g(y))) = (x \twoheadrightarrow^* x', y \twoheadrightarrow^* y')[f(x, g(y))]$. Then, since $f(x, g(y))$ outermost rewrites at the position of g, $R(f(x, g(y)))$ is equivalent to $R'(f(x, g(y))) = (x \twoheadrightarrow^* x')[f(x, g(y))] \wedge y = y'$. Indeed, whatever the ground instance θy, the term $g(\theta y)$ outermost rewrites only at the top position, and there is no outermost reduction of θy. Then, following Definition 4.2, $\theta y' = \theta y$. Finally, note that if $I_t = \{1, \ldots, m\}$, then $R(t)$ is equivalent to the renaming $\bigwedge_{i=1}^m x_i = x'_i$, where the x_i are the variable occurrences from left to right in t.

To schematize outermost rewriting on ground terms, we need to introduce a new specific narrowing relation.

Definition 4.3 A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ *outermost narrows* into a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ at the non-variable position p, using the rule $l \to r \in \mathcal{R}$ with the most general unifier σ, which is written $t \rightsquigarrow^{out}_{p, l \to r, \sigma} t'$ iff

(i) $\sigma(l) = \sigma(t|_p)$ and

(ii) $t' = \sigma(t[r]_p)$ and

(iii) there exist no prefix position p' of p, no rule $l' \to r'$ of $\mathcal{R}$ and no substitution σ' such that σ' can unify $\sigma t|_{p'}$ and l'.

It is always assumed that there is no variable in common between the rule and the term, i.e. that $Var(l) \cap Var(t) = \emptyset$.

The third point of the above definition does not appear in the classical definition of narrowing. This special condition is needed here to define an outermost narrowing mechanism modeling the outermost rewriting relation on ground terms.

The outermost narrowing steps applying to a given term t are computed in the following way. We look at every position p of t such that $t|_p$ unifies with the left-hand side of a rule using a substitution σ. The position p is an outermost narrowing position of t, according to Definition 4.3, iff there is no prefix position p' of t such that $\sigma t|_{p'}$ unifies with a left-hand side of rule. Then we look for every prefix position p' of p in t such that $\sigma t|_{p'}$ narrows with some substitutions σ' and some rule $l' \to r'$, and we set a constraint to exclude these substitutions.

Let us consider the previous system $\{f(g(a)) \to a, f(f(x)) \to b, g(x) \to f(g(x))\}$. With the standard narrowing relation used at the outermost position, $f(g(x_1))$ only narrows into a with the first rule and the substitution $\sigma = (x_1 = a)$. With the outermost narrowing relation defined above, $f(g(x_1))$ narrows into a with the first rule and $\sigma = (x_1 = a)$, and into $f(f(g(x_2)))$ with the third rule and the substitution $\sigma = (x_1 = x_2 \wedge x' = x_2)$ satisfying the disequality $x_2 \neq a$. Observe that x' comes from the renaming of variables in the rule.

As illustrated by this example, the substitutions used to narrow a term according to Definition 4.3 have in general to satisfy a set of disequalities. To make this remark precise, we need a few notations and definitions.

Let σ be a substitution on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ identified with the formula $\bigwedge_i (x_i = t_i)$, with $x_i \in \mathcal{X}$, $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and where $=$ is the syntactic equality. We denote by $\overline{\sigma}$ the formula $\bigvee_i (x_i \neq t_i)$.

Definition 4.4 A substitution σ is said *to satisfy* a formula $\bigwedge_i \bigvee_{j_i} (x_{j_i} \neq t_{j_i})$, iff for all ground instantiation α, $\bigwedge_i \bigvee_{j_i} (\alpha\sigma x_{j_i} \neq \alpha\sigma t_{j_i})$.

During the proof process, we memorize the previously defined reduction renamings of variables and the substitutions satisfying disequalities (also called *constrained substitutions*) used in the successive narrowing steps in a substitution constraint formula.

