HAL
open science

Termination of Rewriting under Strategies

Isabelle Gnaedig, Hélène Kirchner

To cite this version:

Isabelle Gnaedig, Hélène Kirchner. Termination of Rewriting under Strategies. ACM Transactions on Computational Logic, 2009, 10 (2), pp.1-52. 10.1145/1462179.1462182 . inria-00182432

HAL Id: inria-00182432

https://inria.hal.science/inria-00182432

Submitted on 14 Sep 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Termination of Rewriting under Strategies*

ISABELLE GNAEDIG
and
HÉLÈNE KIRCHNER
INRIA & LORIA (UMR 7503 CNRS-INPL-INRIA-Nancy 2-UHP)

A termination proof method for rewriting under strategies, based on an explicit induction on the termination property, is presented and instantiated for the innermost, outermost and local strategies. Rewriting trees are simulated by proof trees generated with an abstraction mechanism, narrowing and constraints representing sets of ground terms. Abstraction introduce variables to represent normal forms without computing them and to control the narrowing mechanism, well-known to easily diverge. The induction ordering is not given a priori, but defined with ordering constraints, incrementally set during the proof. It is established that termination under strategy is equivalent to the construction of finite proof trees schematizing terminating rewriting trees. Sufficient effective conditions to ensure finiteness are studied and the method is illustrated on several examples for each specific strategy.

Categories and Subject Descriptors: F.3.1 [**LOGICS AND MEANINGS OF PROGRAMS**]: Specifying and Verifying and Reasoning about Programs—*Logics of programs, Mechanical verification, Specification techniques*; F.4.2 [**MATHEMATICAL LOGIC AND FORMAL LANGUAGES**]: Grammars and Other Rewriting Systems; F.4.3 [**MATHEMATICAL LOGIC AND FORMAL LANGUAGES**]: Formal Languages—*Algebraic language theory*; I.1.3 [**SYMBOLIC AND ALGEBRAIC MANIPULATION**]: Languages and Systems—*Evaluation strategies, Substitution mechanisms*; I.2.2 [**ARTIFICIAL INTELLIGENCE**]: Automatic Programming—*Automatic analysis of algorithms, Program verification*; I.2.3 [**ARTIFICIAL INTELLIGENCE**]: Deduction and Theorem Proving—*Deduction, Inference engines, Mathematical induction*; D.3.1 [**PROGRAMMING LANGUAGES**]: Formal Definitions and Theory; D.2.4 [**SOFTWARE ENGINEERING**]: Software/Program Verification—*Correctness proofs, Formal methods, Validation*

General Terms: Algorithms, Languages, Verification

Additional Key Words and Phrases: abstraction, induction, innermost, local strategy, narrowing, ordering constraint, outermost, termination

1. INTRODUCING THE PROBLEM

Rewriting techniques are now widely used in automated deduction, especially to handle equality, as well as in programming, in functional, logical or rule-based languages. Termination of rewriting is a crucial property, important in itself to guarantee a result in a finite number of steps. But it is also required to decide properties like confluence and sufficient completeness, or to allow proofs by consistency.

Existing methods for proving termination of rewrite systems essentially tackle the termination problem on free term algebras for rewriting without strategies. Most are based on syntactic or semantic noetherian orderings containing the rewriting relation induced by the rewrite system [Plaisted 1978; Lankford 1979; Kamin and

* This paper is a preliminary version of a paper to appear in ACM Transactions of Computational Logic (accepted September 2007).

Lévy 1982; Dershowitz 1982a; Ben Cherifa and Lescanne 1987; Dershowitz and Hoot 1995; Borralleras et al. 2000].

Other methods consist in transforming the termination problem of a rewrite system into another decreasingness problem on which former techniques may apply. Examples are semantic labelling [Zantema 1995], and the dependency pair method [Arts and Giesl 2000; Giesl et al. 2004]. For most approaches, finding an appropriate ordering is the key problem, that often comes down to solving a set of ordering constraints.

In the context of proof environments for rule-based programming languages, such as ASF+SDF [Brand et al. 2001], Maude [Clavel et al. 2003], CafeOBJ [Futatsugi and Nakagawa 1997], Stratego [Visser 2001], ELAN [Borovanský et al. 1998], or TOM [Moreau et al. 2003], a program is a rewrite system and the evaluation of a query consists in rewriting a first-order term. However, in this context, specific termination proof tools are required, to allow termination proofs under specific reduction strategies. There are still few results in this domain. To our knowledge, methods have only been given with the innermost strategy [Arts and Giesl 1997; Giesl et al. 2003], and for the context-sensitive rewriting [Lucas 1996; Giesl and Middeldorp 1999; Lucas 2002; Giesl and Middeldorp 2003; Alarcón et al. 2006; Alarcón and Lucas 2007], which involves particular kinds of local strategies [Lucas 2001a; 2001b; Alpuente et al. 2004]. In previous works, we already have obtained termination results on ground terms for the innermost strategy [Fissore et al. 2002a], for general local strategies on the operators [Fissore et al. 2001], and for the outermost strategy [Fissore et al. 2002b].

Rewriting under strategies is a particular case of "strategic rewriting". This terminology has emerged in [Visser 2004] and [Kirchner 2005] to denote the capability to express control of rewriting via a strategy language. In [Fissore et al. 2003b], we have studied termination of strategic rewriting, by simplification of strategy expressions.

In this paper, we propose a general proof technique based on an explicit induction mechanism on the termination property. We then show how it can be instantiated to give a termination proof algorithm for the innermost strategy, the outermost strategy, and local strategies on operators. This is a generalization of our previous results that also allowed to considerably simplify the technical features of our approach, proposed in the three original algorithms.

The three considered strategies have been chosen for their relevance to programming languages.

The most widely used is the innermost strategy, consisting in rewriting always at the lowest possible positions. It is often used as a built-in mechanism in evaluation of rule-based or functional languages. In addition, for non-overlapping or locally confluent overlay systems [Gramlich 1995], or systems satisfying critical peak conditions [Gramlich 1996], innermost termination is equivalent to standard termination (i.e. termination for standard rewriting, which consists in rewriting without any strategy). As proved in [Krishna Rao 2000], termination of rewriting is equivalent for the leftmost innermost and the innermost strategies.

The outermost strategy for evaluating expressions in the context of programming is essentially used when one knows that some 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 rewriting fails, as illustrated by the expression $second(dec(1), 0)$, with the rewrite rules $second(x, y) \to y$ and $dec(x) \to dec(x-1)$ 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 may be 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 also of interest for functional languages, where interpreters or compilers generally involve a strategy for call by name. Often, lazy evaluation is used instead: operators are labelled in terms as lazy or eager, and the strategy consists in reducing the eager subterms only when their reduction allows a reduction step higher in the term [Nguyen 2001]. Lazy termination of functional languages has already been studied (see for example [Panitz and Schmidt-Schauss 1997] and [Giesl et al. 2006] for Haskell), but to our knowledge, except our previously cited work, no termination proof method exists for specifically proving outermost termination of rewriting.

Note finally that 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 $f(0)$ with the following two rules : $c(x, c(y, z)) \to b$, $f(x) \to c(x, f(s(x)))$. If f is labelled as eager, $f(0)$ is reduced to $c(0, f(s(0)))$, and then, since application of the first rule fails, the sub-expression $f(s(0))$ has to be evaluated before considering the whole expression, which leads to an infinite evaluation. Evaluated in an outermost manner, $f(0)$ is also reduced to $c(0, f(s(0)))$, but then $f(s(0))$ is reduced to $c(s(0), f(s(s(0))))$, and the whole expression is reduced to b.

Local strategies on operators are also used to force the evaluation of expressions to terminate. A well known example is the evaluation of an *if_then_else* expression, for which evaluating the first argument in priority may allow to avoid divergence. This kind of strategy is available in languages such that OBJ3, CafeOBJ or Maude, and has been studied in [Eker 1998] and [Nakamura and Ogata 2000]. It is defined in the following way: to any operator f is attached an ordered list of integers $LS(f)$, giving the positions of the subterms to be evaluated in a given term, whose top operator is f. For example, the rewrite system

$$\begin{array}{ll}
f(i(x)) & \to if_then_else(zero(x), g(x), f(h(x))) \\
zero(0) & \to true \\
zero(s(x)) & \to false \\
if_then_else(true, x, y) & \to x \\
if_then_else(false, x, y) & \to y \\
h(0) & \to i(0) \\
h(x) & \to s(i(x))
\end{array}$$

does not terminate for the standard rewriting relation, but does with the following strategy: $LS(ite) = [1; 0]$, $LS(f) = LS(zero) = LS(h) = [1; 0]$ and $LS(g) = LS(i) =$

[1], where *ite* denotes *if_then_else* for short.

Local strategies have to be compared with context-sensitive rewriting where rewriting is also allowed at some specified positions only in the terms. The former specify an ordering on these rewriting positions, so they are more specific than context-sensitive rewriting where a redex is chosen in a set of positions.

The termination problem for the three considered strategies is always different: in [Fissore et al. 2002c], examples are given to show that termination for one of these strategies does not imply termination for any other one.

Despite of these distinct behaviours, the termination proofs we propose rely on the same principle and a few common concepts, that are emphasized in this paper. Our approach is based on an explicit induction mechanism on the termination property. The main idea is to proceed by induction on the ground term algebra with a noetherian ordering $\succ$, assuming that for any t' such that $t \succ t'$, t' terminates, i.e. there is no infinite derivation chain starting from t'. 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 only leads to terminating chains. Iterating this process until a non-reducible context is obtained establishes termination of t.

Termination of terms has also been proposed in [Goubault-Larreck 2001], but for inductively proving well-foundedness of binary relations, among which path orderings.

In order to explain more precisely the basic idea of our approach, let us consider the classical example, due to Toyama, of a rewrite system that does not terminate, but terminates with the innermost strategy:

$$\begin{array}{ll} f(0,1,x) & \to f(x,x,x) \\ g(x,y) & \to x \\ g(x,y) & \to y \end{array}$$

Let us prove by induction on the set $\mathcal{T}(\mathcal{F})$ of ground terms built on $\mathcal{F} = \{0,1,f,g\}$ with a noetherian ordering $\succ$, that any term t innermost terminates (i.e. there is no infinite innermost rewriting chain starting from t). The terms of $\mathcal{T}(\mathcal{F})$ are 0, 1, or terms of the form $f(t_1,t_2,t_3)$, or $g(t_1,t_2)$, with $t_1,t_2,t_3 \in \mathcal{T}(\mathcal{F})$. The terms 0 and 1 are obviously terminating.

We now prove that $f(t_1,t_2,t_3)$ is innermost terminating. First, $f(t_1,t_2,t_3) \succ t_1,t_2,t_3$ for any term ordering with the subterm property (i.e. any term is greater than any of its subterms). Then, by induction hypothesis, assume that t_1,t_2 and t_3 innermost terminate. Let $t_1{\downarrow},t_2{\downarrow},t_3{\downarrow}$ be respectively any of their normal forms. The problem is then reduced to innermost termination of all $f(t_1{\downarrow},t_2{\downarrow},t_3{\downarrow})$. If $t_1{\downarrow} = 0$, $t_2{\downarrow} = 1$, then $f(0,1,t_3{\downarrow})$ only rewrites at the top position into $f(t_3{\downarrow},t_3{\downarrow},t_3{\downarrow})$, which is in normal form. Else $f(t_1{\downarrow},t_2{\downarrow},t_3{\downarrow})$ is already in normal form.

Let us finally prove that $g(t_1,t_2)$ is innermost terminating. First, $g(t_1,t_2) \succ t_1$, t_2. Then, by induction hypothesis, assume that t_1 and t_2 innermost terminate. Let $t_1{\downarrow},t_2{\downarrow}$ be respectively any of their normal forms. It is then sufficient to prove that $g(t_1{\downarrow},t_2{\downarrow})$ is innermost terminating. The term $g(t_1{\downarrow},t_2{\downarrow})$ rewrites either into $t_1{\downarrow}$ or into $t_2{\downarrow}$ at the top position, with both $t_1{\downarrow}$ and $t_2{\downarrow}$ in normal form. Note that for $\succ$ in this proof, any ordering having the subterm property is convenient. Our goal

is to provide a procedure implementing such a reasoning, and valid for the three previously presented strategies.

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.

An important difference with other termination proof methods is that in our approach, the studied system $\mathcal{R}$ and the induction ordering $\succ$ are in general not connected in the usual sense, where proving termination of $\mathcal{R}$ amounts at finding a reduction ordering such that $l \succ r$ for all rules $l \to r$ in $\mathcal{R}$. This original aspect allows us to prove termination in a finer way than usual and to take into account rewriting strategies. This is useful not only for systems that only terminate under strategies, but also when rewriting under strategy is used to restrict the derivation space. Moreover, for the induction ordering, simple orderings (such as the subterm ordering or a recursive path ordering) are often sufficient, whereas they cannot be used to directly orient the rules. This is the case for Toyama's example: here, a subterm ordering is obviously not sufficient to compare left- and right-hand sides of the rules, because the system is not terminating, but it is enough as an induction relation to establish innermost termination.

The paper is organized as follows: in Section 2, the background is presented. Section 3 introduces the inductive proof principle of our approach. Section 4 gives the basic concepts of our inductive proof mechanism based on abstraction and narrowing, and the involved constraints. Section 5 presents the termination proof procedure that is further applied to different rewriting strategies. In Section 6, the mechanism is instantiated for the case of innermost termination. In Section 7, the procedure is applied to outermost termination, and in Section 8, to the case of local strategies. Finally, Section 9 addresses implementation and related work.

2. THE BACKGROUND

We assume that the reader is familiar with the basic definitions and notations of term rewriting given for instance in [Dershowitz and Jouannaud 1990; Baader and Nipkow 1998; Dershowitz and Plaisted 2001; Barendsen et al. 2003]. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is the set of terms built from a given finite set $\mathcal{F}$ of function symbols f having arity $n \in \mathbb{N}$ (denoted $f : n$), 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 reduced to 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. The symbol at the top position of a term t is written $top(t)$. Let p and p' be two positions. The position p is said to be (a strict) prefix of p' (and p' suffix of p) if $p' = p\lambda$, where λ is a (non-empty) sequence of integers. Given a term t, $Var(t)$ is the set of variables of t, $\mathcal{P}os(t)$ is the set of positions in t, inductively defined as follows: $\mathcal{P}os(t) = \{\epsilon\}$ if $t \in \mathcal{X}$, $\mathcal{P}os(t) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq n \text{ and } p \in \mathcal{P}os(t_i)\}$ if $t = f(t_1, \ldots, t_n)$. The set of non-variable positions is $\mathcal{P}os_{\mathcal{F}}(t) = \{p \in \mathcal{P}os(t) \mid t|_p \notin \mathcal{X}\}$, where the notation $t|_p$ stands for the subterm of t at position p. If $p \in \mathcal{P}os(t)$, 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 $\sigma = (x \mapsto t) \ldots (y \mapsto u)$. It uniquely extends to an endomorphism of $\mathcal{T}(\mathcal{F}, \mathcal{X})$. 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 $Ran(\sigma) = \bigcup_{x \in Dom(\sigma)} Var(\sigma x)$. An instantiation or ground substitution is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F})$. Id denotes the identity substitution. The composition of substitutions σ_1 followed by σ_2 is denoted $\sigma_2\sigma_1$. 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 $\forall x \in Dom(\sigma_{\mathcal{X}_1}) : \sigma_{\mathcal{X}_1} x = \sigma x$.

A reduction relation on terms is a relation $\rightarrow$ on $\mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X})$. A derivation is a sequence of terms $t_1 \rightarrow t_2 \rightarrow \ldots t_n$. A term t is reducible (for the relation $\rightarrow$) when there exists t' such that $t \rightarrow t'$ and irreducible otherwise. The reflexive transitive closure of the reduction relation is denoted by $\stackrel{*}{\rightarrow}$. If $t \stackrel{*}{\rightarrow} t'$ with t' irreducible, then t' is called a normal form of t and denoted by $t{\downarrow}$. Note that given t, $t{\downarrow}$ may be not unique.

In this work, we will use two families of reduction relations on terms, namely rewriting and narrowing relations.

A set $\mathcal{R}$ of rewrite rules or rewrite system on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is a set of pairs of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, denoted $l \rightarrow r$, such that $Var(r) \subseteq Var(l)$, and $l \notin \mathcal{X}$. Given a rewrite system $\mathcal{R}$, a function symbol in $\mathcal{F}$ is called a *constructor* iff it does not occur in $\mathcal{R}$ at the top position of a left-hand side of a rule, and is called a *defined function symbol* otherwise. The set of constructors of $\mathcal{F}$ for $\mathcal{R}$ is denoted $\mathcal{C}$, and the set of defined function symbols $\mathcal{D}$. In this paper, we only consider finite sets of rewrite rules.

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 $s \rightarrow_{\mathcal{R}}^{p, l \rightarrow r, \sigma} t$ where p, $l \rightarrow r$, σ or $\mathcal{R}$ may be omitted; $s|_p$ is called a redex.

Let $\mathcal{R}$ be a rewrite system 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, when σ is a most general unifier of $t|_p$ and l, and $t' = \sigma(t[r]_p)$. This is denoted $t \rightsquigarrow_{\mathcal{R}}^{p, l \rightarrow r, \sigma} t'$ where p, $l \rightarrow r$, σ or $\mathcal{R}$ may be omitted. It is always assumed that there is no variable in common between the rule and the term, i.e. that $Var(l) \cap Var(t) = \emptyset$.

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 monotone iff for any pair of terms t, t' of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, for any context $f(\ldots\ \ldots)$, $t \succ t'$ implies $f(\ldots t \ldots) \succ f(\ldots t' \ldots)$. It has the subterm property iff for any t of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, $f(\ldots t \ldots) \succ t$. It is stable under substitution iff for any substitution σ, any pair of terms $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $t \succ t'$ implies $\sigma t \succ \sigma t'$. An ordering which is noetherian, monotone and stable by substitution is called a reduction ordering. For $\mathcal{F}$ and $\mathcal{X}$ finite, if $\succ$ is monotone and has the subterm property, then it is noetherian [Kruskal 1960]. If, in addition, $\succ$ is stable under substitution, then it is called a simplification ordering.

The notion of strategy is fundamental for computation and deduction, and can be defined in a general way for reduction relations. We concentrate in this work on reductions defined by a rewrite system and define rewriting strategies.

Definition (REWRITING STRATEGY). A *rewriting strategy* S for a rewrite system $\mathcal{R}$ is a subset of the set of all derivations of $\mathcal{R}$. When a rewrite step belongs to a derivation of the strategy S, it is denoted by $\rightarrow_S$ and is called a rewriting step under the strategy S. A term t reducible for $\rightarrow_S$ is said S-reducible; t is an S-normal

form if it is a normal form for $\rightarrow_S$.

This definition of rewriting strategy generalizes the one proposed in [van Oostrom and de Vrijer 2003] because we do not require the derivations to reach normal forms.

A strategy could be described extensively or more suitably by a *strategy language* like in ELAN, Stratego, TOM or Maude. The semantics of such a language is naturally described in the rewriting calculus [Cirstea and Kirchner 2001; Cirstea et al. 2003]. A strategy language involves rules as basic elements and offers strategy combinators and iterators to build more complex strategy expressions. Well-known rewriting strategies allow controlling the application of rules over subterms, performing term traversal and normalizing terms: in this paper, we consider in particular the innermost, outermost and local strategies on operators.

Definition (INNERMOST/OUTERMOST STRATEGY). Let $\mathcal{R}$ a rewrite system on $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Rewriting under the innermost (resp. outermost) strategy is defined as follows: for any term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $t \rightarrow_{\mathcal{R}} t'$ and the rewriting position p in t is such that there is no suffix (resp. prefix) position p' of p such that t rewrites at position p'.

Rewriting strategies may be more complex to define. This is the case for local strategies on operators, used in the OBJ-like languages. We use here the notion of local strategy as expressed in [Goguen et al. 1992].

Definition (LS-STRATEGY). An *LS-strategy* is given by a function LS from $\mathcal{F}$ to the set of lists of integers $\mathcal{L}(\mathbb{N})$, that induces a rewriting strategy as follows.

Given an LS-strategy such that $LS(f) = [i_1, \ldots, i_p]$, $i_j \in [0..arity(f)]$ for all $j \in [1..p]$, for some symbol $f \in \mathcal{F}$, normalizing a term $t = f(t_1, \ldots, t_m) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ with respect to $LS(f) = [i_1, \ldots, i_p]$ consists in normalizing all subterms of t at positions $i_1, \ldots, i_p$ successively, according to the strategy. If there exists $j \in [1..p]$ such that $i_1, \ldots, i_{j-1} \neq 0$ and $i_j = 0$ (0 is the top position), then

—if the current term t' obtained after normalizing $t|_{i_1}, \ldots, t|_{i_{j-1}}$ is reducible at the top position into a term $g(u_1, \ldots, u_n)$, then $g(u_1, \ldots, u_n)$ is normalized with respect to $LS(g)$ and the rest of the strategy $[i_{j+1}, \ldots, j_p]$ is ignored,

—if t' is not reducible at the top position, then t' is normalized with respect to $i_{j+1}, \ldots, i_p$.

From now on, S can be one of the innermost, outermost or local strategy.

3. THE INDUCTIVE PROOF PROCESS

For proving that a rewrite system $\mathcal{R}$ terminates for the strategy S on $\mathcal{T}(\mathcal{F})$, we proceed by induction on $\mathcal{T}(\mathcal{F})$ with a noetherian ordering $\succ$ as noetherian induction relation, assuming that for any t' such that $t \succ t'$, t' terminates. To warrant non emptiness of $\mathcal{T}(\mathcal{F})$, we assume that $\mathcal{F}$ contains at least one constructor constant.

The original idea is here to consider the termination property on terms, instead of considering it on the rewrite systems. Given a rewrite system $\mathcal{R}$ and a strategy S, a term t of $\mathcal{T}(\mathcal{F})$ terminates for the considered strategy S, or S-terminates, iff every S-rewriting derivation starting from t is finite. Termination of rewriting under strategy on the set of ground terms is thus expressed as the property: for all t of $\mathcal{T}(\mathcal{F})$, t S-terminates.

It is worth noticing here that rewriting ground terms is not a restriction for the study of rule-based languages: the requests are indeed terms with variables but they are reduced with a set of rewrite rules built on a disjoint set of variables. So there is no essential difference between the variables in the requests (which are never instantiated) and constants, from the point of view of the rewriting process.

The induction principle on the termination property will be applied on the derivation tree of any term: "for any ground term t, if on any S-derivation issued from t, there exists t' such that $t \succ t'$, then t S-terminates". On such t', the induction hypothesis can be applied, so each t' terminates, and t terminates too. Of course, generating derivation trees for each term is not a good idea, so we will schematize them into so-called proof trees on which we will lift the induction reasoning.

3.1 Lifting rewriting trees into proof trees

Thus we observe the rewriting derivation tree (for the considered strategy) starting from a ground term $t \in \mathcal{T}(\mathcal{F})$ which is any instance of a term $g(x_1, \ldots, x_m)$, for some defined function symbol $g \in \mathcal{D}$, and variables $x_1, \ldots, x_m$. Proving termination on ground terms amounts at proving that all rewriting derivation trees have only finite branches, using the same induction ordering $\succ$ for all trees.

Each rewriting derivation tree is simulated, using a lifting mechanism, by a proof tree, developed from the patterns $g(x_1, \ldots, x_m)$, by alternatively using two main operations, namely narrowing and abstraction, adapted to the considered rewriting strategy. More precisely, narrowing schematizes all rewriting possibilities of terms. The abstraction process simulates the normalization of subterms in the derivations, according to the strategy. It consists in replacing these subterms by special variables, denoting one of their normal forms, without computing them. This abstraction step is performed on subterms that can be assumed terminating by induction hypothesis.

The common property of the three considered strategies is to allow normalisation of some subterms: all subterms for the innermost case, all subterms having no rewriting position at one of their prefix positions for the outermost case, or all subterms allowed by the local strategy of an operator for local strategies.

The schematization of ground rewriting derivation trees is achieved through constraints. Each node of the developed proof trees is a step of the proof, composed of a current term of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and a set of ground substitutions represented by a constraint progressively built along the successive abstraction and narrowing applications. A node schematizes a set of ground terms: the ground instances of the current term, that are solutions of the constraint.

The constraint is in fact composed of two kinds of formulas: ordering constraints, set to warrant the validity of the inductive steps, and abstraction constraints combined to narrowing substitutions, which effectively define the relevant sets of ground terms. The latter are actually useful for controlling the narrowing process, well known to easily diverge.

The termination proof procedures given in this paper are described by deduction rules applied with a special control $Strat{-}Rules(S)$, depending on the studied rewriting strategy S. To prove termination of $\mathcal{R}$ on any term $t \in \mathcal{T}(\mathcal{F})$ under the strategy S, we consider a so-called reference term $t_{ref} = g(x_1, \ldots, x_m)$ for each defined symbol $g \in \mathcal{D}$, and empty sets $\top$ of constraints. Applying the de-

duction rules according to the control strategy $Strat{-}Rules(S)$ to the initial node $(\{g(x_1, \ldots, x_m)\}, \top, \top)$ builds a proof tree, whose nodes are produced by the inference rules. Branching is produced by the different possible narrowing applications.

Termination is established when the procedure terminates because the deduction rules do not apply anymore and all terminal nodes of all proof trees have an empty set of terms.

3.2 Lifting the inductive reasoning

As said previously, we consider any term of $\mathcal{T}(\mathcal{F})$ as a ground instance of a term t of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ occurring in a proof tree issued from a reference term t_{ref}. Using the termination induction hypothesis on $\mathcal{T}(\mathcal{F})$ naturally leads us to simulate the rewriting relation by two mechanisms:

—first, some subterms t_j of the current term t of the proof tree are supposed to have only terminating ground instances, by induction hypothesis, if $\theta t_{ref} \succ \theta t_j$ for the induction ordering $\succ$ and for every solution θ of the constraint associated to t. They are replaced in t by *abstraction variables* X_j representing respectively one of the normal forms of their ground instances $(\theta t_j)\downarrow$. Reasoning by induction allows us to only suppose the existence of the $(\theta t_j)\downarrow$ *without explicitly computing them*;

—second, narrowing (under the strategy S) the resulting term $u = t[X_j]_{j \in \{p_1, \ldots, p_k\}}$ (where $p_1, \ldots, p_k$ are the positions of the abstracted subterm t_j in t) into terms v, according to the possible instances of the X_j. This corresponds to rewriting (under the strategy S) the possible ground instances of u (characterized by the constraint associated to u) in all possible ways.
In general, the narrowing step of u is not unique. We obviously have to consider all terms v such that θu rewrites into θv, which corresponds to considering all narrowing steps from u.

Then the termination problem of the ground instances of t is reduced to the termination problem of the ground instances of v. If $\theta t_{ref} \succ \theta v$ for every ground substitution θ solution of the constraint associated to v, by induction hypothesis, θv is supposed to be terminating. Else, the process is iterated on v, until getting a term t' such that either $\theta t_{ref} \succ \theta t'$, or $\theta t'$ is irreducible.

We introduce in the next section the necessary concepts to formalize and automate this technique.

4. ABSTRACTION, NARROWING, AND THE INVOLVED CONSTRAINTS

4.1 Ordering constraints

The induction ordering is constrained along the proof by inequalities between terms that must be comparable, each time the induction hypothesis is used in the abstraction mechanism. As we are working with a lifting mechanism on the proof trees with terms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, we directly work with an ordering $\succ_{\mathcal{P}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $t \succ_{\mathcal{P}} u$ implies $\theta t \succ \theta u$, for every solution θ of the constraint associated to u.

Any ordering $\succ_{\mathcal{P}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ satisfying $t \succ_{\mathcal{P}} u$ and which is stable under substitution fulfills the previous implication. The ordering $\succ_{\mathcal{P}}$, defined on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, can then be seen as an extension of the induction ordering $\succ$. For convenience, $\succ_{\mathcal{P}}$ will also be written $\succ$.

This ordering is not defined a priori, but just has to verify inequalities of the form $t > u_1, \ldots, u_m$, accumulated along the proof, and which are called *ordering constraints*. Thus, for establishing the inductive termination proof, it is sufficient to decide whether ordering constraints are satisfiable.

Definition (ORDERING CONSTRAINT). *An ordering constraint* is a pair of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ noted $(t > t')$. It is said to be *satisfiable* if there exists an ordering $\succ$, such that for every instantiation θ whose domain contains $\mathcal{V}ar(t) \cup \mathcal{V}ar(t')$, we have $\theta t \succ \theta t'$. We say that $\succ$ satisfies $(t > t')$.

A conjunction C of ordering constraints is satisfiable if there exists an ordering satisfying all conjuncts. The empty conjunction, always satisfied, is denoted by $\top$.

Satisfiability of a constraint conjunction C of this form is undecidable. But a sufficient condition for an ordering $\succ$ to satisfy C is that $\succ$ is stable under substitution and $t \succ t'$ for any constraint $t > t'$ of C.

Reduction orderings fulfill the first requirement. So in practice, it is sufficient to find a reduction or a simplification ordering $\succ$ such that $t \succ t'$ for any constraint $t > t'$ of C.

Solving ordering constraints in finding simplification orderings is a well-known problem. The simplest way and an automatable way to proceed is to test simple existing orderings like the subterm ordering, the Recursive Path Ordering (RPO) [Dershowitz 1982b], or the Lexicographic Path Ordering (LPO) [Kamin and Lévy 1982]. This is often sufficient for the constraints considered here: as said in the introduction, thanks to the power of induction, they are often simpler than for termination methods directly using ordering for orienting rewrite rules.

If these simple orderings are not powerful enough, automatic solvers like C*i*me [1] can provide adequate polynomial orderings.

4.2 Abstraction

To abstract a term t at positions $p_1, \ldots, p_k$, where the $t|_j$ are supposed to have a normal form $t|_j\downarrow$, we replace the $t|_j$ by abstraction variables X_j representing respectively one of their possible normal forms. Let us define these special variables more formally.

Definition (ABSTRACTION VARIABLE). Let $\mathcal{X}_A$ be a set of variables disjoint from $\mathcal{X}$. Symbols of $\mathcal{X}_A$ are called *abstraction variables*. Substitutions and instantiations are extended to $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$ in the following way: for any substitution σ (resp. instantiation θ) such that $Dom(\sigma)$ (resp. $Dom(\theta)$) contains a variable $X \in \mathcal{X}_A$, σX (resp. θX) is in S-normal form.

Definition (TERM ABSTRACTION). The term $t[t|_j]_{j \in \{p_1, \ldots, p_k\}}$ is said to be *abstracted* into the term u (called *abstraction* of t) at positions $\{p_1, \ldots, p_k\}$ iff $u = t[X_j]_{j \in \{p_1, \ldots, p_k\}}$, where the $X_j, j \in \{p_1, \ldots, p_k\}$ are fresh distinct abstraction variables.

Termination on $\mathcal{T}(\mathcal{F})$ is proved by reasoning on terms with abstraction variables, i.e. on terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. Ordering constraints are extended to pairs of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. When subterms $t|_j$ are abstracted by X_j, we state constraints on

[1] Available at http://cime.lri.fr/

abstraction variables, called *abstraction constraints* to express that their instances can only be normal forms of the corresponding instances of $t|_j$. Initially, they are of the form $t\downarrow = X$ where $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, and $X \in \mathcal{X}_A$, but we will see later how they are combined with the substitutions used for the narrowing process.

4.3 Narrowing

After abstraction of the current term t into $t[X_j]_{j\in\{p_1,\ldots,p_k\}}$, we check whether the possible ground instances of $t[X_j]_{j\in\{p_1,\ldots,p_k\}}$ are reducible, according to the possible values of the instances of the X_j. This is achieved by narrowing $t[X_j]_{j\in\{p_1,\ldots,p_k\}}$.

The narrowing relation depends on the considered strategy S and the usual definition needs to be refined. The first idea is to use innermost (resp. outermost) narrowing. Then, if a position p in a term t is a narrowing position, a suffix (resp. prefix) position of p cannot be a narrowing position too. However, if we consider ground instances of t, we can have rewriting positions p for some instances, and p' for some other instances, such that p' is a suffix (resp. a prefix) of p. So, when narrowing at some position p, the set of relevant ground instances of t is defined by excluding the ground instances that would be narrowable at some suffix (resp. prefix) position of p, that we call S-better position: a position S-better than a position p in t is a suffix position of p if S is the innermost strategy, a prefix position of p if S is the outermost strategy. This definition does not make sense for local strategies: as the redex positions of a term are imposed by the strategy attached to each operator, they are also redex positions for any ground instance of this term. So there is no S-better position in this case.

Moreover, to preserve the fact that a narrowing step of t schematizes a rewriting step of possible ground instances of t, we have to be sure that an innermost (resp. outermost) narrowing redex in t corresponds to the same rewriting redex in a ground instance of t. This is the case only if, in the rewriting chain of the ground instance of t, there is no rewriting redex anymore in the part of the term brought by the instantiation. So before each narrowing step, we schematize the longest rewriting chain of any ground instance of t, whose redexes occur in the variable part of the instantiation, by a linear variable renaming. Linearity is crucial to express that, in the previous rewriting chain, ground instances of the same variables can be reduced in different ways. For the innermost strategy, abstraction of variables performs this schematization. For the outermost strategy, a reduction renaming will be introduced. For local strategies, this variable renaming is not relevant, since by construction, there is no rewriting redex in the part of the term brought by the instantiation.

The S-narrowing steps which apply to a given term t are computed in the following way. After applying the variable renaming to t, 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 S-narrowing position of t, iff there is no S-better position p' of t such that $\sigma t|_{p'}$ unifies with a left-hand side of a rule. Then we look for every S-better position p' than p in t such that $\sigma t|_{p'}$ narrows with some substitution σ' and some rule $l' \to r'$, and we set a constraint to exclude this substitution. So the substitutions used to narrow a term have in general to satisfy a set of disequalities coming from the negation of previous substitutions. To formalize this point, we need the following notations and definitions.

In the following, we identify a substitution $\sigma = (x_1 \mapsto t_1)\ldots(x_n \mapsto t_n)$ on $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$ with the finite set of solved equations $(x_1 = t_1) \wedge \ldots \wedge (x_n = t_n)$, also denoted by the equality formula $\bigwedge_i (x_i = t_i)$, with $x_i \in \mathcal{X} \cup \mathcal{X}_A$, $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, where $=$ is the syntactic equality. We call *negation* $\overline{\sigma}$ of the substitution σ the formula $\bigvee_i (x_i \neq t_i)$.

Definition (CONSTRAINED SUBSTITUTION). A *constrained substitution* σ is a formula $\sigma_0 \wedge \bigwedge_j \bigvee_{i_j} (x_{i_j} \neq t_{i_j})$, where σ_0 is a substitution.

Definition (S-NARROWING). A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$ *S-narrows* into a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$ at the non-variable position p of t, using the rule $l \to r \in \mathcal{R}$ with the constrained substitution $\sigma = \sigma_0 \wedge \bigwedge_{j \in [1..k]} \overline{\sigma_j}$, which is written $t \rightsquigarrow_S^{p, l \to r, \sigma} t'$ iff

$$\sigma_0(l) = \sigma_0(t|_p) \text{ and } t' = \sigma_0(t[r]_p)$$

where σ_0 is the most general unifier of $t|_p$ and l and $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 t|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all position p' which are S-better positions than p in t.

It is always assumed that there is no variable in common between the rule and the term, i.e. that $Var(l) \cap Var(t) = \emptyset$. This requirement of disjoint variables is easily fulfilled by an appropriate renaming of variables in the rules when narrowing is performed. The most general unifier σ_0 used in the above definition can be taken such that its range only contains fresh variables. This is important for controlling the satisfiability of the constraints presented in the next section, as shown in the proof of the narrowing lemma given in the appendix.

Since we are interested in the narrowing substitution applied to the current term t, but not in its definition on the variables of the left-hand side of the rule, the narrowing substitutions can be restricted to the variables of the narrowed term t.

The following lifting lemma, generalized from [Middeldorp and Hamoen 1994], ensures the correspondence between the narrowing relation, used during the proof, and the rewriting relation.

LEMMA (S-LIFTING LEMMA). *Let $\mathcal{R}$ be a rewrite system. Let $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, α a ground substitution such that αs is S-reducible at a non variable position p of s, and $\mathcal{Y} \subseteq \mathcal{X}$ a set of variables such that $Var(s) \cup Dom(\alpha) \subseteq \mathcal{Y}$. If $\alpha s \to_S^{p, l \to r} t'$, then there exist a term $s' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and substitutions $\beta, \sigma = \sigma_0 \wedge \bigwedge_{j \in [1..k]} \overline{\sigma_j}$ such that:*

1. $s \rightsquigarrow_S^{p, l \to r, \sigma} s'$,
2. $\beta s' = t'$,
3. $\beta \sigma_0 = \alpha[\mathcal{Y} \cup Var(l)]$
4. β satisfies $\bigwedge_{j \in [1..k]} \overline{\sigma_j}$

where σ_0 is the most general unifier of $s|_p$ and l and $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 s|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all position p' which are S-better positions than p in s.

4.4 Cumulating constraints

Abstraction constraints have to be combined with the narrowing constrained substitutions to characterize the ground terms schematized by the proof trees. A

narrowing step effectively corresponds to a rewriting step of ground instances of t if the narrowing constrained substitution σ is *compatible* with the abstraction constraint formula A associated to t (i.e. $A \wedge \sigma$ is satisfiable). Else, the narrowing step is meaningless. So the narrowing constraint attached to the narrowing step is added to A. Hence the introduction of abstraction constraint formulas.

Definition (ABSTRACTION CONSTRAINT FORMULA). An *abstraction constraint formula* (ACF in short) is a formula $\bigwedge_i(t_i\downarrow = t'_i) \wedge \bigwedge_j(x_j = t_j) \wedge \bigwedge_k \bigvee_{l_k}(u_{l_k} \neq v_{l_k})$, where $t_i, t'_i, t_j, u_{l_k}, v_{l_k} \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, $x_j \in \mathcal{X} \cup \mathcal{X}_A$.

Definition (SATISFIABILITY OF AN ACF). An *abstraction constraint formula* $\bigwedge_i(t_i\downarrow = t'_i) \wedge \bigwedge_j(x_j = t_j) \wedge \bigwedge_k \bigvee_{l_k}(u_{l_k} \neq v_{l_k})$, is *satisfiable* iff there exists at least one instantiation θ such that $\bigwedge_i(\theta t_i\downarrow = \theta t'_i) \wedge \bigwedge_j(\theta x_j = \theta t_j) \wedge \bigwedge_k \bigvee_{l_k}(\theta u_{l_k} \neq \theta v_{l_k})$. The instantiation θ is then said to satisfy the ACF A and is called solution of A.

Integrating a constrained substitution $\sigma = \sigma_0 \wedge \bigwedge_i \bigvee_{j_i}(x_{j_i} \neq t_{j_i})$ to an ACF A is done by adding the formula defining σ to A, thus giving the formula $A \wedge \sigma$. For a better readability on examples, we can propagate σ into A (by applying σ_0 to A), thus getting instantiated abstraction constraints of the form $t_i\downarrow = t'_i$ from initial abstraction constraints of the form $t_i\downarrow = X_i$.

An ACF A is attached to each term t in the proof trees; its solutions characterize the interesting ground instances of this term, i.e. the θt such that θ is a solution of A. When A has no solution, the current node of the proof tree represents no ground term. Such nodes are then irrelevant for the termination proof. Detecting and suppressing them during a narrowing step allows us to control the narrowing mechanism. So we have the choice between generating only the relevant nodes of the proof tree, by testing the satisfiability of A at each step, or stopping the proof on a branch on an irrelevant node, by testing the unsatisfiability of A. These are both facets of the same question, but in practice, they are handled in different ways.

The satisfiability of A is in general undecidable. The disequality part of an ACF is a particular instance of a disunification problem (a quantifier free equational formula), whose satisfiability has been addressed in [Comon 1991], that provides rules to transform any disunification problem into a solved form. Testing the satisfiability of the equational part of an ACF is undecidable in general, but sufficient conditions can be given, relying on a characterization of normal forms.

The unsatisfiability of A is also undecidable in general, but simple automatable sufficient conditions can be used, very often applicable in practice. They rely on reducibility, unifiability, narrowing and constructor tests.

According to Definition 4.8, an ACF $\bigwedge_i(t_i\downarrow = t'_i) \wedge \bigwedge_j(x_j = t_j) \wedge \bigwedge_k \bigvee_{l_k}(u_{l_k} \neq v_{l_k})$ is unsatisfiable if for instance, one of its conjunct $t_i\downarrow = t'_i$ is unsatisfiable, i.e. is such that $\theta t'_i$ is not a normal form of θt_i for any ground substitution θ. Hence, we get four sufficient conditions for unsatisfiability of an abstraction constraint $t\downarrow = t'$:

Case 1:. $t\downarrow = t'$, with t' reducible. Indeed, in this case, any ground instance of t' is reducible, and hence cannot be a normal form.

Case 2:. $t\downarrow = t' \wedge \ldots \wedge t'\downarrow = t''$, with t' and t'' not unifiable. Indeed, any ground substitution θ satisfying the above conjunction is such that (1) $\theta t\downarrow = \theta t'$ and (2) $\theta t'\downarrow = \theta t''$. In particular, (1) implies that $\theta t'$ is in normal form and hence (2) imposes $\theta t' = \theta t''$, which is impossible if t' and t'' are not unifiable.