Definition 4.5 A *substitution constraint formula* (SCF for short) is a formula $\bigwedge_l (x_{l_1} \twoheadrightarrow^* x'_{l_1}, \ldots, x_{l_m} \twoheadrightarrow^* x'_{l_m})[u_l] \bigwedge_i (x_i = t_i) \bigwedge_j (\bigvee_{k_j} x_{k_j} \neq t_{k_j})$, with $x_{l_1}, x'_{l_1} \ldots, x_{l_m}, x'_{l_m}, x_i, x_{k_j} \in \mathcal{X}, t_i, t_{k_j} \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. The empty formula is denoted $\top$.

Then the nodes of the proof trees are composed of a current term, and a set of ground substitutions represented by a substitution constraint formula. Each node schematizes a possibly empty set of ground terms, namely all instances of the current term given by substitutions that are ground solutions of the substitution constraint formula.

Definition 4.6 A substitution constraint formula $\bigwedge_l(x_{l_1} \twoheadrightarrow^* x'_{l_1}, \dots, x_{l_m} \twoheadrightarrow^* x'_{l_m})[u_l] \bigwedge_i(x_i = t_i) \bigwedge_j(\bigvee_{k_j} x_{k_j} \neq t_{k_j})$ is said to be *satisfiable* iff there exists at least one instantiation θ such that $\bigwedge_i(\theta x_i = \theta t_i) \bigwedge_j(\bigvee_{k_j} \theta x_{k_j} \neq \theta t_{k_j})$ and θ satisfies $\bigwedge_l(x_{l_1} \twoheadrightarrow^* x'_{l_1}, \dots, x_{l_m} \twoheadrightarrow^* x'_{l_m})[u_l]$.

In practice, one can solve the equality and disequality part of the constraint and then check whether the solution θ satisfies the reduction formulas. This is trivial in cases where θ only instantiates the x'_i, since it can be extended by setting $\theta(x_i) = \theta(x'_i)$. Unfortunately, when θ also instantiates the x_i, we get an undecidable problem: given two ground terms t and t', can t be transformed into t' by repeated application of a given set of rewriting rules?

But fortunately, our process is sound even when inference rules are applied on nodes of the proof tree representing empty sets of terms [12]. So in the narrowing step, we do not attempt to check satisfiability of the constraints.

Nevertheless, we keep trace of the cumulated constraints along a branch of narrowing steps. If at any point, the constraints can be shown unsatisfiable, then the branch can be cut.

For the abstraction step, the satisfiability of an SCF has to be combined with the satisfiability of ordering constraints.

Definition 4.7 Let $t, u_1, \dots, u_m \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and Σ an SCF. The constraint $(\Sigma, t > u_1, \dots, u_m)$ is said to be satisfiable iff there exists an $\mathcal{F}$-stable ordering $\succ$ on $\mathcal{T}(\mathcal{F})$ having the subterm property such that $\theta t \succ \theta u_i, i \in [1..m]$, for every ground substitution θ satisfying Σ, and whose domain contains the variables of t and of the $u_i, i \in [1..m]$. Such an ordering $\succ$ is said to satisfy $(\Sigma, t > u_1, \dots, u_m)$.

The satisfiability of $(\Sigma, t > u_1, \dots, u_m)$ is undecidable. But a sufficient condition for an ordering $\succ$ to satisfy this constraint is that $\succ$ is stable by substitution (the induction ordering is then a simplification ordering) and $t \succ u_1, \dots, u_m$. Remark that consequently, $(\Sigma, t > u_1, \dots, u_m)$ may be proved satisfiable, even if Σ is not.

As said before, we store in an SCF Σ the reduction renamings of variables performed before the narrowing steps and the constrained substitutions used at each narrowing step, for unifying the current term u with a left-hand side of rule l, and whose domain is $Var(u) \cup Var(l)$. However, since $Var(u) \cap Var(l) = \emptyset$, we do not need to know the information related to the variables occurring in l because we are only interested by the transformations applied on u. So only the restriction of σ to $Var(u)$ and the negation of the restriction of each σ' (see Definition 4.3) to $Var(\sigma u)$, respectively denoted $\sigma_{Var(u)}$ and $\overline{\sigma'_{Var(\sigma u)}}$, are stored in Σ.

4.3 Abstraction

Let us now formalize our abstracting mechanism. The abstraction of a term u relies on the concept of generalization.