Case 3:. $t\downarrow = t'$ where $top(t)$ is a constructor, and $top(t) \neq top(t')$. Indeed, if the top symbol of t is a constructor c, then any normal form of any ground instance of t is of the form $c(u)$, where u is a ground term in normal form. The above constraint is therefore unsatisfiable if the top symbol of t' is g, for some $g \neq c$.

Case 4:. $t\downarrow = t'$ with $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X}_A)$ not unifiable and $\bigwedge_{t \rightsquigarrow_S v} v\downarrow = t'$ unsatisfiable. This criterion is of interest if unsatisfiability of each conjunct $v\downarrow = t'$ can be shown with one of the four criteria we present here.

So both the satisfiability and the unsatisfiability checks need to use sufficient conditions. But in the first case, the proof process stops with failure as soon as the satisfiability of A cannot be proved. In the second one, it can go on, until A is proved to be unsatisfiable, or until other stopping conditions are fulfilled.

Let us now come back to ordering constraints. If we check the satisfiability of A at each step, we only generate nodes in the proof trees, that represent non empty sets of ground terms. So in fact, the ordering constraints of C have not to be satisfied for every ground instance, but only for those instances that are solution of A, hence the following definition, that can be used instead of Definition 4.1, when constraints of this definition cannot be proved satisfiable, and solutions of A can easily be characterized.

Definition (CONSTRAINT PROBLEM). Let A be an abstraction constraint formula and C a conjunction of ordering constraints. The constraint problem C/A is satisfied by an ordering $\succ$ iff for every instantiation θ satisfying A, then $\theta t \succ \theta t'$ for every conjunct $t > t'$ of C. C/A is satisfiable iff there exists an ordering $\succ$ as above.

Note that C/A may be satisfiable even if A is not.

4.5 Relaxing the induction hypothesis

It is important to point out the flexibility of the proof method that allows the combination with auxiliary termination proofs using different techniques: when the induction hypothesis cannot be applied on a term t introduced along the proof, i.e. when it is not possible to decide whether the ordering constraints are satisfiable, it is often possible to prove termination (for the considered strategy) of any ground instance of the current term t by another way. In the following we use a predicate $TERMIN(S, t)$ that is true for t iff every ground instance of t terminates for the considered strategy S.

To establish $TERMIN(S, t)$, decidable sufficient conditions exist, applicable in practice, because the predicate is only considered for particular terms introduced along the proof, and not for any term.

In particular, $TERMIN(S, t)$ is true when every instance of t is in normal form. This is the case when t is not narrowable, and all variables of t are in $\mathcal{X}_A$. Indeed, by Lifting Lemma and Definition 4.2, every instance of t is in normal form. This includes the cases where t itself is an abstraction variable, and where t is a non narrowable ground term.

Every instance of a narrowable t whose variables are all in $\mathcal{X}_A$, and whose narrowing substitutions are not compatible with A, is also in normal form. As said in Section 4.4, these narrowing possibilities do not represent any reduction step for the ground instances of t, which are then irreducible.

Otherwise, in many cases, for proving that $TERMIN(S, t)$ is true, the notion of usable rules [Arts and Giesl 1997] is relevant. Given a rewrite system $\mathcal{R}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, the *usable rules* of t are a subset of $\mathcal{R}$, which is a computable superset of the rewrite rules that are likely to be used in any rewriting chain (for the standard rewriting relation) starting from any ground instance of t, until its ground normal forms are reached, if they exist.

Proving termination of every ground instance of t then comes down to proving termination of its usable rules, which is in general much easier than proving termination of the whole rewrite system $\mathcal{R}$. If there exists a reduction ordering $\succ_N$ that orients these rules, any ground instance αt is bound to terminate for the standard rewriting relation, and then for the rewriting strategy S. Indeed, if $\alpha t \to t_1 \to t_2 \to \ldots$, then, thanks to the previous hypotheses, $\alpha t \succ_N t_1 \succ_N t_2 \succ_N \ldots$ and, since the ordering $\succ_N$ is noetherian, the rewriting chain cannot be infinite. If an appropriate reduction ordering cannot be found, termination of the usable rules may also be proved with our inductive process itself. The fact that the induction ordering used for usable rules is independent of the main induction ordering, makes the proof very flexible. Complete results on usable rules for the innermost strategy are given in Section 6.2. For the outermost and local strategies, this is developed in [Fissore et al. 2002b] and [Fissore et al. 2001].

5. THE TERMINATION PROOF PROCEDURE

5.1 Strategy-independent proof steps

We are now ready to describe the different steps of the proof mechanism presented in Section 3.

The proof steps generate proof trees in transforming 3-tuples (T, A, C) where

—T is a set of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, containing the current term t whose termination has to be proved. T is either a singleton or the empty set. For local strategies, the term is enriched by the list of positions where t has to be evaluated, $LS(top(t))$. This is denoted by $u^{LS(top(t))}$.

—A is a conjunction of abstraction constraints.

—C is a conjunction of ordering constraints.

Starting from initial nodes $(T = \{t_{ref} = g(x_1, \ldots, x_m)\}, A = \top, C = \top)$, where $g \in \mathcal{D}$, the proof process consists in iterating the following steps:

—The first step abstracts the current term t at given positions $p_1, \ldots, p_k$. If the conjunction of ordering constraints $\bigwedge_j t_{ref} > t|_j$ is satisfiable for some $j \in \{p_1, \ldots, p_k\}$, we suppose, by induction, the existence of irreducible forms for the ground instances of the $t|_j$. We must have $TERMIN(S, t|_j)$ for the other $t|_j, j \in \{p_1, \ldots, p_k\}$. Then, $t|_{p_1}, \ldots, t|_{p_k}$ are abstracted into abstraction variables $X_{p_1}, \ldots, X_{p_k}$. The abstraction constraints $t|_{p_1}\!\downarrow = X_{p_1}, \ldots, t|_{p_k}\!\downarrow = X_{p_k}$ are added to the ACF A. We call that step the *abstract* step.

—The second step narrows the resulting term u with all possible rewrite rules of the rewrite system $\mathcal{R}$, and all possible substitutions σ, into terms v, according to Definition 4.5. This branching step creates as many nodes as narrowing possibilities. The substitution σ is added to A. This is the *narrow* step.

—We then have a *stop* step halting the proof process on the current branch of the proof tree, when A is detected to be unsatisfiable, or when the ground instances of the current term can be stated terminating for the considered rewriting strategy. This happens when the whole current term t can be abstracted, i.e. when the induction hypothesis applies on it, or when we have $TERMIN(S, t)$.

The satisfiability and unsatisfiability tests of A are integrated in the previously presented steps. If testing the unsatisfiability of A is chosen, the test is integrated in the *stop* step. If testing the satisfiability of A is chosen, the test is made at each attempt of an abstraction or a narrowing step, which are then effectively performed only if A can be proved satisfiable. Otherwise, the proof cannot go on anymore and stops with failure.

As we will see later, for a given rewriting strategy S, these proof steps are instantiated by more precise mechanisms, depending on the strategy S, and taking advantage of its specificity. We will define these specific instances by inference rules.

5.2 Discussion on abstraction and narrowing positions

There are different ways to simulate the rewriting relation on ground terms, using abstraction and narrowing.

For example, the abstraction positions can be chosen to abstract the greatest subterms in the term, that are the immediate subterms of the term. Then, if a narrowing step follows, the abstracted term has to be narrowed in all possible ways at the top position only. This may yield a deadlock if some of the direct subterms cannot be abstracted.

We can instead abstract all greatest possible subterms of $t = f(t_1, \ldots, t_n)$. More concretely, we try to abstract $t_1, \ldots, t_n$ and, for each $t_i = g(t'_1, \ldots, t'_p)$ that cannot be abstracted, we try to abstract $t'_1, \ldots, t'_p$, and so on. In the worst case, we are driven to abstract leaves of the term, which are either variables, that do not need to be abstracted if they are abstraction variables, or constants. Obviously, usable rules can be used to prove termination of constants, and then to allow them to be abstracted.

On the contrary, we can choose in priority the smallest possible subterms u_i, that are constants or variables. The ordering constraints $t > u_i$ needed to apply the induction hypothesis, and then to abstract the term, are easier to satisfy than in the previous case since the u_i are smaller.

Beyond these cases, there are a finite but possibly big number of ways to choose the positions where terms are abstracted. Anyway it is not useful to abstract the subterms, whose ground instances are in normal form. Identifying these subterms is performed using the conditions given in Section 4.5.

From the point of view of the narrowing step following the abstraction, there is no general optimal abstracting choice either: the greater the term to be narrowed, the greater is the possible number of narrowing positions. On another side, more general the term to be narrowed, greater is the possible number of narrowing substitutions for a given redex.

5.3 How to combine the proof steps

The previous proof steps, applied to every reference term $t_{ref} = g(x_1, \ldots, x_m)$, where $x_1, \ldots, x_m \in \mathcal{X}$ and $g \in \mathcal{D}$, can be combined in the same way whatever

$S \in \{Innermost, Outermost, LS\}$:

$$Strat{-}Rules(S) = repeat^*(try(abstract),\ try(narrow),\ try(stop)).$$

"$repeat^*(T_1, \ldots, T_n)$" repeats the control strategies of the sequence $(T_1, \ldots, T_n)$ until none of them is applicable anymore. The operator "try" is a parametric operator that can be instantiated, following S, by $try{-}skip(T)$, expressing that the control strategy or rule T is tried, and skipped when it cannot be applied, or by $try{-}stop(T)$, stopping $Strat{-}Rules(S)$ if T cannot be applied.

5.4 The termination theorem

For $S \in \{Innermost, Outermost, LS\}$, we write $SUCCESS(g, \succ)$ if the application of $Strat{-}Rules(S)$ on $(\{g(x_1, \ldots, x_m)\}, \top, \top)$ gives a finite proof tree, whose sets C of ordering constraints are satisfied by a same ordering $\succ$, and whose leaves are either nodes of the form $(\emptyset, A, C)$ or nodes whose set of constraints A is unsatisfiable. This general definition of the *SUCCESS* predicate holds whatever the rewriting strategy instantiation, whatever the corresponding inference rules, and the way to apply them.

Our main result states that termination under a strategy S is equivalent to finiteness of the proof trees described above.

THEOREM 5.1. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols containing at least one constructor constant. Every term of $\mathcal{T}(\mathcal{F})$ terminates under the strategy S iff there exists a noetherian ordering $\succ$ such that for each symbol $g \in \mathcal{D}$, we have $SUCCESS(g, \succ)$.*

The proof is given in the appendix. Intuitively, if every ground term is S-terminating, the construction of the proof trees stops by applying the *stop* step on each branch, which is possible since by hypothesis on any term t, $TERMIN(S, t)$. In this case, the noetherian ordering required for the proof is simply the subterm ordering. For the converse part, we need an emptiness lemma, an abstraction lemma, a narrowing lemma, and a stopping lemma to prove S-termination for all ground instances of terms of the proof tree that satisfy the associated constraint.

We are now ready to instantiate the general proof process, according to the different rewriting strategies.

6. THE INNERMOST CASE

6.1 Abstraction and narrowing

As said before, when rewriting according to the innermost principle, the ground instances of variables have to be normalized before a redex appears higher in the term. The variable renaming performed before narrowing corresponds here to abstracting variables in the current term. Then, here, narrowing has only to be performed on terms of $\mathcal{T}(\mathcal{F}, \mathcal{X}_A)$.

Moreover, for the most general unifiers σ produced during the proof process, all variables of $Ran(\sigma)$ are abstraction variables. Indeed, by Definition 4.2, if $X \in Dom(\sigma)$, σX is in normal form, as well as θX for any instantiation θ. By definition of the innermost strategy, this requires that variables of σX can only be instantiated by terms in normal form, i.e. variables of σX are abstraction variables.

Then, since before the first narrowing step, all variables are renamed into variables of $\mathcal{X}_A$, and the narrowing steps only introduce variables of $\mathcal{X}_A$, variable renamings are superfluous before the further narrowing steps.

6.2 Relaxing the induction hypothesis

To establish $TERMIN(Innermost, t)$, a simple narrowing test of t can first be tried. Except for the initial node, the variables of t are in $\mathcal{X}_A$. If t is not narrowable, or if t is narrowable with a substitution σ that is not compatible with A, then every ground instance of t is in innermost normal form. Else, we compute the usable rules.

When t is a variable of $\mathcal{X}$, the set of usable rules of t is $\mathcal{R}$ itself. When $t \in \mathcal{X}_A$, the set of usable rules of t is empty, since the only possible instances of such a variable are ground terms in normal form. Otherwise, it is recursively computed on the term structure.

Definition Usable rules. Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols. Let $Rls(f) = \{l \to r \in \mathcal{R} \mid root(l) = f\}$. For any $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, the set of usable rules of t, denoted $\mathcal{U}(t)$, is defined by:

—$\mathcal{U}(t) = \mathcal{R}$ if $t \in \mathcal{X}$,
—$\mathcal{U}(t) = \emptyset$ if $t \in \mathcal{X}_A$,
—$\mathcal{U}(f(u_1, \ldots, u_n)) = Rls(f) \cup \bigcup_{i=1}^{n} \mathcal{U}(u_i) \cup \bigcup_{l \to r \in Rls(f)} \mathcal{U}(r)$.

Lemma 6.2. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. For any ground instance αt of t and any rewrite chain $\alpha t \to^{p_1, l_1 \to r_1} t_1 \to^{p_2, l_2 \to r_2} t_2 \to \ldots \to^{p_n, l_n \to r_n} t_n$, then $l_i \to r_i \in \mathcal{U}(t)$, $\forall i \in [1..n]$.*

A sufficient criterion for ensuring standard termination (and then innermost termination) of any ground instance of a term t can be given.

Proposition 6.3. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols, and t a term of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. If there exists a reduction ordering $\succ$ such that for every $l \to r \in \mathcal{U}(t)$, we have $l \succ r$, then any ground instance of t is terminating.*

6.3 The innermost termination proof procedure

The inference rules **Abstract**, **Narrow** and **Stop** respectively instantiate the proof steps *abstract*, *narrow*, and *stop* defined in Section 5.1. They are given in Table I. Their application conditions depend on whether the satisfiability of A or the unsatisfiability of A is checked. These conditions are specified in Tables II and III respectively.

As said above, the ground terms whose termination is studied are defined by the solutions of A. When the satisfiability of A is checked at each inference step, the nodes of the proof tree exactly model the ground terms generated during the rewriting derivations. The satisfiability of A, although undecidable in general, can be proved by exhibiting a ground substitution satisfying the constraints of A.

When the satisfiability of A is not checked, nodes are generated in the proof tree, that can represent empty sets of ground terms, so the generated proof trees can have branches that do not represent any derivation on the ground terms. The unsatisfiability test of A is only used to stop the development of meaningless branches as soon as possible, with the sufficient conditions presented in Section 4.4.

Table I. Inference rules for the innermost strategy

Abstract: $\dfrac{\{t\},\quad A,\quad C}{\{u\},\ A \wedge t|_{i_1}\!\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\!\downarrow = X_{i_p},\ C \wedge H_C(t|_{i_1}) \ldots \wedge H_C(t|_{i_p})}$

where t is abstracted into u at positions $i_1, \ldots, i_p \neq \epsilon$
if $COND{-}ABSTRACT$

Narrow: $\dfrac{\{t\},\quad A,\quad C}{\{u\},\quad A \wedge \sigma,\quad C}$

if $t \rightsquigarrow^{\sigma}_{Innermost} u$ and $COND{-}NARROW$

Stop: $\dfrac{\{t\},\quad A,\quad C}{\emptyset,\quad A \wedge H_A(t),\quad C \wedge H_C(t)}$

if $COND{-}STOP$

and $H_A(t) = \begin{cases} \top & \text{if any ground instance of } t \\ & \text{is in normal form} \\ t\!\downarrow = X & \text{otherwise.} \end{cases}$

$H_C(t) = \begin{cases} \top & \text{if } TERMIN(Innermost, t) \\ t_{ref} > t & \text{otherwise.} \end{cases}$

Table II. Conditions for inference rules dealing with satisfiability of A

$COND{-}ABSTRACT:\quad (A \wedge t|_{i_1}\!\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\!\downarrow = X_{i_p})$
and $(C \wedge H_C(t|_{i_1}) \ldots \wedge H_C(t|_{i_p}))$ are satisfiable

$COND{-}NARROW:\quad A \wedge \sigma$ is satisfiable

$COND{-}STOP:\quad (A \wedge H_A(t))$ and $(C \wedge H_C(t))$ are satisfiable

Table III. Conditions for inference rules dealing with unsatisfiability of A

$COND{-}ABSTRACT:\quad C \wedge H_C(t|_{i_1}) \ldots \wedge H_C(|t_{i_p})$ is satisfiable

$COND{-}NARROW:\quad true$

$COND{-}STOP:\quad (C \wedge H_C(t))$ is satisfiable or A is unsatisfiable.

Once instantiated, the control strategy $Strat{-}Rules(S)$ simply becomes:

$$repeat * (try{-}skip(\mathbf{Abstract}), try{-}stop(\mathbf{Narrow}), try{-}skip(\mathbf{Stop}))$$

with conditions of Table II, and

$$repeat * (try{-}skip(\mathbf{Abstract}), try{-}skip(\mathbf{Narrow}), try{-}skip(\mathbf{Stop}))$$

with conditions of Table III. Note that **Narrow** with conditions of Table II is the only rule stopping the proof procedure when it cannot be applied: when $A \wedge \sigma$ is

Table IV. Conditions for inference rules dealing with satisfiability of A

$COND{-}ABSTRACT$:	$(C \wedge H_C(t\vert_{i_1}) \ldots \wedge H_C(t\vert_{i_p}))$ is satisfiable
$COND{-}NARROW$:	$A \wedge \sigma$ is satisfiable
$COND{-}STOP$:	$(C \wedge H_C(t))$ is satisfiable

satisfiable, the narrowing step can be applied, while, if satisfiability of $A \wedge \sigma$ cannot be proved, the procedure stops.

The procedure can diverge, with infinite alternate applications of **Abstract** and **Narrow**. With conditions of Table II, it can stop on **Narrow** with a node of the form $(\{t\} \neq \emptyset, A, C)$ on at least one branch of the proof tree. In both cases, nothing can be said on termination. Termination is proved when, for all proof trees, the procedure stops with an application of **Stop** on each branch, generating only final nodes of the form $(\emptyset, A, C)$.

According to the strategy $Strat{-}Rules(Innermost)$, testing the satisfiability of A in conditions of Table II can be optimized on the basis of the following remarks. In the first application of **Abstract** for each initial node, $(A \wedge t|_{i_1}\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\downarrow = X_{i_p}) = (\top \wedge x_1\downarrow = X_1 \ldots \wedge x_m\downarrow = X_m)$, which is always satisfiable, since the signature admits at least one constructor constant. Moreover, the following possible current application of **Abstract** comes after an application of **Narrow**, for which it has been checked that $A \wedge \sigma$ is satisfiable. So $(A \wedge \sigma \wedge t|_{i_1}\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\downarrow = X_{i_p})$ is also satisfiable since $X_{i_1}, \ldots, X_{i_p}$ are fresh variables, not used in $A \wedge \sigma$. So it is useless to verify satisfiability of $(A \wedge t|_{i_1}\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\downarrow = X_{i_p})$ in $COND{-}ABSTRACT$.

In a similar way, as **Stop** is applied with a current abstraction constraint formula A, which is satisfiable, $A \wedge t\downarrow = X$ is also satisfiable since X is a fresh variable, not used in A. So it is also useless to verify that $A \wedge t\downarrow = X$ is satisfiable in $COND{-}STOP$.

This leads to the conditions expressed in Table IV, simplifying those of Table II.

6.4 Examples

For a better readability, when a constrained substitution σ is added to the ACF A, we propagate it into A in applying the substitution part σ_0 of σ to A.