Definition 4.8 A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is a *generalization* of $u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ iff u is an instance of t (i.e. $\exists \sigma$ such that $\sigma t = u$). Given two generalizations s and t of a term u, t is said to be greater than s iff s is a generalization of t. The generalization t is called *linear generalization* if t is linear.

Then, abstracting a term u consists of finding the greatest linear generalization t with variables at positions $p_1, \dots, p_m$ such that

- the induction hypothesis applies to the $u|_{p_i}$, i.e. $(\Sigma, t_{ref} > u|_{p_1}, \dots, u|_{p_m})$ is satisfiable,
- the generalization $t = u[y_1]_{p_1} \dots [y_m]_{p_m}$, where $y_1, \dots, y_m$ are fresh distinct variables, is not narrowable.

The following obvious lemma allows us to conclude on termination of any ground instance of u, if u can be abstracted into a term t.

Lemma 4.9 *Let $u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Let t be a generalization of u, and $\{p_1, \dots, p_m\}$ be the set of variable positions in t. If t is not narrowable, then:*

(i) *in every ground instance of t,the only redex positions are suffixes of p_i, $i \in [1 \dots m]$,*

(ii) *consequently, if all ground instances of $u|_{p_1}, \dots, u|_{p_m}$ outermost terminate, then every ground instance of u outermost terminates.*

4.4 Inference rules

The inference rules for outermost termination, given in Table 1, work on pairs (T, Σ), where:

- T is a set of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, containing the current term whose ground instances have to be proved outermost terminating. This is either a singleton or the empty set.
- Σ is an SCF, enriched by the formulas expressing the reduction renaming of the variables of the terms to be narrowed and a new constrained substitution each time a narrowing step is performed.

The inference rules work as follows:

- The narrowing step is expressed by a rule **Narrow** applying on $(\{u\}, \Sigma)$: the variables of u are renamed as specified in Definition 4.1. Then $Ren(u)$ is outermost narrowed in all possible ways in one step, with all possible rewrite rules of the rewrite system $\mathcal{R}$, into terms v. For any possible v, $(\{u\}, \Sigma)$ is replaced by $(\{v\}, \Sigma \wedge R(u) \wedge \sigma_{Var(Ren(u))} \bigwedge_i \overline{\sigma'_{i\, Var(\sigma Ren(u))}})$ where $\sigma_{Var(Ren(u))} \bigwedge_i \overline{\sigma'_{i\, Var(\sigma Ren(u))}}$ is the SCF defining the constrained substitution allowing outermost narrowing of $Ren(u)$ into v.
- The abstracting step is expressed by a rule **Abstract** applying on $(\{u\}, \Sigma)$: we look for the greatest possible generalization t of the current term u, such as required in Section 4.3. If $(\Sigma, t_{ref} > u|_{p_1}, \dots, u|_{p_k})$ is satisfiable for $\{p_1, \dots, p_k\} \subseteq \mathcal{O}_{\mathcal{V}}(t)$ and $TERMIN(u|_i)$ for $i \in \mathcal{O}_{\mathcal{V}}(t) \setminus \{p_1, \dots, p_k\}$

Table 1
Inference rules for outermost t_{ref}-termination

Abstract : $\dfrac{\{u\},\ \ \Sigma}{\emptyset,\ \ \Sigma}$

if there is a greatest linear generalization t of u such that
t is not outermost narrowable and
$(\Sigma, t_{ref} > u|_{p_1}, \ldots, u|_{p_k})$ is satisfiable for $\{p_1, \ldots, p_k\} \subseteq \mathcal{O}_{\mathcal{V}}(t)$ and
$TERMIN(u|_i)$ for $i \in \mathcal{O}_{\mathcal{V}}(t) \setminus \{p_1, \ldots, p_k\}$

Narrow : $\dfrac{\{u\},\ \ \Sigma}{\{v\},\ \ \Sigma \wedge R(u) \wedge \sigma_{Var(Ren(u))} \bigwedge\limits_i \overline{\sigma'_{i\,Var(\sigma Ren(u))}}}$

if $Ren(u) \leadsto_{\sigma}^{out} v$ **where** $\sigma_{Var(Ren(u))} \bigwedge_i \overline{\sigma'_{i\,Var(\sigma Ren(u))}}$ is the constrained substitution allowing outermost narrowing of $Ren(u)$ with σ.