Example 6.4. Let R be the previous example of Toyama. We prove that R is innermost terminating on $\mathcal{T}(\mathcal{F})$, where $\mathcal{F} = \{f{:}3, g{:}2, 0{:}0, 1{:}0\}$.

$$\begin{array}{l} f(0,1,x) \to f(x,x,x) \\ g(x,y) \to x \\ g(x,y) \to y \end{array}$$

The defined symbols of $\mathcal{F}$ are here f and g. Applying the rules on $f(x_1, x_2, x_3)$, we get:

$$
\begin{array}{c}
t_{ref} = f(x_1, x_2, x_3) \\
A = \top \\
C = \top \\
\Big\downarrow \textbf{Abstract} \\
f(X_1, X_2, X_3) \\
A = (x_1\!\downarrow = X_1 \wedge x_2\!\downarrow = X_2 \wedge x_3\!\downarrow = X_3) \\
C = (f(x_1, x_2, x_3) > x_1, x_2, x_3) \\
\sigma{=}(X_1{=}0 \wedge X_2{=}1) \Big\downarrow \textbf{Narrow} \\
f(X_3, X_3, X_3) \\
A = (x_1\!\downarrow = 0 \wedge x_2\!\downarrow = 1 \wedge x_3\!\downarrow = X_3) \\
C = (f(x_1, x_2, x_3) > x_1, x_2, x_3) \\
\Big\downarrow \textbf{Stop} \\
\emptyset \\
A = (x_1\!\downarrow = 0 \wedge x_2\!\downarrow = 1 \wedge x_3\!\downarrow = X_3) \\
C = (f(x_1, x_2, x_3) > x_1, x_2, x_3)
\end{array}
$$

Abstract applies since $f(x_1, x_2, x_3) > x_1, x_2, x_3$ is satisfiable by any simplification ordering.

If we are using the conditions for inference rules dealing with the satisfiability of A given in Table IV, we have to justify the **Narrow** application. Here, **Narrow** applies because $A \wedge \sigma = (x_1\!\downarrow = 0 \wedge x_2\!\downarrow = 1 \wedge x_3\!\downarrow = X_3)$, where $\sigma = (X_1 = 0 \wedge X_2 = 1)$, is satisfiable by any ground instantiation θ such that $\theta x_1 = 0$, $\theta x_2 = 1$ and $\theta x_3 = \theta X_3 = 0$.

Then **Stop** applies because $f(X_3, X_3, X_3)$ is a non narrowable term whose all variables are abstraction variables, and hence we have $TERMIN(Innermost, f(X_3, X_3, X_3))$.

Considering now $g(x_1, x_2)$, we get:

$$
\begin{array}{c}
t_{ref} = g(x_1, x_2) \\
A = \top \\
C = \top \\
\Big\downarrow \textbf{Abstract} \\
g(X_1, X_2) \\
A = (x_1\!\downarrow = X_1 \wedge x_2\!\downarrow = X_2) \\
C = (g(x_1, x_2) > x_1, x_2)
\end{array}
$$

$$
\begin{array}{ccc}
\xrightarrow{\textbf{Narrow}\ \sigma=Id} & & \xrightarrow{\textbf{Narrow}\ \sigma=Id} \\
\begin{array}{c} X_1 \\ A = (x_1\!\downarrow = X_1 \wedge x_2\!\downarrow = X_2) \\ C = (g(x_1, x_2) > x_1, x_2) \\ \Big\downarrow \textbf{Stop} \\ \emptyset \\ A = (x_1\!\downarrow = X_1 \wedge x_2\!\downarrow = X_2) \\ C = (g(x_1, x_2) > x_1, x_2) \end{array} & & \begin{array}{c} X_2 \\ A = (x_1\!\downarrow = X_1 \wedge x_2\!\downarrow = X_2) \\ C = (g(x_1, x_2) > x_1, x_2) \\ \Big\downarrow \textbf{Stop} \\ \emptyset \\ A = (x_1\!\downarrow = X_1 \wedge x_2\!\downarrow = X_2) \\ C = (g(x_1, x_2) > x_1, x_2) \end{array}
\end{array}
$$

Abstract applies since $g(x_1, x_2) > x_1, x_2$ is satisfiable by the same simplification ordering than for the proof tree of f.

Again, we have to justify the **Narrow** application. Here, **Narrow** applies because $A \wedge \sigma = (x_1{\downarrow} = X_1 \wedge x_2{\downarrow} = X_2)$, where $\sigma = Id$, is satisfiable by any ground instantiation θ such that $\theta x_1 = \theta X_1 = 0$ and $\theta x_2 = \theta X_2 = 0$.

Then **Stop** applies on both branches because X_1 and X_2 are abstraction variables, hence we trivially have $TERMIN(Innermost, X_1)$ and $TERMIN(Innermost, X_2)$.

Example 6.5. Let us now give an example dealing with the unsatisfiability of A and illustrating the relevance of usable rules. Let us consider the following system $\mathcal{R}$:

$$\begin{array}{lll} plus(x,0) & \to x & (1) \\ plus(x,s(y)) & \to s(plus(x,y)) & (2) \\ f(0,s(0),x) & \to f(x,plus(x,x),x) & (3) \\ g(x,y) & \to x & (4) \\ g(x,y) & \to y & (5) \end{array}$$

Let us first note that $\mathcal{R}$ is not terminating, as illustrated by the following cycle, where successive redexes are underlined:

$$\begin{array}{ll} \underline{f(0,s(0),g(0,s(0)))} & \to^{(3)} f(\underline{g(0,s(0))}, plus(g(0,s(0)), g(0,s(0))), g(0,s(0))) \\ & \to^{(4)} f(0, plus(\underline{g(0,s(0))}, g(0,s(0))), g(0,s(0))) \\ & \to^{(5)} f(0, plus(s(0), \underline{g(0,s(0))}), g(0,s(0))) \\ & \to^{(4)} f(0, \underline{plus(s(0),0)}, g(0,s(0))) \\ & \to^{(1)} \underline{f(0,s(0),g(0,s(0)))} \\ & \to^{(3)} \ldots \end{array}$$

Let us prove innermost termination of $\mathcal{R}$ on $\mathcal{T}(\mathcal{F})$, where $\mathcal{F} = \{0:0, s:1, plus:2, g:2, f:3\}$. The defined symbols of $\mathcal{F}$ are $f, plus$ and g.

Let us apply the inference rules checking the unsatisfiability of A, whose conditions are given in Table III. Applying the rules on $f(x_1, x_2, x_3)$, we get:

$$
\begin{array}{c}
f(x_1, x_2, x_3) \\
A = \top \\
C = \top \\
\Big\downarrow \textbf{Abstract} \\
f(X_1, X_2, X_3) \\
A = (x_1\downarrow = X_1 \wedge x_2\downarrow = X_2 \wedge x_3\downarrow = X_3) \\
C = f(x_1, x_2, x_3) > x_1, x_2, x_3 \\
\Big\downarrow {\scriptstyle \sigma=(X_1=0 \wedge X_2=s(0))} \ \textbf{Narrow} \\
f(X_3, plus(X_3, X_3), X_3) \\
A = (x_1\downarrow = 0 \wedge x_2\downarrow = s(0) \wedge x_3\downarrow = X_3) \\
C = f(x_1, x_2, x_3) > x_1, x_2, x_3 \\
\Big\downarrow \textbf{Abstract} \\
f(X_3, X_4, X_3) \\
A = (x_1\downarrow = 0 \wedge x_2\downarrow = s(0) \wedge x_3\downarrow = X_3 \wedge plus(X_3, X_3)\downarrow = X_4) \\
C = f(x_1, x_2, x_3) > x_1, x_2, x_3 \\
\Big\downarrow {\scriptstyle \sigma=(X_3=0 \wedge X_4=s(0))} \ \textbf{Narrow} \\
f(0, plus(0, 0), 0) \\
A = (x_1\downarrow = 0 \wedge x_2\downarrow = s(0) \wedge x_3\downarrow = 0 \wedge plus(0, 0)\downarrow = s(0)) \\
C = f(x_1, x_2, x_3) > x_1, x_2, x_3 \\
\Big\downarrow \textbf{Stop} \\
\emptyset \\
A = (x_1\downarrow = 0 \wedge x_2\downarrow = s(0) \wedge x_3\downarrow = 0 \wedge plus(0, 0)\downarrow = s(0)) \\
C = f(x_1, x_2, x_3) > x_1, x_2, x_3
\end{array}
$$

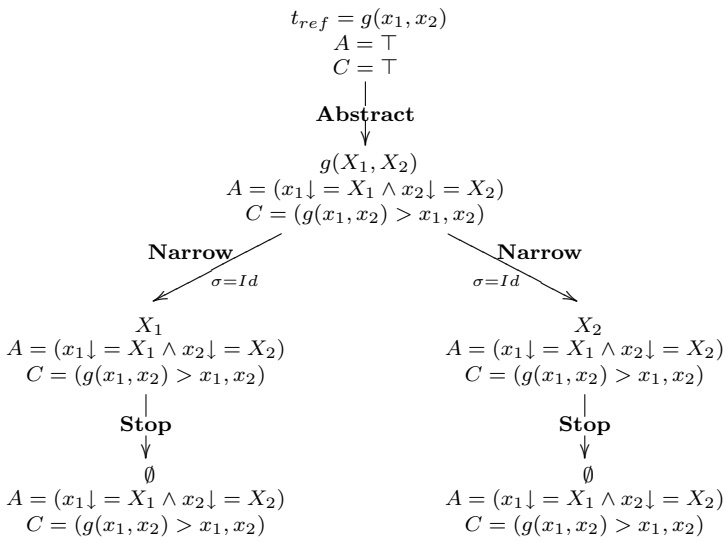
The first **Abstract** applies since $f(x_1, x_2, x_3) > x_1, x_2, x_3$ is satisfiable by any simplification ordering.

Since we are using the inference rules checking the unsatisfiability of A, whose conditions are given in Table III, we do not have to justify the **Narrow** applications.

The second **Abstract** applies by using the *TERMIN* predicate. Indeed, the usable rules of $plus(X_3, X_3)$ consist of the system $\{plus(x, 0) \to x, plus(x, s(y)) \to s(plus(x, y))\}$, that can be proved terminating with any precedence based ordering, independent of the induction ordering, with the precedence $plus \succ_{\mathcal{F}} s$, which ensures the property $TERMIN(Innermost, plus(X_3, X_3))$. Without abstraction here, the process would have generated a branch containing an infinite number of **Narrow** applications.

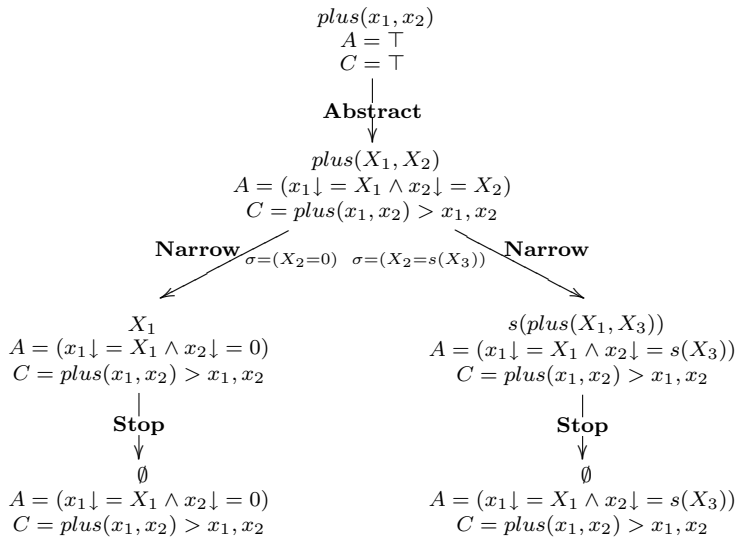
Finally, **Stop** applies because the constraint A becomes unsatisfiable. Indeed, it contains the abstraction constraint $plus(0, 0)\downarrow = s(0)$, which is not true since the unique normal form of $plus(0, 0)$ is 0. Note that if we would have chosen to apply the inference rules checking the satisfiability of A, whose conditions are given in Table IV, then the last narrowing step would not have applied, and would have been replaced by a **Stop** application.

Considering now $g(x_1, x_2)$, we get:



The proof tree is the same as in the previous example. **Abstract** applies since $g(x_1, x_2) > x_1, x_2$ is satisfiable by the simplification ordering used for the first **Abstract** of the previous proof tree.

Let us finally apply the inference rules of Table III on $plus(x_1, x_2)$:



Abstract applies since $plus(x_1, x_2) > x_1, x_2$ is satisfiable by the simplification ordering used for the first **Abstract** of the first proof tree. **Stop** applies on the left branch because X_1 is an abstraction variable. **Stop** applies on the right branch thanks to the *TERMIN* predicate. Indeed, the usable rules of $s(plus(X_1, X_3))$ are $\{plus\ (x, 0) \rightarrow x, plus(x, s(y)) \rightarrow s(plus(x,\ y))\}$ and terminate.

7. THE OUTERMOST CASE

7.1 Abstraction

According to the outermost strategy, abstraction can be performed on subterms t_i only if during their normalization, the t_i's do not introduce outermost redexes higher in the term t. More formally, the induction hypothesis is applied to the subterms $t|_{p_1}, \ldots, t|_{p_k}$ of the current term t, provided $\alpha t_{ref} \succ \alpha t|_{p_1}, \ldots, \alpha t|_{p_k}$ for every ground substitution α, for the induction ordering $\succ$ and provided $u = t[y_1]_{p_1} \ldots [y_n]_{p_k}$ is not narrowable at prefix positions of $p_1, \ldots, p_k$, for the outermost narrowing relation defined below.

7.2 The narrowing mechanism

Outermost narrowing is defined by Definition 4.5, where an S-better position is a prefix position. In order to support intuition, let us consider for instance the 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, $f(g(x_1))$ narrows into a with the first rule and $\sigma = (x_1 = a)$, and into $f(f(g(x_1)))$ with the third rule and the constrained substitution $\sigma = Id \wedge x_1 \neq a$.

In the outermost termination proof, the variable renaming performed before the narrowing step has a crucial meaning for the schematization of outermost derivations. This renaming, applied on the current term t, replaces the variable occurrences $x_1, \ldots, x_m$ of t by new and mutually distinct variables $x'_1, \ldots, x'_m$, defined as follows. Given any ground instance αt, $x'_1, \ldots, x'_m$ represent the first reduced form of $\alpha x_1, \ldots, \alpha x_m$ encountered in any outermost rewriting chain starting from αt, such that the next redex in the chain is not in $\alpha x_1, \ldots, \alpha x_m$ anymore, but higher in the term.

This replacement is memorized in a reduction formula before we apply a step of outermost narrowing to $g(x'_1, \ldots, x'_m)$. The abstraction variables are not renamed: since their ground instances are in normal form, they are not concerned by the rewriting chain schematized by the variable renaming.

Definition 7.1. 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, noted $\rho = (x_1 \rightarrowtail x'_1)...(x_m \rightarrowtail x'_m)$, consists in replacing the x_i by new and mutually distinct variables x'_i in t, giving a term t^ρ. This is denoted by the so-called reduction formula

$$R(t) = t \rightarrowtail t^\rho.$$

Notice that the reduction renaming linearizes the term. For instance, the two occurrences of x in $g(x, x)$ are respectively renamed into x'_1 and x'_2, and $g(x, x) \rightarrowtail g(x'_1, x'_2)$.

Definition 7.2. 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 $R(t) = t \rightarrowtail t^\rho$, where $\rho = (x_1 \rightarrowtail x'_1)...(x_m \rightarrowtail x'_m)$, iff there exists an outermost rewriting chain $\theta t \xrightarrow{*}{}^{p \notin \mathcal{P}os_{\mathcal{F}}(t)}_{Outermost} \theta t^\rho \xrightarrow{}{}^{p \in \mathcal{P}os_{\mathcal{F}}(t)}_{Outermost} u$, i.e. such that:

—either $\theta t^\rho = t[\theta x'_1]_{p_1} \dots [\theta x'_m]_{p_m}$ is the first reduced form of $\theta t = t[\theta x_1]_{p_1} \dots [\theta x_m]_{p_m}$ on this chain having an outermost rewriting position at a non variable position of t, if this position exists,

—or $\theta x'_1 = (\theta x_1\downarrow), \dots, \theta x'_m = (\theta x_m\downarrow)$ if there is no such position.

Before going on, a few remarks on this definition can be made. In the second case of satisfiability, $t[\theta x_1\downarrow]_{p_1} \dots [\theta x_m\downarrow]_{p_m}$ is in outermost normal form. In any case, $R(t)$ is always satisfiable : it is sufficient to take a ground substitution θ such that $t[\theta x_1]_{p_1} \dots [\theta x_m]_{p_m}$ has an outermost rewriting position at a non variable position of t, and then to extend its domain $\{x_1, \dots, x_m\}$ to $\{x_1, \dots, x_m, x'_1, \dots, x'_m\}$ 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 $\theta x_1 = \dots = \theta x_m = \theta x'_1 = \dots = \theta x'_m = u$, with u any ground term in normal form.

However, there may exist several instantiations solutions of such constraints. Let us consider for instance the rewrite system $R = \{f(a) \to f(c), b \to a\}$ and the reduction formula $R(f(x)) = f(x) \rightarrowtail f(x')$. 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).

Notice also that if t is outermost reducible at position p, variables of t whose position is a suffix of p are not affected by the reduction renaming. Indeed, if t is reducible at position p, a ground instance αt of t cannot be outermost reduced in the instance of x, whose positions are suffix of p. So x', representing the first reduced form of αx in any outermost rewriting chain starting from αt, such that the reduction is performed higher in the current term, is equal to x.

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). Then, since $f(x, g(y))$ outermost rewrites at the position of g, the variable y does not need to be renamed. So $R(f(x, g(y))) = (f(x, g(y)) \rightarrowtail f(x', g(y)))$.

Because of the previously defined renaming process, the formula A for cumulating constraints has to be completed in the following way.

Definition 7.3. A *renaming-abstraction constraint formula* (RACF for short) is a formula
$\bigwedge_m u_m \rightarrowtail u^\rho_m \bigwedge_i (t_i\downarrow = t'_i) \wedge \bigwedge_j (x_j = t_j) \wedge \bigwedge_k \bigvee_{l_k} (u_{l_k} \neq v_{l_k})$, where $u_m, u^\rho_m, t_i, t'_i, t_j, u_{l_k}, v_{l_k} \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, $x_j \in \mathcal{X} \cup \mathcal{X}_A$. The empty formula is denoted $\top$.

Definition 7.4. A renaming-abstraction constraint formula
$\bigwedge_m u_m \rightarrowtail u^\rho_m \bigwedge_i (t_i\downarrow = t'_i) \wedge \bigwedge_j (x_j = t_j) \wedge \bigwedge_k \bigvee_{l_k} (u_{l_k} \neq v_{l_k})$ is said to be *satisfiable* iff there exists at least one instantiation θ such that $\bigwedge_i (\theta t_i\downarrow = \theta t'_i) \wedge \bigwedge_j (\theta x_j = \theta t_j) \wedge \bigwedge_k \bigvee_{l_k} (\theta u_{l_k} \neq \theta v_{l_k})$ and θ satisfies $\bigwedge_m u_m \rightarrowtail u^\rho_m$.

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 when 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 of reachability: $\theta t \xrightarrow{*\ p \notin \mathcal{P}os_{\mathcal{F}}(t)}_{Outermost} \theta t^\rho \xrightarrow{p \in \mathcal{P}os_{\mathcal{F}}(t)}_{Outermost} u$.

So here again, we can test either the satisfiability of the formula of cumulated constraints, or the unsatisfiability. As the satisfiability is in general more difficult to show than in the innermost case, we only present here inference rules checking the unsatisfiability.

Unlike in the innermost case, the variables of the narrowed terms here are in $\mathcal{X} \cup \mathcal{X}_A$. Indeed, following the definition of the reduction renaming above, renaming variables of $\mathcal{X}$ still gives variables of $\mathcal{X}$. Moreover, abstraction may let unchanged subterms containing variables of $\mathcal{X}$, in the abstracted term.

7.3 Inference rules for the outermost case

The inference rules **Abstract**, **Narrow** and **Stop** respectively instantiate the proof steps *abstract*, *narrow*, and *stop*.

They work as follows:

—The narrowing step is expressed by a rule **Narrow** applying on $(\{t\}, A, C)$: the variables of t are renamed as specified in Definition 7.1. Then t^ρ is outermost narrowed in all possible ways in one step, with all possible rewrite rules of the rewrite system $\mathcal{R}$, into terms u. For any possible u, we generate the node $(\{u\}, R(t) \wedge A \wedge \sigma, C)$ where σ is the constrained substitution allowing outermost narrowing of t^ρ into u.

—The rule **Abstract** works as in the innermost case, except that the abstraction positions are such that the abstracted term is not narrowable at prefix positions of the abstraction positions.

—The rule **Stop** also works as in the innermost case.

To prove outermost termination of $\mathcal{R}$ on every term $t \in \mathcal{T}(\mathcal{F})$, for each defined symbol $g \in \mathcal{D}$, we apply the rules on the initial node $(\{t_{ref} = g(x_1, \ldots, x_m)\}, \top, \top)$, with the strategy:

$$\mathit{Strat-Rules}(\mathit{Outermost}) = \mathit{repeat} * (\mathit{try-skip}(\mathbf{Abstract}), \mathit{try-skip}(\mathbf{Narrow}), \mathit{try-skip}(\mathbf{Stop})).$$

There are two cases for the behavior of the strategy: either there is a branch in the proof tree with infinite applications of **Abstract** and **Narrow**, in which case we cannot say anything about termination, or the procedure stops on each branch with the rule **Stop**. Then, outermost termination is established, if all proof trees are finite.