then, by induction hypothesis in the first case, and by hypothesis in the second one, all ground instances of $u|_{p_1}, \ldots, u|_{p_k}$ and of the $u|_i$ terminate. By Lemma 4.9, every ground instance of u outermost terminates, which ends the proof on the current derivation chain. So $(\{u\}, \Sigma)$ is replaced by $(\emptyset, \Sigma)$.

- As said before, we also can test for the current term u, whether $(\Sigma, t_{ref} > u)$ is satisfiable or $TERMIN(u)$. Then, as previously, by induction hypothesis or by hypothesis, every ground instance of u outermost terminates, which also ends the proof on the current derivation chain. This is a particular case of the rule **Abstract**, where the generalization of u is a variable y.

For establishing that $TERMIN(u)$ is true, in some cases, the notion of usable rules, proposed in [1], can be used. Given a TRS $\mathcal{R}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and a term $u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, the usable rules are defined as a computable superset of the rewrite rules, that may apply to any ground instance of u, for the standard rewriting relation, until its ground normal form is reached, if it exists. Proving termination of any ground instance of u then comes down to proving termination of its usable rules, which is often much easier than orienting the whole TRS. The usable rules can be proved terminating with classical termination methods like simplification orderings - we then obtain termination of the standard rewriting relation, which implies outermost termination of u - or with our inductive method itself : we then directly prove outermost termination of u. An interesting point of this method is that the ordering used to prove termination of the usable rules, either with classical methods or inductively, is completely independent of the main induction ordering. Remark that the case where the initial TRS can be proved terminating with other well known methods (orienting rules by a simplification ordering or using dependency pairs) is taken into account with our method: the usable rules of the first term

$g(x_1, \ldots, x_m)$ of any proof tree consist of the initial whole TRS, and then the previously mentioned methods can be used to prove their termination.

Remark also that for proving termination of the constants, usable rules can also be used. The notion of usable rules, their computation and their properties are fully developed in [14].

According to the remark following Definition 4.2, the reduction formulas in Σ may be reduced to simple variable renamings. In this case, Σ only contains variable renamings and constrained substitutions, that can be used to show that the ordering constraint needed to apply **Abstract** is satisfiable (see Examples B.1 and B.4 in [12]). The following lemma can also be established. It enables to apply **Abstract** when the current term is either a variable, or a non narrowable term.

Lemma 4.10 *Let $(\{t_i\}, \Sigma_i)$ be the i^{th} state of any branch of the derivation tree obtained by applying the strategy S on $(\{t_{ref}\}, \top)$, and $\succ$ an $\mathcal{F}$-stable ordering having the subterm property. If every reduction formula in Σ_i can be reduced to a formula $\bigwedge_j x_j = x'_j$, then we have:*

$$\forall x \in Var(t_i) : (\Sigma_i, t_{ref} > x) \text{ is satisfiable by } \succ.$$

To prove outermost termination of $\mathcal{R}$ on every term $t \in \mathcal{T}(\mathcal{F})$, for each defined symbol $g \in \mathcal{D}ef$, we apply the rules on the initial set of terms $T = \{t_{ref} = g(x_1, \ldots, x_m)\}$ and the trivial constraint $\top$. These rules must be applied with the strategy S: try to apply **Abstract**. If **Abstract** does not apply, then apply **Narrow**. Then repeat the process until no rule applies anymore.

Let us emphasize some important points about the inference rules:

- **Narrow** is a non-deterministic rule that can produce several results: it is applied with all possible rewrite rules at all outermost positions.
- After **Abstract**, no rule applies anymore.

The three cases for the behavior of the termination proof procedure can now be described more precisely. The strategy applied to the initial state $(\{t_{ref}\}, \top)$ may stop with states having non empty sets of terms, it may not terminate if there is an infinite number of applications of **Narrow**. The good case is when the strategy terminates because the rules do not apply anymore and all states are of the form $(\emptyset, \Sigma)$.

Let us write $SUCCESS(g, \succ)$ iff the application of the inference rules on $(\{g(x_1, \ldots, x_m)\}, \top)$, whose conditions are satisfied by $\succ$, gives a state of the form $(\emptyset, \Sigma)$ at the end of all branches of the derivation tree. We then can state the main result.

Theorem 4.11 *Let $\mathcal{R}$ be a TRS on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, such that the constants of $\mathcal{F}$ are outermost terminating. If there exists an $\mathcal{F}$-stable ordering $\succ$ having the subterm property, such that for each non-constant defined symbol g, $SUCCESS(g, \succ)$, then every term of $\mathcal{T}(\mathcal{F})$ outermost terminates.*

We now develop examples, giving the states of the proof trees. Other examples can be found in [12].

Example 4.12 Consider the previous example $\mathcal{R} = \{f(g(a)) \rightarrow a, f(f(x)) \rightarrow b, g(x) \rightarrow f(g(x))\}$, that is outermost terminating, but not terminating for the standard rewriting relation. We prove that $\mathcal{R}$ is outermost terminating on $\mathcal{T}(\mathcal{F})$ where $\mathcal{F} = \{f : 1, g : 1, a : 0, b : 0\}$.

The defined symbols of $\mathcal{F}$ for $\mathcal{R}$ are f and g. Applying the rules on $f(x_1)$, we get:

$$\begin{tikzcd}
& \begin{array}{c} f(x_1) \\ \Sigma = \top \end{array} \arrow[dl, "Narrow"'] \arrow[dr, "Narrow"] & \\
\begin{array}{c} a \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[f(x_1)] \\ \wedge\, x_1' = g(a)) \\ (\sigma = (x_1' = g(a))) \end{array} \arrow[d, "Abstract"] & & \begin{array}{c} b \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[f(x_1)] \\ \wedge\, x_1' = f(x_2)) \\ (\sigma = (x_1' = f(x_2) \wedge x' = x_2)) \end{array} \arrow[d, "Abstract"] \\
\begin{array}{c} \emptyset \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[f(x_1)] \\ \wedge\, x_1' = g(a)) \end{array} & & \begin{array}{c} \emptyset \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[f(x_1)] \\ \wedge\, x_1' = f(x_2)) \end{array}
\end{tikzcd}$$

The variable x' comes from the renaming of x in the left-hand side of rule. For the first **Abstract**, a is generalized into x_3, since $(\Sigma, f(x_1) > a)$ is satisfiable by a Lexicographic Path Ordering (LPO in short) $\succ$ with the precedence (ordering on $\mathcal{F}$) $f \succ_{\mathcal{F}} a$. For the second **Abstract**, b is generalized into x_4, since $(\Sigma, f(x_1) > b)$ is satisfiable by the previous LPO with the additional precedence $f \succ_{\mathcal{F}} b$. We recall that the induction ordering has to be the same for all the branches of all the derivation trees.

Applying the rules on $g(x_1)$, we get:

$$\begin{tikzcd}
& \begin{array}{c} g(x_1) \\ \Sigma = \top \end{array} \arrow[d, "\mathit{Narrow}"] & \\
& \begin{array}{c} f(g(x'_1)) \\ \Sigma = ((x_1 \twoheadrightarrow^* x'_1)[g(x_1)]) \\ (\sigma = (x' = x'_1)) \end{array} \arrow[dl, "\mathit{Narrow}"'] \arrow[dr, "\mathit{Narrow}"] & \\
\begin{array}{c} a \\ \Sigma = ((x_1 \twoheadrightarrow^* x'_1)[g(x_1)] \\ \wedge (x'_1 \twoheadrightarrow^* x''_1)[f(g(x'_1))] \\ \wedge x''_1 = a) \\ (\sigma = (x''_1 = a)) \end{array} \arrow[d, "\mathit{Abstract}"] & & \begin{array}{c} f(f(g(x''_1))) \\ \Sigma = ((x_1 \twoheadrightarrow^* x'_1)[g(x_1)] \\ \wedge (x'_1 \twoheadrightarrow^* x''_1)[f(g(x'_1))] \\ \wedge x''_1 \neq a) \\ (\sigma = (x'' = x''_1) \\ \text{satisfies } (x''_1 \neq a)) \end{array} \arrow[d, "\mathit{Narrow}"] \\
\begin{array}{c} \emptyset \\ \Sigma = ((x_1 \twoheadrightarrow^* x'_1)[g(x_1)] \\ \wedge (x'_1 \twoheadrightarrow^* x''_1)[f(g(x'_1))] \\ \wedge x''_1 = a) \end{array} & & \begin{array}{c} b \\ \Sigma = ((x_1 \twoheadrightarrow^* x'_1)[g(x_1)] \\ \wedge (x'_1 \twoheadrightarrow^* x''_1)[f(g(x'_1))] \\ \wedge (x''_1 \twoheadrightarrow^* x'''_1)[f(f(g(x''_1)))] \\ \wedge x''_1 \neq a) \\ (\sigma = (x''' = g(x'''_1))) \end{array} \arrow[d, "\mathit{Abstract}"] \\
& & \begin{array}{c} \emptyset \\ \Sigma = ((x_1 \twoheadrightarrow^* x'_1)[g(x_1)] \\ \wedge (x'_1 \twoheadrightarrow^* x''_1)[f(g(x'_1))] \\ \wedge (x''_1 \twoheadrightarrow^* x'''_1)[f(f(g(x''_1)))] \\ \wedge x''_1 \neq a) \end{array}
\end{tikzcd}$$

Abstract is applied on a and b with the previous LPO with the extra precedence $g \succ_{\mathcal{F}} a, b$. When narrowing $f(g(x'_1))$, we first try the top position, and find a possible unification with the first rule (the left branch). One also must consider the third rule if x''_1 is such that $x''_1 \neq a$; thus, if $x''_1 \neq a$, $f(g(x''_1))$ is narrowed at position 1 with the third rule (second branch).

Example 4.13 Let $\mathcal{R}$ be the TRS cited in the introduction, built on $\mathcal{F} = \{cons : 2, inf : 1, big : 0\}$:

$$\begin{aligned} cons(x, cons(y, z)) &\rightarrow big \\ inf(x) &\rightarrow cons(x, inf(s(x))) \end{aligned}$$

Applying the inference rules on $inf(x_1)$, we get :

$$\begin{gathered} inf(x_1) \\ \Sigma = \top \end{gathered}$$

$$\Big\downarrow {\scriptstyle Narrow}$$

$$\begin{gathered} cons(x_1', inf(s(x_1'))) \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[inf(x_1)]) \\ (\sigma = (x' = x_1')) \end{gathered}$$

$$\Big\downarrow {\scriptstyle Narrow}$$

$$\begin{gathered} cons(x_1'', cons(s(x_1'''), inf(s(s(x_1'''))))) \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[inf(x_1)] \\ \wedge\ (x_1' \twoheadrightarrow^* x_1'', x_1' \twoheadrightarrow^* x_1''')[cons(x_1', inf(s(x_1')))]) \\ (\sigma = (x'' = s(x_1'''))) \end{gathered}$$

$$\Big\downarrow {\scriptstyle Narrow}$$

$$\begin{gathered} big \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[inf(x_1)] \\ \wedge\ (x_1' \twoheadrightarrow^* x_1'', x_1' \twoheadrightarrow^* x_1''')[cons(x_1', inf(s(x_1')))] \\ \wedge\ (x_1'' \twoheadrightarrow^* x_1^{(iv)}, x_1''' \twoheadrightarrow^* x_1^{(v)}, x_1''' \twoheadrightarrow^* x_1^{(vi)}) \\ [cons(x_1'', cons(s(x_1'''), inf(s(s(x_1'''))))))]) \\ (\sigma = (x''' = x_1^{(iv)} \wedge y' = s(x_1^{(v)}) \wedge z' = inf(s(s(x_1^{(vi)}))))) \end{gathered}$$

$$\Big\downarrow {\scriptstyle Abstract}$$

$$\begin{gathered} \emptyset \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1')[inf(x_1)] \\ \wedge\ (x_1' \twoheadrightarrow^* x_1'', x_1' \twoheadrightarrow^* x_1''')[cons(x_1', inf(s(x_1')))] \\ \wedge\ (x_1'' \twoheadrightarrow^* x_1^{(iv)}, x_1''' \twoheadrightarrow^* x_1^{(v)}, x_1''' \twoheadrightarrow^* x_1^{(vi)}) \\ [cons(x_1'', cons(s(x_1'''), inf(s(s(x_1'''))))))]) \end{gathered}$$