According to the remark following Definition 7.2, the reduction formulas in A may often be reduced to simple variable renamings. In this case, A may exclusively contain variable renamings and constrained substitutions, that can be used to show that the ordering constraint needed to apply **Abstract** or **Stop** is satisfiable (see Examples B.1 and B.4 in [Fissore et al. 2002c]). The following lemma can also be used, if satisfiability of C is considered with Definition 4.9 (see Examples B.2, B.3 and B.4 in [Fissore et al. 2002c]). It enables to compare the variables of the current term in a proof tree with the reference term, so it allows in particular to apply **Stop** when the current term is either a variable, or a non narrowable term containing variables of $\mathcal{X}$.

Table V. Inference rules for the outermost strategy

Abstract: $$\frac{\{t\},\quad A,\quad C}{\{u\},\ A \wedge t|_{i_1}\!\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\!\downarrow = X_{i_p},\ C \wedge H_C(t|_{i_1}) \ldots \wedge H_C(t|_{i_p})}$$

where t is abstracted into u at positions $i_1, \ldots, i_p \neq \epsilon$
if $C \wedge H_C(t|_{i_1}) \ldots \wedge H_C(|t_{i_p})$ is satisfiable
and u is not narrowable at prefix positions of $i_1, \ldots, i_p$

Narrow: $$\frac{\{t\},\quad A,\quad C}{\{u\},\quad R(t) \wedge A \wedge \sigma,\quad C}$$

if $t^\rho \rightsquigarrow^\sigma_{Outermost} u$

Stop: $$\frac{\{t\},\quad A,\quad C}{\emptyset,\quad A \wedge H_A(t),\quad C \wedge H_C(t)}$$

if $C \wedge H_C(t)$ is satisfiable or A is unsatisfiable

and $H_A(t) = \begin{cases} \top & \text{if any ground instance of } t \\ & \text{is in normal form} \\ t\!\downarrow = X & \text{otherwise.} \end{cases}$

$H_C(t) = \begin{cases} \top & \text{if } \mathit{TERMIN}(\mathit{Outermost}, t) \\ t_{ref} > t & \text{otherwise.} \end{cases}$

Lemma 7.5. *Let $(\{t_i\}, A_i, C_i)$ be the ith node of any branch of the proof tree obtained by applying the strategy Strat−Rules(Outermost) on $(\{t_{ref}\}, \top, \top)$, and $\succ$ a noetherian ordering having the subterm property. If every reduction formula in A_i can be reduced to a formula $\bigwedge_j x_j = x'_j$, then we have:*

for every variable x of t_i in $\mathcal{X}$: $(t_{ref} > x)/A_i$ is satisfiable by $\succ$.

7.4 Examples

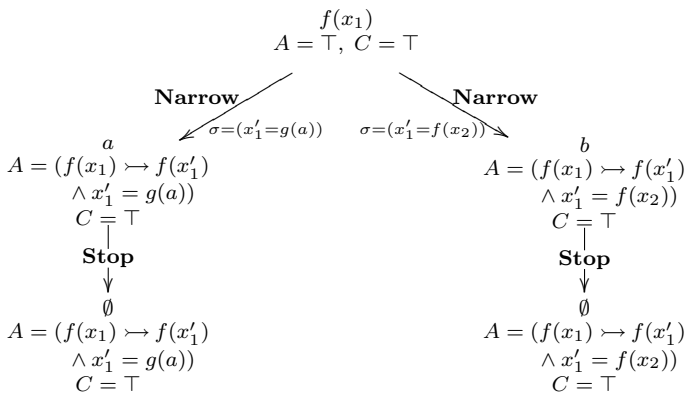
Example 7.6. Consider the previous example

$$\begin{array}{l} f(g(a)) \to a \\ f(f(x)) \to b \\ g(x) \to f(g(x)) \end{array}$$

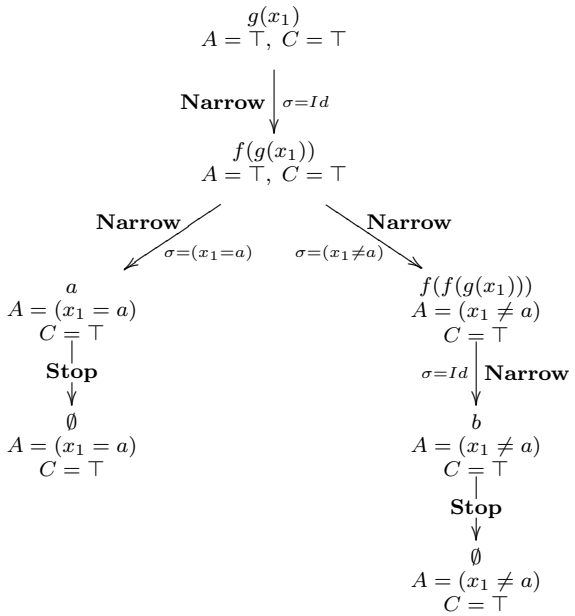
that is outermost terminating, but not terminating for the standard rewriting relation, especially because of the third rule.

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:



The first **Stop** is applied because a is in normal form, the second **Stop** because b is in normal form. Applying the rules on $g(x_1)$, we get:



There is no reduction renaming before the **Narrow** steps, since $g(x_1)$, $f(g(x_1))$ and $f(f(g(x_1)))$ are reducible at prefix positions of the position of x_1.

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 must also consider the third rule if x_1 is such that $x_1 \neq a$ (second branch). **Stop** is applied on a and b as previously.

Example 7.7. Let $\mathcal{R}$ be the rewrite system cited in the introduction, built on $\mathcal{F} = \{c : 2, f : 1, b : 0\}$:

$$\begin{array}{ll} c(x, c(y, z)) & \rightarrow b \\ f(x) & \rightarrow c(x, f(s(x))) \end{array}$$

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

$$\begin{array}{c}
f(x_1) \\
A = \top,\ C = \top \\
\Big\downarrow{\scriptstyle \sigma = Id}\ \textbf{Narrow} \\
c(x_1, f(s(x_1))) \\
A = \top,\ C = \top \\
\Big\downarrow{\scriptstyle \sigma = Id}\ \textbf{Narrow} \\
c(x'_1, c(s(x_1), f(s(s(x_1))))) \\
A = (c(x_1, f(s(x_1))) \rightarrowtail c(x'_1, f(s(x_1)))) \\
C = \top \\
\Big\downarrow{\scriptstyle \sigma = Id}\ \textbf{Narrow} \\
b \\
A = (c(x_1, f(s(x_1))) \rightarrowtail c(x'_1, f(s(x_1)))) \\
C = \top \\
\Big\downarrow\ \textbf{Stop} \\
\emptyset \\
A = (c(x_1, f(s(x_1))) \rightarrowtail c(x'_1, f(s(x_1)))) \\
C = \top
\end{array}$$

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

$$\begin{array}{c}
c(x_1, x_2) \\
A = \top,\ C = \top \\
\Big\downarrow{\scriptstyle \sigma = (x'_2 = c(x_3, x_4))}\ \textbf{Narrow} \\
b \\
A = (c(x_1, x_2) \rightarrowtail c(x'_1, x'_2)) \\
\wedge\ x'_2 = c(x_3, x_4)) \\
C = \top \\
\Big\downarrow\ \textbf{Stop} \\
\emptyset \\
A = (c(x_1, x_2) \rightarrowtail c(x'_1, x'_2)) \\
\wedge\ x'_2 = c(x_3, x_4)) \\
C = \top
\end{array}$$

8. LOCAL STRATEGIES ON OPERATORS

We now address the termination problem for rewriting with local strategies on operators.

8.1 Abstraction and narrowing

A subtle point related to local strategies is that some subterms may never be rewritten according to the LS-strategy. Identifying reducible subterms according

to the LS-strategy is thus an important point, and the notion of LS-position is introduced to denote possible reducible positions of any instantiation of t, with respect to the LS-strategy.

Definition 8.1. A position p of a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is an LS-position in t if the LS-strategy allows rewriting t at position p, or if the LS-strategy allows rewriting any ground instance of t at position p or at a suffix position of p.

This notion enables to identify which variables may be replaced by abstraction variables. The information that variables are abstraction variables can be very important to conclude the proofs here: if the current term is an abstraction variable, its strategy is set to the empty list [] in the *Narrow* step, and then the *Stop* step applies. This information can be easily deduced when new variables are introduced: the abstracting process directly introduces abstraction variables, by definition. But the resulting term may still have variables of $\mathcal{X}$ since the abstracted subterms of a term may not cover all variables of the term.

Moreover, narrowing is performed on terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. Indeed, there is no variable renaming before the narrowing steps, that could transform all variables into abstraction variables. In addition, even if the variables of a narrowed term are all in $\mathcal{X}_A$, the range of the narrowing substitution can introduce variables of $\mathcal{X}$, according to the LS-strategies, if these variables do not appear at LS-positions.

However some variable occurrences can be particularized into variables of $\mathcal{X}_A$ in the narrowing process: the narrowing substitution σ, whose range only contains new variables of $\mathcal{X}$, can be transformed into a new substitution σ_A by replacing some of these variables by abstraction variables. Let us consider an equality of the form $X = u$, introduced by the narrowing substitution σ, where $X \in \mathcal{X}_A$, and $u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. As X is an abstraction variable, every ground instance of u must be in LS-normal form. So the variables in u that occur at an LS-position can be replaced by abstraction variables. Let now μ be the substitution $(x_i = X_i)$, for all $x_i \in Var(u)$ such that $X = u$ is an equality of σ with $X \in \mathcal{X}_A$, $u \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, and x_i occurs at an LS-position in u. Then $\sigma_A = \mu\sigma$.

Combining abstraction and narrowing is achieved here in the following way. Abstraction is tried on the immediate subterms of the current term. If the abstraction is possible, then a narrowing step is applied, only at the top position, which limits the number of narrowing steps, more complicated here than for the other strategies, since, as we will see later, they involve complementary branches.

If **Abstract** cannot be applied at all LS-positions of the term, the simulation of LS-rewriting by abstraction steps is blocked, so the proof process is stopped, and nothing can be concluded about termination.

8.2 The termination proof procedure for local strategies

The inference rules **Abstract**, **Narrow** and **Stop** respectively instantiate the proof steps *abstract*, *narrow*, and *stop*. They work in the following way on a node $(\{t^{[i_1,\ldots,i_p]}\}, A, C)$, where $top(t) = f$ and $LS(f) = [i_1, \ldots, i_p]$.

—The rule **Abstract** can apply:
  —when there exists $k \in [2..p]$, $i_j \neq 0$ for $1 \leq j \leq k-1$ and $i_k = 0$. The term t is abstracted at positions $i_j \neq 0$ for $1 \leq j < k$ if $C \wedge (t_{ref} > t|_{i_j}, 1 \leq j < k)$

is satisfiable. Indeed, by induction hypothesis, all ground instances of $t|_{i_j}, 1 \leq j < k$, LS-terminate. We can instead have $TERMIN(LS, t|_{i_j})$ for some of the previous $t|_{i_j}$. The list of positions then becomes $[0, i_{k+1}, \ldots, i_p]$.

—when there is no position 0 in the strategy of the current term, and the subterms can be abstracted as previously. Any ground instance of the term obtained after abstraction is irreducible, by definition of the LS-strategy, which ends the proof on the current derivation chain. The set containing the current term is then replaced by the empty set.

—when $i_1 = 0$. The rule applies but does not change the node. This is the case where the local strategy of the current term expresses that it has to be narrowed at the top, so there is no abstraction here, and the **Narrow** rule is tried just after. Note that instead of applying **Abstract** without effect, we could have suppressed this case and use here a "try-skip" strategy to enable the application of **Narrow**. But this would be incompatible with the other failure cases of **Abstract**, needing a "try-stop".

—The rule **Narrow** works as follows:

—if the current term t is narrowable at position 0, t is narrowed in all possible ways in one step, with all possible rewrite rules of the rewrite system R, and all possible substitutions σ_i, into $u_i, i \in [1..l]$. Then from the node $(\{t^{[0,i_1,\ldots,i_p]}\}, A, C)$ we generate the nodes $(\{u_i^{LS(top(u_i))}\}, A \wedge \sigma_i, C)$, $i \in [1..l]$, where the σ_i are all most general unifiers allowing narrowing of t into terms u_i, such that $A \wedge \sigma_i$ is satisfiable. This narrowing step means that $\sigma_1 t, \ldots, \sigma_l t$ are all most general instances of t that are reducible at the top position. As a consequence, if $\Phi = \overline{\sigma_1} \wedge \ldots \wedge \overline{\sigma_l}$ is satisfiable, for each instantiation μ satisfying Φ, μt is not reducible at the top position. Then, as these μt have to be reduced at positions $[i_1, \ldots, i_p]$, we also generate the complementary node $(\{t^{[i_1,\ldots,i_p]}\}, A \wedge \bigwedge_{i=1}^{l} \overline{\sigma_i}, C)$.
Let us also notice that if u_i is a variable $x \in \mathcal{X}$, we cannot conclude anything about termination of ground instances of x. Setting $LS(x)$ to $[0]$ or $[]$ would wrongly lead to conclude, with the rule **Narrow**, that ground instances of x are terminating. So we force the proof process to stop in setting $LS(x)$ to a particular symbol $\sharp$. However, if $u_i = X \in \mathcal{X}_A$, $LS(X)$ is set to $[]$, which is coherent with the fact that any ground instance of X is in LS-normal form.

—if t is not narrowable at position 0 or is narrowable with a substitution that is not compatible with the current constraint formula A, then no narrowing is applied and the current term is evaluated at positions following the top position in the strategy. The list of positions then becomes $[i_1, \ldots, i_p]$.

—We can also check for the current term t whether $C \wedge t_{ref} > t$ is satisfiable, or whether we have $TERMIN(LS, t)$. Then, by induction hypothesis, any ground instance of t terminates for the LS-strategy, which ends the proof on the current derivation chain. The **Stop** rule then replaces the set containing the current term by the empty set. It also allows to stop the inference process when the list of positions is empty.

The set of inference rules is given in Table VI. In the conditions of these rules, the satisfiability of A is checked. Working with the unsatisfiability of A would be more

Table VI. Inference rules for t_{ref} LS-termination

Abstract: $$\frac{\{t^{[i_1,\ldots,i_p]}\},\quad A,\quad C}{\{u^S\},\quad A \wedge \bigwedge_{j\in POS}(t|_j\downarrow = X_j),\quad C \wedge \bigwedge_{j\in POS} H_C(t|_j)}$$

where t is abstracted into u at the positions $j \in POS$

if $A \wedge \bigwedge_{j\in POS}(t|_j\downarrow = X_j)$, $C \wedge \bigwedge_{j\in POS} H_C(t|_j)$ are satisfiable and

$POS = \{i_1,\ldots,i_{k-1}\}, S = [0, i_{k+1},\ldots,i_p]$ if $\exists k \in [2..n] : i_1,\ldots,i_{k-1} \neq 0$ and $i_k = 0$

$POS = \{i_1,\ldots,i_p\}, S = []$ if $i_1,\ldots,i_p \neq 0$ or $[i_1,\ldots,i_p] = []$

$POS = \emptyset, S = [i_1,\ldots,i_p]$ if $i_1 = 0$

Narrow: $$\frac{\{t^{[0,i_1,\ldots,i_p]}\},\quad A,\quad C}{\{u^S\},\quad A',\quad C}$$

where $u = u_i, S = LS(top(u_i)), A' = A \wedge \sigma_i$ if $t \rightsquigarrow_{\mathcal{R}}^{\epsilon,\sigma_i} u_i$ and $A \wedge \sigma_i$ is satisfiable

or $u^S = t^{[i_1,\ldots,i_p]}, A' = A \wedge (\bigwedge_{i=1}^{l} \overline{\sigma_i})$, and $\sigma_i, i \in [1..l]$ are all nar. subst. as above

or $u^S = t^{[i_1,\ldots,i_p]}, A' = A$

if t is not narrowable at the top position

or $\forall\sigma$ nar. subst. of t at the top position, $A \wedge \sigma$ is not satisfiable

Stop: $$\frac{\{t^{[i_1,\ldots,i_p]}\},\quad A,\quad C}{\emptyset,\quad A \wedge H_A(t),\quad C \wedge H_C(t)}$$

if $A \wedge H_A(t)$, $C \wedge H_C(t)$ are satisfiable

$$\text{and } H_A(t) = \begin{cases} \top & \text{if } [i_1,\ldots,i_p] = [] \\ & \text{or any ground instance of } t \\ & \text{is in normal form} \\ t\downarrow = X & \text{otherwise.} \end{cases}$$

$$H_C(t) = \begin{cases} \top & \text{if } [i_1,\ldots,i_p] = [] \\ & \text{or } TERMIN(LS, t) \\ t_{ref} > t & \text{otherwise.} \end{cases}$$

technical to handle here than in the innermost case, because of the complementary branches generated by the **Narrow** rule.

The strategy for applying these rules is:

$$repeat * (try{-}stop(\mathbf{Abstract}), try{-}stop(\mathbf{Narrow}), try{-}skip(\mathbf{Stop})).$$

There are here three cases for the behavior of the proof process. It can diverge as previously, or stop and the leaves have then to be considered. The good case is when the process stops and all leaves of all proof trees are of the form $(\emptyset, A, C)$: the termination w.r.t the given LS-strategy is established.

8.3 Examples

Example 8.2. Let us recall the rules of the example given in the introduction.

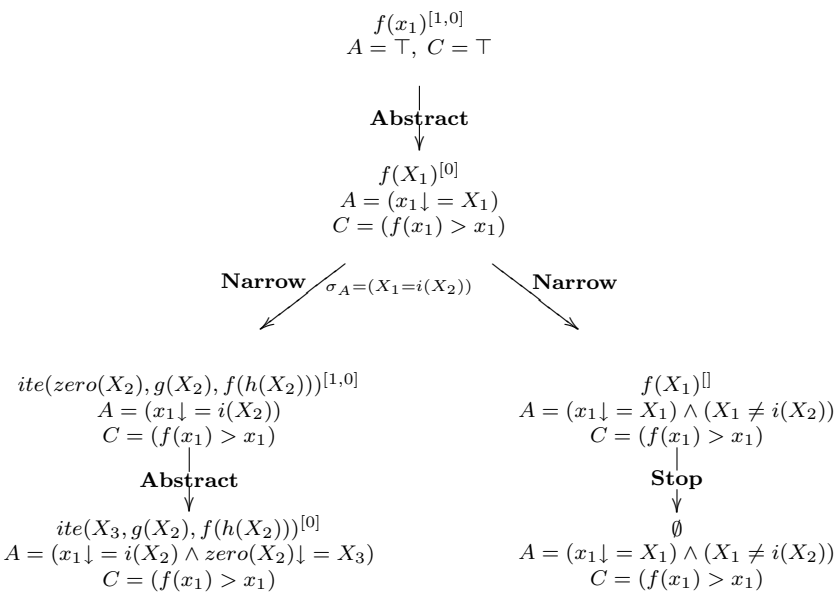
$$\begin{array}{ll}
f(i(x)) & \to ite(zero(x), g(x), f(h(x))) \\
zero(0) & \to true \\
zero(s(x)) & \to false \\
ite(true, x, y) & \to x \\
ite(false, x, y) & \to y \\
h(0) & \to i(0) \\
h(x) & \to s(i(x))
\end{array}$$

The LS-strategy is the following :

—$LS(ite) = [1; 0]$,
—$LS(f) = LS(zero) = LS(h) = [1; 0]$ and
—$LS(g) = LS(i) = [1]$.

Let us prove the termination of this system on the signature $\mathcal{F} = \{f : 1, zero : 1, ite : 3, h : 1, s : 1, i : 1, g : 1, 0 : 0\}$.

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



Abstract applies on $f(x_1)$, since C is satisfiable by any ordering having the subterm property. A is satisfiable with any instantiation θ such that $\theta x_1 = \theta X_1 = 0$.

Narrow expresses the fact that $\sigma f(X_1)$ is reducible if σ is such that $\sigma X_1 = i(X_2)$, and that the other instances ($\sigma' f(X_1)$ with $\sigma' X_1 \neq i(X_2)$) cannot be reduced.