Applying the inference rules on $cons(x_1, x_2)$, we get :

$$\begin{gathered} cons(x_1, x_2) \\ \Sigma = \top \end{gathered}$$

$$\Big\downarrow {\scriptstyle Narrow}$$

$$\begin{gathered} big \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1', x_2 \twoheadrightarrow^* x_2')[cons(x_1, x_2)] \\ \wedge\ x_2' = cons(x_3, x_4)) \\ (\sigma = (x_2' = cons(x_3, x_4) \wedge\ x' = x_1' \wedge\ y' = x_3 \wedge\ z' = x_4)) \end{gathered}$$

$$\Big\downarrow {\scriptstyle Abstract}$$

$$\begin{gathered} \emptyset \\ \Sigma = ((x_1 \twoheadrightarrow^* x_1', x_2 \twoheadrightarrow^* x_2')[cons(x_1, x_2)] \\ \wedge\ x_2' = cons(x_3, x_4)) \end{gathered}$$

5 Conclusion and perspectives

In this paper, we have proposed a method to prove outermost termination of term rewriting systems by explicit induction on the termination property. Our method works on the ground term algebra using as induction relation an $\mathcal{F}$-stable ordering having the subterm property. The general proof principle relies on the simple idea that for establishing termination of a ground term t, it is enough to suppose that subterms of t are smaller than t for this ordering, and that rewriting the context leads to terminating chains. Iterating this process until obtaining a context which is not reducible anymore establishes the termination of t.

An important point to automate our proof principle is the satisfaction of the ordering constraints. On many examples, this is immediate: since the ordering constraints only express the subterm property, they are trivially satisfied by any simplification ordering, so we do not need any ordering constraint solver. Elsewhere, we can use an ordering constraint solver as C*i*ME [6].

Up to our knowledge, our algorithm is the first automated method for proving outermost termination of non-terminating systems, and whose generality allows covering other reduction strategies.

Indeed, our process, based on narrowing and abstraction, can also be extended to other strategies. We recently have proposed inference rules for the innermost strategy [14], and for local strategies on operators [10], as they are used in OBJ3, CafeOBJ or Maude. Note that the leftmost innermost strategy can be expressed by local strategies, and, as shown in [18], the termination problem of leftmost innermost rewriting is equivalent to the termination problem of innermost rewriting. So our termination procedure for local strategies holds for proving innermost termination. On the contrary, the outermost strategy cannot be expressed by local strategies. Indeed, strategies like innermost, lazy or local strategies are intrinsically recursive in the sense that, during the evaluation, subterms have to be completely evaluated (i.e. normalized) before the whole term is reduced. This is not the case for the outermost evaluation. The termination procedure proposed in this paper is thus specific to the outermost case. With respect to innermost or local strategies, the order in which narrowing and abstraction apply is changed. Moreover, the narrowing process itself had to be adapted to suitably schematize outermost rewriting on ground terms. This led us to define and deal with specific appropriate constraints, different from those of the innermost and local strategy cases.

Since our induction principle is based on the rewriting relation itself, the extension to the associative-commutative case, as well as to the typed case seems to be easy.

We recently have implemented the algorithm proposed in this paper. This implementation is integrated in CARIBOO [11], a tool recently proposed for proving termination of rewriting under strategies, in which our inductive procedures for the innermost and local strategies are also implemented.

References

[1] T. Arts and J. Giesl. Proving innermost normalization automatically. Technical Report 96/39, Technische Hochschule Darmstadt, Germany, 1996.

[2] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[3] A. Ben Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9(2):137–160, October 1987.