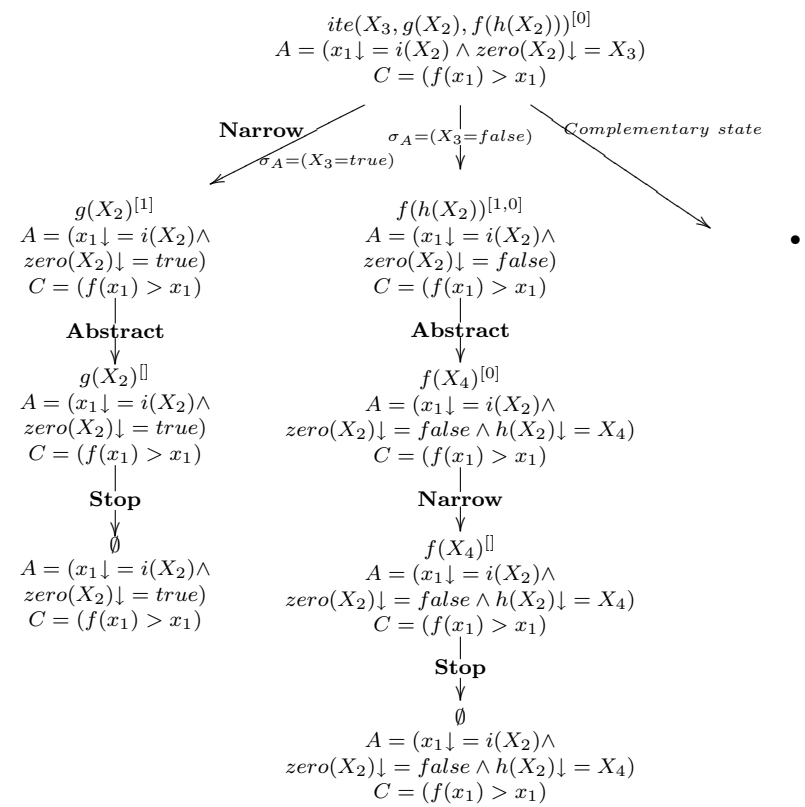
The renaming of x_2 into X_2 in σ_A comes from the fact that x_2 occurs in $i(x_2)$ at an LS-position in $\sigma = (X_1 = i(x_2))$.

Then, the constraint formula A on the left branch is satisfiable by any instantiation θ such that $\theta X_2 = 0$ and $\theta x_1 = i(0)$. The constraint formula on the complementary branch is satisfied by any instantiation θ such that $\theta x_1 = \theta X_1 = \theta X_2 = 0$.

Abstract applies here on the first branch, since $zero(X_2)$ can be abstracted, thanks to a version of Proposition 6.3 adapted to local strategies [Fissore et al. 2001]. Indeed, $\mathcal{U}(zero(X_2)) = \{zero(0) \to true, zero(s(x)) \to false\}$, and both rules can be oriented by a LPO $\succ$ with the precedence $zero \succ_{\mathcal{F}} true$ and $zero \succ_{\mathcal{F}} false$. Then we have $TERMIN(LS, zero(X_2))$.

The next constraint formula A is satisfiable with any instantiation θ such that $\theta X_2 = 0$, $\theta X_3 = true$ and $\theta x_1 = i(0)$.

Then, **Narrow** applies on the left branch:



The first constraint formula A is satisfiable by any instantiation θ such that $\theta X_2 = 0$ and $\theta x_1 = i(0)$. The second one is satisfiable by any instantiation θ such that $\theta X_2 = s(0)$ and $\theta x_1 = i(s(0))$. The third one (see below) is satisfiable by any instantiation θ such that $\theta X_3 = zero(i(0))$, $\theta X_2 = i(0)$ and $\theta x_1 = i(i(0))$.

Abstract trivially applies on $g(X_2)$: since X_2 is an abstraction variable, there is no need to abstract it.

The second **Abstract** applies on $f(h(X_2))$, thanks to the previous adaptation of Proposition 6.3 to local strategies. Indeed, $\mathcal{U}(h(X_2)) = \{h(0) \to i(0), h(x) \to s(i(x))\}$, and both rules can be oriented by an LPO with the precedence $h \succ_{\mathcal{F}} i$ and $h \succ_{\mathcal{F}} s$. Then we have $TERMIN(LS, h(X_2))$.

The constraint formula associated to $f(X_4)^{[0]}$ is satisfiable by any instantiation θ such that $\theta X_4 = s(i(s(0)))$, $\theta X_2 = s(0)$ and $\theta x_1 = i(s(0))$.

Then, $f(X_4)^{[0]}$ narrows into $f(X_4)^{[]}$: we are here in the last application case of the rule **Narrow**. Indeed, there is no narrowing possibility satisfying A. The only possible narrowing would use the first rewriting rule and the narrowing substitution $\sigma_A = (X_4 = i(X_5))$.

But then $A \wedge \sigma_A$ would lead to $(x_1{\downarrow} = i(X_2) \wedge zero(X_2){\downarrow} = false \wedge h(X_2){\downarrow} = i(X_5))$. For any θ satisfying $A \wedge \sigma_A$, θ must be such that $\theta h(X_2){\downarrow} = h(\theta X_2{\downarrow}){\downarrow} = i(\theta X_5)$. If $\theta X_2{\downarrow} \neq 0$, then, according to $\mathcal{R}$, $h(\theta X_2{\downarrow}) \to s(i(\theta X_2{\downarrow}))$, where s is a constructor. Then we cannot have $h(\theta X_2{\downarrow}){\downarrow} = i(\theta X_5)$, so θ must be such that $\theta X_2{\downarrow} = 0$. But then $\theta zero(X_2){\downarrow} = true$, which makes $A \wedge \sigma_A$ unsatisfied.

For the third branch, we have:

$$\begin{array}{c}
\bullet \\
ite(X_3, g(X_2), f(h(X_2)))^{[]} \\
A = (x_1{\downarrow} = i(X_2) \wedge \\
zero(X_2){\downarrow} = X_3 \\
\wedge X_3 \neq true \wedge X_3 \neq false) \\
C = (f(x_1) > x_1) \\
| \\
\textbf{Stop} \\
\downarrow \\
\emptyset \\
A = (x_1{\downarrow} = i(X_2) \wedge \\
zero(X_2){\downarrow} = X_3 \wedge \\
X_3 \neq true \wedge X_3 \neq false) \\
C = (f(x_1) > x_1)
\end{array}$$

For the defined symbols $f, zero, h$, the inference rules apply successfully with a common scheme: with an application of **Abstract**, **Narrow**, **Abstract** with no abstraction position, **Narrow** and **Stop**. Therefore $\mathcal{R}$ is LS-terminating.

Let us now give an example of a system that cannot be expressed in the context-sensitive formalism which, for each operator, does not consider a list of specified positions but a set.

Example 8.3. Let $\mathcal{R}$ be the following rewrite system

$$\begin{array}{l}
f(a, g(x)) \to f(a, h(x)) \\
h(x) \to g(x)
\end{array}$$

with the LS-strategy : $LS(f) = [0; 1; 2]$, $LS(h) = [0]$ and $LS(g) = [1]$.

The context-sensitive strategy would allow to permute the reducible arguments of f, so that we could also evaluate terms with $LS(f) = [1; 2; 0]$. We let the reader check that, with this strategy, $\mathcal{R}$ does not terminate.

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

$$f(x_1, x_2)^{[0,1,2]}$$
$$A = \top,\ C = \top$$

Narrow $\sigma=(x_1=a \wedge x_2=g(x_3))$ **Narrow**

$$f(a, h(x_3))^{[0,1,2]}$$
$$A = (x_1 = a \wedge x_2 = g(x_3))$$
$$C = \top$$

Narrow

$$f(a, h(x_3))^{[1,2]}$$
$$A = (x_1 = a \wedge x_2 = g(x_3))$$
$$C = \top$$

Abstract

$$f(a, X_3)^{[]}$$
$$A = (x_1 = a \wedge x_2 = g(x_3) \wedge h(x_3)\downarrow = X_3)$$
$$C = (f(x_1, x_2) > h(x_3))$$

Stop

$$\emptyset$$
$$A = (x_1 = a \wedge x_2 = g(x_3) \wedge h(x_3)\downarrow = X_3)$$
$$C = (f(x_1, x_2) > h(x_3))$$

$$f(x_1, x_2)^{[1,2]}$$
$$A = (x_1 \neq a \vee x_2 \neq g(x_3))$$
$$C = \top$$

Abstract

$$f(X_1, X_2)^{[]}$$
$$A = (x_1\downarrow = X_1 \wedge x_2\downarrow = X_2\ x_1 \neq a \wedge x_2 \neq g(x_3))$$
$$C = (f(x_1, x_2) > x_1, x_2)$$

Stop

$$\emptyset$$
$$A = (x_1\downarrow = X_1 \wedge x_2\downarrow = X_2\ x_1 \neq a \wedge x_2 \neq g(x_3))$$
$$C = (f(x_1, x_2) > x_1, x_2)$$

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

$$h(x_1)^{[0]}$$
$$A = \top,\ C = \top$$

$\sigma=Id$ **Narrow**

$$g(x_1)^{[1]}$$
$$A = \top,\ C = \top$$

Abstract

$$g(X_1)^{[]}$$
$$A = (x_1\downarrow = X_1),\ C = (h(x_1) > x_1)$$

Stop

$$\emptyset$$
$$A = (x_1\downarrow = X_1),\ C = (h(x_1) > x_1)$$

9. IMPLEMENTATION, DISCUSSION, AND COMPARISON WITH RELATED WORKS

Our three original procedures for proving termination of rewriting under the innermost, outermost and local strategies have been implemented in a system named CARIBOO [Fissore et al. 2002a; Fissore 2003]. Although the unified proof framework we propose in this paper is different from the three original procedures, the experiments on our inductive technique allowed by this first implementation [Fissore et al. 2005] have been helpful and are interesting to mention, in particular from the point of view of the generated constraints.

CARIBOO consists of two main parts :

(1) The proof procedures, written in ELAN, which are direct translations of the inference rules. They generate the proof trees, dealing with the ordering and the abstraction constraints. It is worth emphasizing the reflexive aspect of these implementations, developed in a rule-based language, to allow termination of rule-based programs.

(2) A graphical user interface (GUI), written in Java. It provides an edition tool to define specifications of rewrite systems which are then transformed into ELAN specifications used by the proof procedure. It also displays the detailed results of the proof process : which defined symbols have already been treated and, for each of them, the proof tree together with the detail of each node. Trace files can be generated in different formats (HTML, ps, pdf...)

To deal with the generated constraints, the proof processes of CARIBOO can use integrated features, like the computation of usable rules, the use of the subterm ordering or the Lexicographic Path ordering (LPO) to satisfy ordering constraints, and the test of sufficient conditions of Section 4.4 for detecting the unsatisfiability of A.

CARIBOO can also delegate features, as ordering constraint solving, the orientation of the usable rules when the LPO fails, the satisfiability test of A or the termination proof of a term by any other mean than those proposed in Section 4.5, which are implemented in CARIBOO. For the first two cases, delegation can be either proposed to the user, or automatically ensured by the ordering constraint solver C*i*me.

CARIBOO provides several automation modes for dealing with constraints. Dealing with the unsatisfiability of A and using the sufficient conditions given in Section 4.4 allows a complete automatic mode for the innermost and outermost strategies.

Experiments have been made with the CARIBOO implementation [Fissore et al. 2005] on Version 2.0 of the Termination Problem Data Base (TPDB) for termination tools [2]. The TPDB base contains many examples that are universally terminating (i.e. for standard rewriting). We did not focus on them since we were mostly interested in rewriting under specific strategies, but we used them to test our innermost proof process with the automatic mode.

From these experiments, a few remarks can be drawn to enlight the specificities of our approach and comparisons with other works.

[2]Experiments have been made in the beginning of 2006 with the data base available at that time at http://www.lri.fr/∼marche/tpdb/

Ordering constraints. It is interesting to note that thanks to the power of induction, and to the help of usable rules, the generated ordering constraints are often simple, and easily satisfied by the subterm ordering or an LPO. This is the case for 189 of the 229 examples of the data base for standard termination successfully treated by CARIBOO. For the 40 other successful ones, a polynomial ordering computed by C*i*me solves them. The failure cases are due to generated ordering constraints whose resolution requires to compare abstraction variables with standard variables, or which cannot be satisfied by simplification orderings, due to constraints uncompatible with term embedding. In the first case, additional knowledge on normal forms should solve the problem. For the second case, other orderings could be tried.

The automatic mode has been used on the innermost examples of the data base: 27 among 47 have run sucessfully. For all of them, except one requiring C*i*me, ordering constraints were solved by the internal orderings of CARIBOO.

The outermost examples in the TPDB data base are those we gave when participating to the second termination tool competition [Fissore et al. 2003a]. For this strategy, in our initial procedure, the ordering constraints were less important to control the proof process, because they were only used at terminal steps: the abstract step was stopping the process on the current branch of the proof tree.

Abstraction constraints. Since such constraints are rather specific to our approach, there is no existing solver. In full generality, solving such constraints relies on the characterisation of normal forms [Genet 1998; Comon et al. 1997]. In CARIBOO, ad-hoc sufficient conditions for both satisfiability and unsatisfiablity have been implemented. The first ones rely on a syntactic analysis of the signature and the rewrite rules. The second ones are among the integrated features mentioned above.

When the sufficient conditions do not apply, the abstraction constraints are submitted to the user. In the satisfiability mode, a positive answer on satisfiability is required to ensure the next step of the proof process, otherwise it stops. The unsatisfiability test, however, is not blocking: if it succeeds, it enables the process to end. Otherwise, the process goes on and remains sound.

Completeness of definitions. With respect to automation, an additional advantage can be taken from the sufficient completeness property when it is satisfied by the specification. When the rewrite systems are sufficiently complete, every ground term has a constructor normal form, which is often easy to describe and provides intuition for solving abstraction constraints. Moreover, the assumption that every constructor term is minimal for the induction ordering is easy to ensure with a precedence where constructors are minimal, and this yields a straightforward way to solve ordering constraints [Gnaedig and Kirchner 2006]. Several examples of the TPDB data base have been enriched to satisfy sufficient completeness, then leading to a successful proof.

Local versus context-sensitive strategies. There is no example in the TPDB data base for local strategies, but we have considered those for context-sensitive strategies, which are more general. We have realized tests in replacing every context-sensitive strategy $(n_1\ n_2 \ldots n_p)$ by the more specific local strategy $[0\ n_1\ n_2 \ldots n_p]$. This is of course more restrictive, since this encodes only one possible behaviour

of the context-sensitive rewriting strategy. But a local strategy, where the order of reduced positions is relevant, can enable termination while the context sensitive strategy diverges.

On the 37 examples of the data base, 30 were successfully proved terminating under local strategies, among which 7 in a completely automatic way. Moreover, the user interactions were only to authorize skipping a stop rule at some steps of the proof (17 examples), or to solve simple ordering constraints (where an RPO was sufficient) to apply a stop rule (6 examples). All these cases are easily automatable.

Comparison with dependency pairs. Designed from 1996 by T. Arts and J.Giesl, the dependency pair method has proved its efficiency and automatic power for the universal and innermost termination problems. The comparison between this method and our approach is not easy to do. Indeed, several basic ingredients are shared: narrowing, ordering constraints, usable rules are present in both contexts. But while the dependency pairs method is initially based on an analysis of the rules syntax to detect forward closures, our approach was guided by the idea to schematize derivation trees, which allows us to abstract the reduction relation. Except for the innermost case, handling specific rewriting strategies seems more difficult with the dependency pair approach.

Narrowing has been used as the basis of our method, since 1999 [Gnaedig et al. 1999], to schematize rewriting steps of terms, following their possible ground instances. In the dependency pair approach instead [Arts and Giesl 1997], narrowing has been introduced to provide a sufficient condition to detect the dependencies between pairs.

Usable rules have been introduced in the dependency pair approach [Arts and Giesl 1997] for innermost termination. We then have adapted the notion to local strategies [Fissore et al. 2001], and to the outermost strategy [Fissore et al. 2002b], to enrich our inductive proof principle.

Characterization of orderings for proving innermost termination. In [Fernández et al. 2005], the relationship between innermost termination and well-founded orderings is studied. It is shown that orderings suitable for proving innermost termination have to be at least monotonic after each maximal parallel innermost rewriting step. A similar structural requirement can be made on the term ordering needed for our inductive approach: the abstraction process requires in general for the induction ordering to verify $f(x_1, \ldots, x_m) \succ x_i$, for some of the $x_i \in [1..m]$.

General power of the method. As argued above, from a theoretical point of view, our approach has great assets to handle proofs of different rewriting properties: induction is a universal proof mechanism, and the simulation of rewriting by narrowing and abstraction is fine enough to capture refinements of the rewriting relation, like strategies. We could also consider extensions of rewriting like conditional, equational, typed or constrained rewriting, provided the corresponding narrowing relation can be defined. We have shown in [Gnaedig and Kirchner 2006; 2007; Gnaedig 2007] how the method applies to properties other than termination. This is a clear advantage with respect to other existing approaches dedicated to termination.

Our general termination result is complete: it gives a necessary and sufficient condition of termination under the strategy S on ground terms. The undecidability

of termination is reflected in the undecidability of finiteness of the proof trees, in two ways: first the $TERMIN$ predicate is undecidable in general, but it can be checked on specific terms introduced along the proof and under sufficient conditions. Second, satisfiablity (or unsatisfiability) of ordering and abstraction constraints is undecidable and here also we have given several sufficient conditions, in the general case as well as in the specific case of each considered strategy.

All complete results about termination lead anyway to deal with sufficient conditions in practice. Comparing different approaches then amounts at comparing these sufficient conditions, which is very hard to do. They can always be improved, either by finding new sufficient conditions, or by improving the efficiency of their implementation, and this comparison goes beyond the limits of this paper.

In the CARIBOO prototype, the implementation effort was put in priority on experiments and validation of our approach to handle the termination problem for different strategies. The framework proposed here should now lead to a more concise and structurated code, with shared procedures for the different strategies. We also expect to improve automation and efficiency.

10. CONCLUSION

The termination proof method presented in this paper is based on the simple ideas of schematizing and observing the derivation trees of ground terms, supposing that terms smaller that the starting terms for an induction ordering are terminating.

By using the induction ordering, we skip normalizing subderivations and we stop derivations as soon as termination is ensured by induction.

The framework we have given unifies the three different procedures we had previously proposed for proving termination of rewriting respectively under the innermost, the outermost and the local strategies. Analyzing and extracting their common features, and identifying the specific ones led us to a more general and simple process expressed with three inference steps. Each of them enlights one basic concept of the proposed method. Induction on a noetherian ordering is used to stop the process. Abstraction also takes advantage of induction for dealing in one step with many rewriting steps leading to a specific normal form. Narrowing represents all potential applications of rewrite rules at a given position. As a consequence of this unified presentation, proofs of the results have also been factorized and simplified.

The current work also makes clear that the specificities related to each strategy are localized in the control, either in constraints or conditions of the inference rules, or in the application strategies of these inference steps.

Another characteristic of this work is that constraints are heavily used on one hand to gather conditions that the induction ordering must satisfy, on the other hand to represent the sets of ground instances of terms considered in the proof. The power of deduction with constraints [Kirchner et al. 1990] is once more illustrated in this proof process where the construction of the ordering for instance may be delayed until the end of the process.

The techniques presented here look promising in different directions. They already have been applied successfully to weak termination proofs in [Fissore et al. 2004; Gnaedig and Kirchner 2007], where the analysis of proof trees provides a

constructive algorithm to reach a normal form. Moreover, since the proof process can be expressed with respect to any rewriting relation, it can easily be extended to prove ground termination of conditional, equational and typed rewriting, with an adequate definition of narrowing.

The approach can also be applied to other properties than termination. For instance, it has been recently adapted to the sufficient completeness problem [Gnaedig and Kirchner 2006] and to termination of probabilistic rewriting [Gnaedig 2007]. We expect it to be interesting too to tackle other properties like ground confluence, or other reduction frameworks like transition systems.

ACKNOWLEDGMENTS

We sincerely thank Olivier Fissore for fruitful exchanges on previous works on the topic, the Protheo group for providing support to our ideas, the referees for their pertinent remarks and suggestions and Nachum Dershowitz for his interest and friendly advices.

REFERENCES

Alarcón, B., Gutiérrez, R., and Lucas, S. 2006. Context-sensitive dependency pairs. In *Proceedings of the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science*. Lecture Notes in Computer Science, vol. 4337. Springer, 297–308.

Alarcón, B. and Lucas, S. 2007. Termination of innermost context-sensitive rewriting using dependency pairs. In *Proceedings of the 6th International Symposium on Frontiers of Combining Systems*. lnai. Springer-Verlag.

Alpuente, M., Escobar, S., and Lucas, S. 2004. Correct and complete (positive) strategy annotations for OBJ. In *Proceedings of the 5th International Workshop on Rewriting Logic and its Applications*. Elecronic Notes In Theoretical Computer Science, vol. 71. esp, 70–89.

Arts, T. and Giesl, J. 1997. Proving innermost normalisation automatically. In *Proceedings 8th Conference on Rewriting Techniques and Applications, Sitges (Spain)*. Lecture Notes in Computer Science, vol. 1232. Springer-Verlag, 157–171.

Arts, T. and Giesl, J. 2000. Termination of term rewriting using dependency pairs. *Theor. Comput. Sci. 236*, 133–178.

Baader, F. and Nipkow, T. 1998. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA.

Barendsen, E., Bethke, I., Heering, J., Kennaway, R., Klint, P., van Oostrom, V., van Raamsdonk, F., de Vries, F.-J., and Zantema, H. 2003. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press.

Ben Cherifa, A. and Lescanne, P. 1987. Termination of rewriting systems by polynomial interpretations and its implementation. *Sci. Comput. Program. 9,* 2 (Oct.), 137–160.

Borovanský, P., Kirchner, C., Kirchner, H., Moreau, P.-E., and Ringeissen, C. 1998. An Overview of ELAN. In *Proceedings of the 2nd International Workshop on Rewriting Logic and its Applications*, C. Kirchner and H. Kirchner, Eds. Electronic Notes in Theoretical Computer Science. Elsevier Science Publishers B. V. (North-Holland), Pont-à-Mousson (France).

Borralleras, C., Ferreira, M., and Rubio, A. 2000. Complete monotonic semantic path orderings. In *Proceedings of the 17th International Conference on Automated Deduction*. Lecture Notes in Computer Science, vol. 1831. Springer-Verlag, Pittsburgh, PA, USA, 346–364.

Brand, M., Deursen, A., Heering, J., Jong, H., Jonge, M., Kuipers, T., Klint, P., Moonen, L., Olivier, P., Scheerder, J., Vinju, J., Visser, E., and Visser, J. 2001. The ASF+SDF Meta-Environment: a Component-Based Language Development Environment. In *Compiler Construction (CC '01)*, R. Wilhelm, Ed. Lecture Notes in Computer Science, vol. 2027. Springer, 365–370.

Cirstea, H. and Kirchner, C. 2001. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics 9*, 427–498.

Cirstea, H., Kirchner, C., Liquori, L., and Wack, B. 2003. Rewrite strategies in the rewriting calculus. In *Electronic Notes in Theoretical Computer Science*, B. Gramlich and S. Lucas, Eds. Vol. 86. Elsevier.

Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., and Talcott, C. 2003. The Maude 2.0 system. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, R. Nieuwenhuis, Ed. Lecture Notes in Computer Science, vol. 2706. Springer, 76–87.

Comon, H. 1991. Disunification: a survey. In *Computational Logic. Essays in honor of Alan Robinson*, J.-L. Lassez and G. Plotkin, Eds. The MIT press, Cambridge (MA, USA), Chapter 9, 322–359.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. 1997. Tree automata techniques and applications. Release October, 1rst 2002.

Dershowitz, N. 1982a. Orderings for term rewriting systems. *Theor. Comput. Sci. 17*, 279–301.

Dershowitz, N. 1982b. Orderings for term-rewriting systems. *Theor. Comput. Sci. 17*, 279–301.

Dershowitz, N. and Hoot, C. 1995. Natural termination. *Theor. Comput. Sci. 142(2)*, 179–207.

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

Dershowitz, N. and Plaisted, D. 2001. Rewriting. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, Eds. Vol. I. Elsevier Science, Chapter 9, 535–610.

Eker, S. 1998. Term rewriting with operator evaluation strategies. In *Proceedings of the 2nd International Workshop on Rewriting Logic and its Applications*, C. Kirchner and H. Kirchner, Eds. Pont-à-Mousson (France).

Fernández, M.-L., Godoy, G., and Rubio, A. 2005. Orderings for innermost termination. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications*, J. Giesl, Ed. Lecture Notes in Computer Science, vol. 3467. Springer-Verlag, Nara, Japan, 17–31.

Fissore, O. 2003. Terminaison de la réécriture sous stratégies. Ph.D. thesis, Université Henri Poincaré-Nancy I, France.

Fissore, O., Gnaedig, I., and Kirchner, H. 2001. Termination of rewriting with local strategies. In *Selected papers of the 4th International Workshop on Strategies in Automated Deduction*, M. P. Bonacina and B. Gramlich, Eds. Electronic Notes in Theoretical Computer Science, vol. 58. Elsevier Science Publishers B. V. (North-Holland).

Fissore, O., Gnaedig, I., and Kirchner, H. 2002a. CARIBOO : An induction based proof tool for termination with strategies. In *Proceedings of the 4th International Conference on Principles and Practice of Declarative Programming*. ACM Press, Pittsburgh, USA, 62–73.

Fissore, O., Gnaedig, I., and Kirchner, H. 2002b. Outermost ground termination. In *Proceedings of the 4th International Workshop on Rewriting Logic and Its Applications*. Electronic Notes in Theoretical Computer Science, vol. 71. Elsevier Science Publishers B. V. (North-Holland), Pisa, Italy.

Fissore, O., Gnaedig, I., and Kirchner, H. 2002c. Outermost ground termination - Extended version. Tech. Rep. A02-R-493, LORIA, Nancy (France). December.

Fissore, O., Gnaedig, I., and Kirchner, H. 2003a. CARIBOO: A multi-strategy termination proof tool based on induction. In *Proceedings of the 6th International Workshop on Termination 2003*, A. Rubio, Ed. Valencia (Spain), 77–79.

Fissore, O., Gnaedig, I., and Kirchner, H. 2003b. Simplification and termination of strategies in rule-based languages. In *Proceedings of the 5th International Conference on Principles and Practice of Declarative Programming*. ACM Press, Uppsala, Sweden, 124–135.

Fissore, O., Gnaedig, I., and Kirchner, H. 2004. A proof of weak termination providing the right way to terminate. In *Proceedings of the 1st International Colloquium on THEORETICAL ASPECTS OF COMPUTING*. Lecture Notes in Computer Science, vol. 3407. Springer-Verlag, Guiyang, China, 356–371.

Fissore, O., Gnaedig, I., Kirchner, H., and Moussa, L. 2005. Cariboo, a termination proof tool for rewriting-based programming languages with strategies, Version 1.1. Free GPL Licence, APP registration IDDN.FR.001.170013.001.S.P.2005.000.10600. `Available at http://cariboo.loria.fr/`.

Futatsugi, K. and Nakagawa, A. 1997. An overview of the 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*.

Genet, T. 1998. Decidable approximations of sets of descendants and sets of normal forms. In *Proceedings of the 9th Conference on Rewriting Techniques and Applications, Tsukuba (Japan)*. Lecture Notes in Computer Science, vol. 1379. Springer-Verlag, 151–165.

Giesl, J. and Middeldorp, A. 1999. Transforming Context-Sensitive Rewrite Systems. In *Proceedings of the 10th International Conference on Rewriting Techniques and Applications*. Lecture Notes in Computer Science, vol. 1631. Springer-Verlag, Trento (Italy), 271–285.

Giesl, J. and Middeldorp, A. 2003. Innermost termination of context-sensitive rewriting. In *Proceedings of the 6th International Conference on Developments in Language Theory (DLT 2002)*. Lecture Notes in Computer Science, vol. 2450. Springer-Verlag, Kyoto, Japan, 231–244.

Giesl, J., Swiderski, S., Schneider-Kamp, P., and Thiemann, R. 2006. Automated Termination Analysis for Haskell: From term rewriting to programming languages. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*. 297–312.

Giesl, J., Thiemann, R., and Schneider-Kamp, P. 2004. The dependency pair framework: Combining techniques for automated termination proofs. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Lecture Notes in Artificial Intelligence, vol. 3452. Springer, 301–331.

Giesl, J., Thiemann, R., Schneider-Kamp, P., and Falke, S. 2003. Improving dependency pairs. In *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Lecture Notes in Artificial Intelligence, vol. 2850. Springer-Verlag, Almaty, Kazakhstan, 165–179.

Gnaedig, I. 2007. Induction for Positive Almost Sure Termination. In *Proceedings of the 9th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming*. ACM Press, Wroclaw, Poland, 167–177.

Gnaedig, I. and Kirchner, H. 2006. Computing Constructor Forms with Non Terminating Rewrite Programs. In *Proceedings of the 8th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming*. ACM Press, Venice, Italy, 121–132.

Gnaedig, I. and Kirchner, H. 2007. Narrowing, abstraction and constraints for proving properties of reduction relations. In *Rewriting, Computation and Proof. Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, H. Comon-Lundh, C. Kichner, and H. Kirchner, Eds. Lecture Notes in Computer Science, vol. 4600. Springer-Verlag, 44–67.

Gnaedig, I., Kirchner, H., and Genet, T. 1999. Induction for Termination. Tech. Rep. 99.R.338, LORIA, Nancy (France). December.

Goguen, J., Winkler, T., Meseguer, J., Futatsugi, K., and Jouannaud, J. 1992. Introducing OBJ3. Tech. rep., Computer Science Laboratory, SRI International. Mar.

Goubault-Larreck, J. 2001. Well-founded recursive relations. In *Proceedings of the 15th International Workshop in Computer Science Logic (CSL'2001)*. Lecture Notes in Computer Science, vol. 2142. Springer-Verlag, Paris.

Gramlich, B. 1995. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae 24*, 3–23.

Gramlich, B. 1996. On proving termination by innermost termination. In *Proceedings 7th Conference on Rewriting Techniques and Applications, New Brunswick (New Jersey, USA)*, H. Ganzinger, Ed. Lecture Notes in Computer Science, vol. 1103. Springer-Verlag, 93–107.

Kamin, S. and Lévy, J.-J. 1982. Attempts for generalizing the recursive path ordering. *Inria, Rocquencourt*.

Kirchner, C. 2005. Strategic rewriting. *Electr. Notes Theor. Comput. Sci. 124,* 2, 3–9.

Kirchner, C., Kirchner, H., and Rusinowitch, M. 1990. Deduction with symbolic constraints. *Revue d'Intelligence Artificielle 4,* 3, 9–52. Special issue on Automatic Deduction.

Krishna Rao, M. 2000. Some characteristics of strong normalization. *Theor. Comput. Sci. 239*, 141–164.

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

Lankford, D. S. 1979. On proving term rewriting systems are noetherian. Tech. rep., Louisiana Tech. University, Mathematics Dept., Ruston LA.

Lucas, S. 1996. Termination of context-sensitive rewriting by rewriting. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 1099. Springer-Verlag, 122–133.

Lucas, S. 2001a. Termination of on-demand rewriting and termination of OBJ programs. In *Proceedings of the 3rd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'01*, H. Sondergaard, Ed. ACM Press, New York, Firenze, Italy, 82–93.

Lucas, S. 2001b. Termination of rewriting with strategy annotations. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, A. Voronkov and R. Nieuwenhuis, Eds. Lecture Notes in Artificial Intelligence, vol. 2250. Springer-Verlag, Berlin, La Habana, Cuba, 669–684.

Lucas, S. 2002. Context-sensitive rewriting strategies. *Inf. Comput. 178,* 1, 294–343.

Middeldorp, A. and Hamoen, E. 1994. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computation 5,* 3 & 4, 213–253.

Moreau, P.-E., Ringeissen, C., and Vittek, M. 2003. A Pattern Matching Compiler for Multiple Target Languages. In *Proceedings of the 12th Conference on Compiler Construction, Warsaw (Poland)*, G. Hedin, Ed. Lecture Notes in Computer Science, vol. 2622. Springer, 61–76.

Nakamura, M. and Ogata, K. 2000. The evaluation strategy for head normal form with and without on-demand flags. In *Proceedings of the 3rd International Workshop on Rewriting Logic and its Applications, WRLA'2000*, K. Futatsugi, Ed. Electronic Notes in Theoretical Computer Science, Kanazawa City Cultural Halt, Kanazawa, Japan, 211–227.

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

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

Plaisted, D. 1978. Well-founded orderings for proving termination of systems of rewrite rules. Tech. Rep. R-78-932, Department of Computer Science, Univesity of Illinois at Urbana Champaign. July.

van Oostrom, V. and de Vrijer, R. 2003. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press, Chapter 9. Strategies.

Visser, E. 2001. Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description for Stratego 0.5. In *Proceedings of the 12th International Conference on Rewriting Techniques and Applications*, A. Middeldorp, Ed. Lecture Notes in Computer Science, vol. 2051. Springer, 357–361.

Visser, E. 2004. Program transformation with Stratego/XT: Rules, strategies, tools, and systems in StrategoXT-0.9. In *Domain-Specific Program Generation*, C. Lengauer et al., Eds. Lecture Notes in Computer Science, vol. 3016. Spinger-Verlag, 216–238.

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

APPENDIX

A. THE LIFTING LEMMA

The lifting lemma for standard narrowing [Middeldorp and Hamoen 1994] can be locally adapted to S-rewriting (rewriting under the innermost, outermost or local strategies) with non-normalized substitutions provided they fulfill some constraints on the positions of rewriting. To do so, we need the following two propositions (the first one is obvious).

PROPOSITION A.1. *Let $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and σ be a substitution of $\mathcal{T}(\mathcal{F},\mathcal{X})$. Then $Var(\sigma t) = (Var(t) - Dom(\sigma)) \cup Ran(\sigma_{Var(t)})$.*

PROPOSITION A.2. *Suppose we have substitutions σ, μ, ν and sets A, B of variables such that $(B - Dom(\sigma)) \cup Ran(\sigma) \subseteq A$. If $\mu = \nu[A]$ then $\mu\sigma = \nu\sigma[B]$.*

PROOF. Let us consider $(\mu\sigma)_B$, which can be divided as follows: $(\mu\sigma)_B = (\mu\sigma)_{B\cap Dom(\sigma)} \cup (\mu\sigma)_{B-Dom(\sigma)}$.
For $x \in B \cap Dom(\sigma)$, we have $\mathcal{V}ar(\sigma x) \subseteq Ran(\sigma)$, and then $(\mu\sigma)x = \mu(\sigma x) = \mu_{Ran(\sigma)}(\sigma x) = (\mu_{Ran(\sigma)}\sigma)x$. Therefore $(\mu\sigma)_{B\cap Dom(\sigma)} = (\mu_{Ran(\sigma)}\sigma)_{B\cap Dom(\sigma)}$.
For $x \in B - Dom(\sigma)$, we have $\sigma x = x$, and then $(\mu\sigma)x = \mu(\sigma x) = \mu x$. Therefore we have $(\mu\sigma)_{B-Dom(\sigma)} = \mu_{B-Dom(\sigma)}$. Henceforth we get $(\mu\sigma)_B = (\mu_{Ran(\sigma)}\sigma)_{B\cap Dom(\sigma)} \cup \mu_{B-Dom(\sigma)}$.
By a similar reasoning, we get $(\nu\sigma)_B = (\nu_{Ran(\sigma)}\sigma)_{B\cap Dom(\sigma)} \cup \nu_{B-Dom(\sigma)}$.
By hypothesis, we have $Ran(\sigma) \subseteq A$ and $\mu = \nu[A]$. Then $\mu_{Ran(\sigma)} = \nu_{Ran(\sigma)}$. Likewise, since $B - Dom(\sigma) \subseteq A$, we have $\mu_{B-Dom(\sigma)} = \nu_{B-Dom(\sigma)}$.
Then we have $(\mu\sigma)_B = (\mu_{Ran(\sigma)}\sigma)_{B\cap Dom(\sigma)} \cup \mu_{B-Dom(\sigma)} = (\nu_{Ran(\sigma)}\sigma)_{B\cap Dom(\sigma)} \cup \nu_{B-Dom(\sigma)} = (\nu\sigma)_B$. Therefore $(\mu\sigma) = (\nu\sigma)[B]$. $\square$

LEMMA (S-LIFTING LEMMA). *Let $\mathcal{R}$ be a rewrite system. Let $s \in \mathcal{T}(\mathcal{F},\mathcal{X})$, α a ground substitution such that αs is S-reducible at a non variable position p of s, and $\mathcal{Y} \subseteq \mathcal{X}$ a set of variables such that $Var(s) \cup Dom(\alpha) \subseteq \mathcal{Y}$. If $\alpha s \rightarrow_S^{p,l\rightarrow r} t'$, then there exist a term $s' \in \mathcal{T}(\mathcal{F},\mathcal{X})$ and substitutions $\beta, \sigma = \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j}$ such that:*

1. $s \rightsquigarrow_S^{p,l\rightarrow r,\sigma} s'$,
2. $\beta s' = t'$,
3. $\beta\sigma_0 = \alpha[\mathcal{Y} \cup Var(l)]$
4. β satisfies $\bigwedge_{j\in[1..k]} \overline{\sigma_j}$.

where σ_0 is the most general unifier of $s|_p$ and l and $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 s|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all position p' which are S-better positions than p in s.

PROOF. In the following, we assume that $\mathcal{Y} \cap \mathcal{V}ar(l) = \emptyset$ for every $l \rightarrow r \in \mathcal{R}$.
If $\alpha s \rightarrow_S^{p,l\rightarrow r} t'$, then there exists a substitution τ such that $Dom(\tau) \subseteq \mathcal{V}ar(l)$ and $(\alpha s)|_p = \tau l$. Moreover, since p is a non variable position of s, we have $(\alpha s)|_p = \alpha(s|_p)$. Denoting $\mu = \alpha\tau$, we have:

$$\begin{array}{lll} \mu(s|_p) & = \alpha(s|_p) & \text{for } Dom(\tau) \subseteq \mathcal{V}ar(l) \text{ and } \mathcal{V}ar(l) \cap \mathcal{V}ar(s) = \emptyset \\ & = \tau l & \text{by definition of } \tau \\ & = \mu l & \text{for } Dom(\alpha) \subseteq \mathcal{Y} \text{ and } \mathcal{Y} \cap \mathcal{V}ar(l) = \emptyset, \end{array}$$

and therefore $s|_p$ and l are unifiable. Let us note σ_0 the most general unifier of $s|_p$ and l, and $s' = \sigma_0(s[r]_p)$.

Since σ_0 is more general than μ, there exists a substitution ρ such that $\rho\sigma_0 = \mu[\mathcal{Y} \cup Var(l)]$. Let $\mathcal{Y}_1 = (\mathcal{Y} - Dom(\sigma_0)) \cup Ran(\sigma_0)$. We define $\beta = \rho_{\mathcal{Y}_1}$. Clearly $Dom(\beta) \subseteq \mathcal{Y}_1$.
We now show that $\mathcal{V}ar(s') \subseteq \mathcal{Y}_1$, by the following reasoning:

—since $s' = \sigma_0(s[r]_p)$, we have $\mathcal{V}ar(s') = \mathcal{V}ar(\sigma_0(s[r]_p))$;

—the rule $l \to r$ is such that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$, therefore we have $\mathcal{V}ar(\sigma_0(s[r]_p)) \subseteq \mathcal{V}ar(\sigma_0(s[l]_p))$, and then, thanks to the previous point, $\mathcal{V}ar(s') \subseteq \mathcal{V}ar(\sigma_0(s[l]_p))$;

—since $\sigma_0(s[l]_p) = \sigma_0 s[\sigma_0 l]_p$ and since σ_0 unifies l and $s|_p$, we get $\sigma_0(s[l]_p) = (\sigma_0 s)[\sigma_0(s|_p)]_p = \sigma_0 s[s|_p]_p = \sigma_0 s$ and, thanks to the previous point: $\mathcal{V}ar(s') \subseteq \mathcal{V}ar(\sigma_0 s)$;

—according to Proposition A.1, we have $\mathcal{V}ar(\sigma_0(s)) = (\mathcal{V}ar(s) - Dom(\sigma_0)) \cup Ran(\sigma_{0\mathcal{V}ar(s)})$; by hypothesis, $\mathcal{V}ar(s) \subseteq \mathcal{Y}$. Moreover, since $Ran(\sigma_{0\mathcal{V}ar(s)}) \subseteq Ran(\sigma_0)$, we have
$\mathcal{V}ar(\sigma_0(s)) \subseteq (\mathcal{Y} - Dom(\sigma_0)) \cup Ran(\sigma_0)$, that is $\mathcal{V}ar(\sigma_0 s) \subseteq \mathcal{Y}_1$. Therefore, with the previous point, we get $Var(s') \subseteq \mathcal{Y}_1$.

From $Dom(\beta) \subseteq \mathcal{Y}_1$ and $Var(s') \subseteq \mathcal{Y}_1$, we infer $Dom(\beta) \cup Var(s') \subseteq \mathcal{Y}_1$.