[4] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and C. Ringeissen. An overview of ELAN. In C. Kirchner and H. Kirchner, editors, *Proceedings of the second International Workshop on Rewriting Logic and Applications*, volume 15, `http://www.elsevier.nl/locate/entcs/volume15.html`, Pont-à-Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science. Report LORIA 98-R-316.

[5] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, editor, *Proceedings of the 1st International Workshop on Rewriting Logic and its Applications*, volume 5 of *Electronic Notes in Theoretical Computer Science*, Asilomar, Pacific Grove, CA, USA, September 1996. North Holland.

[6] E. Contejean, C. Marché, B. Monate, and X. Urbain. CiME version 2. Prerelease available at `http://cime.lri.fr`.

[7] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[8] N. Dershowitz and C. Hoot. Natural termination. *Theoretical Computer Science*, 142(2):179–207, 1995.

[9] N. Dershowitz and J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990. Also as: Research report 478, LRI.

[10] O. Fissore, I. Gnaedig, and H. Kirchner. Termination of rewriting with local strategies. In M. P. Bonacina and B. Gramlich, editors, *Selected papers of the 4th International Workshop on Strategies in Automated Deduction*, volume 58 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001. Also available as Technical Report A01-R-177, LORIA, Nancy, France.

[11] O. Fissore, I. Gnaedig, and H. Kirchner. CARIBOO : An induction based proof tool for termination with strategies. In *Proceedings of the Fourth International Conference on Principles and Practice of Declarative Programming (PPDP)*, Pittsburgh, USA, October 2002. ACM Press.

[12] O. Fissore, I. Gnaedig, and H. Kirchner. Outermost ground termination - Extended version. Technical report A02-R-493, LORIA, Nancy, France, December 2002.

[13] K. Futatsugi and A. Nakagawa. An overview of CAFE specification environment – an algebraic approach for creating, verifying, and maintaining formal specifications over networks. In *Proceedings of the 1st IEEE Int. Conference on Formal Engineering Methods*, 1997.

[14] I. Gnaedig, H. Kirchner, and O. Fissore. Induction for innermost and outermost ground termination. Technical Report A01-R-178, LORIA, Nancy, France, 2001.

[15] S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path ordering. Unpublished manuscript, 1980.

[16] H. Kirchner and I. Gnaedig. Termination and normalisation under strategy - Proofs in ELAN. In *Proceedigs of the Third International Workshop on Rewriting Logic and its Applications*, volume 36 of *Elecronic Notes In Theoretical Computer Science*, Kanazawa, JAPAN, September 2000. Elsevier Science Publishers.

[17] P. Klint. A meta-environment for generating programming environments. *ACM Transactions on Software Engineering and Methodology*, 2:176–201, 1993.

[18] M. R. K. Krishna Rao. Some characteristics of strong normalization. *Theoretical Computer Science*, 239:141–164, 2000.

[19] J. B. Kruskal. Well-quasi ordering, the tree theorem and Vazsonyi's conjecture. *Trans. Amer. Math. Soc.*, 95:210–225, 1960.

[20] D. S. Lankford. On proving term rewriting systems are noetherian. Technical report, Louisiana Tech. University, Mathematics Dept., Ruston LA, 1979.

[21] S. Lucas. Termination of rewriting with strategy annotations. In A. Voronkov and R. Nieuwenhuis, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 669–684, La Habana, Cuba, December 2001. Springer-Verlag, Berlin.

[22] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *Applicable algebra in Engineering, Communication and Computation*, 5(3&4):213–253, 1994.

[23] Q.-H. Nguyen. Compact normalisation trace via lazy rewriting. In S Lucas and B Gramlich, editors, *Proc. 1st International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2001)*, volume 57. Elsevier Science Publishers B. V. (North-Holland), May 2001. Available at `http://www.elsevier.com/locate/entcs/volume57.html`.

[24] S. E. Panitz and M. Schmidt-Schauss. TEA: Automatically proving termination of programs in a non-strict higher-order functional language,. In *Proceedings of Static Analysis Symposium'97*, volume 1302 of *Lecture Notes in Computer Science*, pages 345–360. Springer-Verlag, 1997.

[25] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.