Let us now prove that $\beta s' = t'$.
Since $\beta = \rho_{\mathcal{Y}_1}$, we have $\beta = \rho[\mathcal{Y}_1]$. Since $Var(s') \subseteq \mathcal{Y}_1$, we get $\beta s' = \rho s'$. Since $s' = \sigma_0(s[r]_p)$, we have $\rho s' = \rho\sigma_0(s[r]_p) = \mu(s[r]_p) = \mu s[\mu r]_p$. Then $\beta s' = \mu s[\mu r]_p$.
We have $Dom(\tau) \subseteq \mathcal{V}ar(l)$ and $\mathcal{Y} \cap \mathcal{V}ar(l) = \emptyset$, then we have $\mathcal{Y} \cap Dom(\tau) = \emptyset$. Therefore, from $\mu = \alpha\tau[\mathcal{Y} \cup Var(l)]$, we get $\mu = \alpha[\mathcal{Y}]$. Since $\mathcal{V}ar(s) \subseteq \mathcal{Y}$, we get $\mu s = \alpha s$.
Likewise, by hypothesis we have $Dom(\alpha) \subseteq \mathcal{Y}$, $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ and $\mathcal{Y} \cap \mathcal{V}ar(l) = \emptyset$, then we get $Var(r) \cap Dom(\alpha) = \emptyset$, and then we have $\mu = \tau[Var(r)]$, and therefore $\mu r = \tau r$.
From $\mu s = \alpha s$ and $\mu r = \tau r$ we get $\mu s[\mu r]_p = \alpha s[\tau r]_p$. Since, by hypothesis, $\alpha s \to^p t'$, with $\tau l = (\alpha s)|_p$, then $\alpha s[\tau r]_p = t'$. Finally, as $\beta s' = \mu s[\mu r]_p$, we get $\beta s' = t'$ (2).

Next let us prove that $\beta\sigma_0 = \alpha[\mathcal{Y}]$. Reminding that $\mathcal{Y}_1 = (\mathcal{Y} - Dom(\sigma_0)) \cup Ran(\sigma_0)$, Proposition A.2 (with the notations A for $\mathcal{Y}_1$, B for $\mathcal{Y}$, μ for β, ν for ρ and σ for σ_0) yields $\beta\sigma_0 = \rho\sigma_0[\mathcal{Y}]$. We already noticed that $\mu = \alpha[\mathcal{Y}]$. Linking these two equalities via the equation $\rho\sigma_0 = \mu$ yields $\beta\sigma_0 = \alpha[\mathcal{Y}]$ (3).

Let us now suppose that there exist a rule $l' \to r' \in \mathcal{R}$, a position p' S-better than p and a substitution σ_i such that $\sigma_i(\sigma_0(s|_{p'})) = \sigma_i l'$.

Let us now suppose that β does not satisfy $\bigwedge_{j\in[1..k]} \overline{\sigma_j}$. There exists $i \in [1..k]$ such that β satisfies $\sigma_i = \bigwedge_{i_l\in[1..n]}(x_{i_l} = u_{i_l})$. So β is such that $\bigwedge_{i_l\in[1..n]}(\beta x_{i_l} = \beta u_{i_l})$.

Thus, on $Dom(\beta) \cap Dom(\sigma_i) \subseteq \{x_{i_l}, i_l \in [1..n]\}$, we have $(\beta x_{i_l} = \beta u_{i_l})$, so $\beta\sigma_i = \beta$. Moreover, as β is a ground substitution, $\sigma_i\beta = \beta$. Thus, $\beta\sigma_i = \sigma_i\beta$.

On $Dom(\beta) \cup Dom(\sigma_i) - (Dom(\beta) \cap Dom(\sigma_i))$, either $\beta = Id$, or $\sigma_i = Id$, so $\beta\sigma_i = \sigma_i\beta$.

As a consequence, $\alpha(s) = \sigma_i\alpha(s) = \sigma_i\beta\sigma_0(s) = \beta\sigma_i\sigma_0(s)$ is reducible at position p' with the rule l', which is impossible by definition of reducibility of $\alpha(s)$ at position p under the strategy S. So the ground substitution β satisfies $\bigwedge_{i\in[1..k]} \overline{\sigma_i}$ for all

most general unifiers σ_i of $\sigma_0 s$ and a left-hand side of a rule of $\mathcal{R}$ at S-better positions of p (4).

Therefore, denoting $\sigma = \sigma_0 \wedge \bigwedge_{i\in[1..k]} \overline{\sigma_i}$, from the beginning of the proof, we get $s \rightsquigarrow_S^{p,l\to r,\sigma} s'$, and then the point (1) of the current lemma holds. □

B. PROOF OF THE TERMINATION RESULT

Let us remind that $SUCCESS(g, \succ)$ means that the application of $Strat{-}Rules(S)$ on $(\{g(x_1, \ldots, x_m)\}, \top, \top)$ gives a finite proof tree, whose sets C of ordering constraints are satisfied by a same ordering $\succ$, and whose leaves are either nodes of the form $(\emptyset, A, C)$ or nodes whose set of constraints A is unsatisfiable.

THEOREM 5.1. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols containing at least one constructor constant. Every term of $\mathcal{T}(\mathcal{F})$ terminates under the strategy S iff there exists a noetherian ordering $\succ$ such that for each symbol $g \in \mathcal{D}$, we have $SUCCESS(g, \succ)$.*

PROOF. Let us suppose that every ground term is S-terminating and show that the construction of the proof trees always terminate. Let $f(x_1, \ldots, x_m), f \in \mathcal{D}$ any initial pattern of a proof tree.

If $S = Innermost$, **Abstract** applies to give $f(X_1, \ldots, X_m), X_1, \ldots, X_m \in \mathcal{X}_A$. Indeed, by hypothesis, we have $TERMIN(Innermost, x_i)$. Then **Stop** applies, because we have $TERMIN(Innermost, f(X_1, \ldots, X_m))$.

If $S = LS$, the reasoning is the same, except that only some subterms x_i of $f(x_1, \ldots, x_m)$ are abstracted, according to the local strategy of f.

If $S = Outermost$, **Narrow** applies on $f(x_1, \ldots, x_m)$, to give some term u. In a similar way than in the previous cases, we have $TERMIN(Outermost, u)$, and **Stop** applies.

So any proof tree is finite, and $SUCCESS(f, \succ)$ for every $f \in \mathcal{D}$, with any noetherian ordering $\succ$.

For the converse part, we use an emptiness lemma, an abstraction lemma, a narrowing lemma, and a stopping lemma, which are given after this main proof.

We prove by induction on $\mathcal{T}(\mathcal{F})$ that any ground instance $\theta f(x_1, \ldots, x_m)$ of any term $f(x_1, \ldots, x_m) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ with $f \in \mathcal{F}$, S-terminates. The induction ordering is constrained along the proof. At the beginning, it has at least to be noetherian. Such an ordering always exists on $\mathcal{T}(\mathcal{F})$ (for instance the embedding relation). Let us denote it $\succ$.

If f is a defined symbol, let us denote it g and prove that $g(\theta x_1, \ldots, \theta x_m)$ S-terminates for any θ satisfying $A = \top$. Note that g may be a reducible constant. Let us denote $g(x_1, \ldots, x_m)$ by t_{ref} in the sequel of the proof.

To each node of the proof tree of g, characterized by a current term t and the set of constraints A, we associate the set of ground terms $G = \{\alpha t \mid \alpha \text{ satisfies } A\}$, that is the set of ground instances of t schematized by A. When t is a reducible constant, the set of ground instances is reduced to t itself.

Inference rule **Abstract** (resp. **Narrow**) transforms $(\{t\}, A, C)$ into $(\{t'\}, A', C')$ to which is associated $G' = \{\beta t' \mid \beta \text{ satisfies } A'\}$ (resp. into $(\{t'_i\}, A'_i), i \in [1..l]$ to which are associated $G' = \{\beta_i t'_i \mid \beta_i \text{ satisfies } A'_i\}$).

By abstraction (resp. narrowing) Lemma, applying **Abstract** (resp. **Narrow**), for each reducible αt in G, there exists a $\beta t'$ (resp. $\beta_i t'_i$) in G' and such that S-termination of $\beta t'$ (resp. of the $\beta_i t'_i$) implies S-termination of αt.

When the inference rule **Stop** applies on $(\{t\}, A, C)$:

—either A is satisfiable, in which case, by stopping lemma, every term of $G = \{\alpha t \mid \alpha \text{ satisfies } A\}$ is S-terminating,

—or A is unsatisfiable. In this case, G is empty. By emptiness lemma, all previous nodes on the branch correspond to empty sets G_i, until an ancestor node $(\{t_p\}, A_p, C_p)$, where A_p is satisfiable. Then every term αt of G_p is irreducible, otherwise, by Abstraction and Narrowing lemmas, G_{p+1} would not be empty.

Therfore, S-termination is ensured for all terms in all sets G of the proof tree. As the process is initialized with $\{t_{ref}\}$ and a constraint problem satisfiable by any ground substitution, we get that $g(\theta x_1, \ldots, \theta x_m)$ is S-terminating, for any $t_{ref} = g(x_1, \ldots, x_m)$ with $g \in \mathcal{D}$, and any ground instance θ.

If f is a constructor, either it is a constant, which is irreducible, by hypothesis, and then S-terminating, or we consider the pattern $f(x_1, \ldots, x_m)$. The proof then works like in the case of defined symbols, but with only two proof steps, namely *abstract* and *stop*. Indeed, $f(x_1, \ldots, x_m)$ always abstracts into $f(x_1, \ldots, x_m)[X_{p_1}]_{p_1} \ldots [X_{p_k}]_{p_k}$, where $\{p_1, \ldots, p_k\}$ are the abstraction positions allowed by the strategy ($[1..m]$ if $S = Innermost$ or $S = Outermost$, $\{p_1, \ldots, p_k\} = LS(f)$ if $S = LS$). Then *stop* applies because $f(x_1, \ldots, x_m)[X_{p_1}]_{p_1} \ldots [X_{p_k}]_{p_k}$ is not narrowable and all its variables at positions allowed by the strategy S are in $\mathcal{X}_A$.

We thus get that any ground term, written as a ground instance of a pattern $f(x_1, \ldots, x_m)$ with $f \in \mathcal{F}$, is S-terminating. □

Lemma (Emptiness lemma). *Let $(\{t\}, A, C)$ be a node of any proof tree, giving $(\{t'\}, A', C')$ by application of* **Abstract** *or* **Narrow**. *If A is unsatisfiable, then so is A'.*

Proof. If **Abstract** is applied, then if A is unsatisfiable, $A' = A \wedge t|_{p_1}\downarrow = X_{p_1} \ldots \wedge t|_{p_k}\downarrow = X_{p_k}$ is also unsatisfiable.

If **Narrow** is applied, then if A is unsatisfiable (which never happens for local strategies), $A' = A \wedge \sigma$ in the innermost case, and $A' = R(t) \wedge A \wedge \sigma$ in the outermost case are also unsatisfiable. □

Lemma (Abstraction lemma). *Let $(\{t\}, A, C)$ be a node of any proof tree, giving the node $(\{t' = t[X_j]_{j \in \{p_1, \ldots, p_k\}}\}, A', C')$ by application of* **Abstract**.

For any ground substitution α satisfying A, if αt is reducible, there exists β such that S-termination of $\beta t'$ implies S-termination of αt. Moreover, β satisfies A'.

Proof. We prove that $\alpha t \xrightarrow{*}_S \beta t'$, where $\beta = \alpha \cup \bigcup_{j \in \{p_1, \ldots, p_k\}} X_j = \alpha t|_j\downarrow$.

First, whatever the strategy S, the abstraction positions in t are chosen so that the $\alpha t|_j$ can be supposed terminating under S. Indeed, each term $t|_j$ is such that:

—either $TERMIN(S, t|_j)$ is true, and then by definition of the predicate $TERMIN$, $\alpha t|_j$ S-terminates;

—or $t_{ref} > t|_j$ is satisfiable by $\succ$, and then, by induction hypothesis, $\alpha t|_j$ S-terminates.

So the $\alpha t|_j\downarrow exist$.

Then, let us consider the different choices of abstraction positions w.r.t the strategy S:

—either $S = Innermost$, and whatever the positions $p_1, \ldots, p_k$ in the term t, we have $\alpha t \xrightarrow{*}_{Innermost} \alpha t[\alpha t|_{p_1}\downarrow]_{p_1} \ldots [\alpha t|_{p_k}\downarrow]_{p_k} = \beta t'$;

—either $S = Outermost$ and t is abstracted at positions $p_1, \ldots, p_k$ if $t[X_j]_{j\in\{p_1,\ldots,p_k\}}$ is not outermost narrowable at prefix positions of $p_1, \ldots, p_k$, which warrants that the only redex positions of αt are suffixes of the j, and then that $\alpha t \xrightarrow{*}_{Outermost} \alpha t[\alpha t|_{p_1}\downarrow]_{p_1} \ldots [\alpha t|_{p_k}\downarrow]_{p_k} = \beta t'$;

—or $S = LS$ and $top(t) = f$ with $LS(f) = [p_1, \ldots, p_n]$. The term t is abstracted at positions $p_1, \ldots, p_{k-1}$, if $\exists k \in [2..n] : p_1, \ldots, p_{k-1} \neq 0, p_k = 0$, or at positions $p_1, \ldots, p_n$ if $p_1, \ldots, p_n \neq 0$. According to the definition of local strategies, $\alpha t \xrightarrow{*}_{LS} \alpha t[\alpha t|_{p_1}\downarrow]_{p_1} \ldots [\alpha t|_{p_k}\downarrow]_{p_k} = \beta t'$.
If $LS(f) = []$ or $LS(f) = [0, p_2, \ldots, p_n]$, then $t = t'$ and $A = A'$, so $\alpha t = \beta t'$.

Thus, $\alpha t \xrightarrow{*}_S \beta t'$ for every derivation that normalizes all subtems $\alpha t|_j\downarrow$, for $j \in \{p_1, \ldots, p_k\}$. As every $\beta t'$ represents a reduced form of αt on every possible rewriting branch of αt, then S-termination of $\beta t'$ implies S-termination of αt.

Clearly in all cases, β satisfies $A' = A \wedge t|_{p_1}\downarrow = X_{p_1} \ldots \wedge t|_{p_k}\downarrow = X_{p_k}$ provided the X_i are neither in A, nor in $Dom(\alpha)$, which is true since the X_i are fresh variables neither appearing in A, nor in $Dom(\alpha)$. □

Lemma (narrowing lemma). *Let $(\{t\}, A, C)$ be a node of any proof tree, giving the nodes $(\{v_i\}, A'_i, C'_i), i \in [1..l]$, by application of* **Narrow**. *For any ground substitution α satisfying A, if αt is reducible, then, for each $i \in [1..l]$, there exist β_i such that S-termination of the $\beta_i v_i, i \in [1..l]$, implies S-termination of αt. Moreover, β_i satisfies A'_i for each $i \in [1..l]$.*

Proof. We reason by case on the different strategies.

—Either $S = Innermost$, and by Lifting Lemma, there is a term v and substitutions β and $\sigma = \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j}$, corresponding to each rewriting step $\alpha f(u_1, \ldots, u_m) \rightarrow^{p,l\rightarrow r}_{Innermost} t'$, such that:

1. $t = f(u_1, \ldots, u_m) \rightsquigarrow^{p,l\rightarrow r,\sigma}_{Innermost} v$,
2. $\beta v = t'$,
3. $\beta\sigma_0 = \alpha[\mathcal{Y} \cup Var(l)]$
4. β satisfies $\bigwedge_{j\in[1..k]} \overline{\sigma_j}$.

where σ_0 is the most general unifier of $t|_p$ and l and $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 t|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all position p' which are suffix positions of p in t.
These narrowing steps are effectively produced by the rule **Narrow**, applied in all possible ways on $f(u_1, \ldots, u_m)$. So a term βv is produced for every innermost

rewriting branch starting from αt. Then innermost termination of the βv implies innermost termination of αt.
Let us prove that β satisfies $A' = A \wedge \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j}$.
By Lifting Lemma, we have $\alpha = \beta\sigma_0$ on $\mathcal{Y}$. As we can take $\mathcal{Y} \supseteq Var(A)$, we have $\alpha = \beta\sigma_0$ on $Var(A)$.
More precisely, on $Ran(\sigma_0)$, β is such that $\beta\sigma_0 = \alpha$ and on $Var(A) \setminus Ran(\sigma_0)$, $\beta = \alpha$. As $Ran(\sigma_0)$ only contains fresh variables, we have $Var(A) \cap Ran(\sigma_0) = \emptyset$, so $Var(A) \setminus Ran(\sigma_0) = Var(A)$. So $\beta = \alpha$ on $Var(A)$ and then, β satisfies A.
Moreover, as $\beta\sigma_0 = \alpha$ on $Dom(\sigma_0)$, β satisfies σ_0.
So β satisfies $A \wedge \sigma_0$. Finally, with the point 4. of the lifting lemma, we conclude that β satisfies $A' = A \wedge \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j}$.

—Either $S = LS$, and **Narrow** is applied on $\{t = f(u_1, \ldots, u_m)\}$ with $l = [0, i_1, \ldots, i_p]$. For any α satisfying A,
—either $\alpha f(u_1, \ldots, u_m)$ is irreducible at the top position, but may be reduced at positions $i_1, \ldots, i_p$. In this case, either $f(u_1, \ldots, u_m)$ is not narrowable at the top position, either $f(u_1, \ldots, u_m) \rightsquigarrow^{\epsilon,\sigma_i} v_i$ for $i \in [1..l]$ and $A \wedge \sigma_i$ is unsatisfiable for each i, or for $i \in [1..l]$, $f(u_1, \ldots, u_m) \rightsquigarrow^{\epsilon,\sigma_i} v_i$ and $A \wedge \sigma_i$ is satisfiable.
In the first two cases, **Narrow** produces the node $(\{t^{[i_1,\ldots,i_p]}\}, A, C)$, and setting $\beta = \alpha$, we obtain that termination of $\beta t^{[i_1,\ldots,i_p]}$ implies termination of $\alpha t^{[0,i_1,\ldots,i_p]}$, and that β satisfies $A' = A$.
In the third case, **Narrow** produces the node $(\{t^{[i_1,\ldots,i_p]}\}, A \wedge (\bigwedge_{i=1}^{l} \overline{\sigma_i}), C)$, and setting $\beta = \alpha$, we have termination of $\beta t^{[i_1,\ldots,i_p]}$ implies termination of $\alpha t^{[0,i_1,\ldots,i_p]}$. Moreover, as αt is not reducible at the top position, $\alpha = \beta$ satisfies $(\bigwedge_{i=1}^{l} \overline{\sigma_i})$. Thus, as α satisfies A, β satisfies $A' = A \wedge (\bigwedge_{i=1}^{l} \overline{\sigma_i})$.
—or $\alpha f(u_1, \ldots, u_m)$ is reducible at the top position, and by Lifting Lemma, there is a term v and substitutions β and σ_0 corresponding to each rewriting step $\alpha f(u_1, \ldots, u_m) \rightarrow^{\epsilon,l\rightarrow r} t'$, such that:

1. $t = f(u_1, \ldots, u_m) \rightsquigarrow^{\epsilon,l\rightarrow r,\sigma_0} v$,
2. $\beta v = t'$,
3. $\beta\sigma_0 = \alpha[\mathcal{Y} \cup Var(l)]$.

where σ_0 is the most general unifier of t and l.
These narrowing steps are effectively produced by **Narrow**, which is applied in all possible ways on $f(u_1, \ldots, u_m)$ at the top position. So a term βv is produced for every LS-rewriting step applying on αt at the top position. Then termination of the βv implies termination of αt for the given LS-strategy.
We prove that β satisfies $A \wedge \sigma_0$ like in the innermost case, except that there is no negation of substitution here.

—Or $S = Outermost$, and in this case, $t = f(u_1, \ldots, u_n)$ is renamed into $t_0 = f(u_1, \ldots, u_n)^\rho$. A then becomes $A_0 = A \cup R(f(u_1, \ldots, u_n))$ where $\rho = (x_1 \rightarrowtail x'_1) \ldots (x_k \rightarrowtail x'_k)$.
We first show that if every $\beta_0 t_0$ outermost terminates, for β_0 satisfying A_0, then every αt outermost terminates.

If A is satisfiable, then A_0 is satisfiable. Indeed, $A_0 = A \cup f(u_1, \dots, u_m) \rightarrowtail f(u_1, \dots, u_m)^\rho$, with $\rho = (x_1 \rightarrowtail x'_1) \dots (x_k \rightarrowtail x'_k)$. In addition, the x_i are the variables of $f(u_1, \dots, u_n)$.
If $A = \top$, then $A_0 = f(u_1, \dots, u_m) \rightarrowtail f(u_1, \dots, u_m)^\rho$, which is always satisfiable. If $A \neq \top$, since they are the variables of $f(u_1, \dots, u_n)$, the x_i can appear in A, either in abstracted subterms, either as new abstraction variables, either in the right hand-sides of equalities and disequalities defining the substitution of the previous narrowing step, or as new variables introduced by the previous reduction renaming step. In any case, the formula in which they appear is compatible with $f(u_1, \dots, u_m) \rightarrowtail f(u_1, \dots, u_m)^\rho$. Indeed, for the θx_i such that θ satisfies A, θ can be extended on the variables x'_i, in such a way that A_0 is satisfiable. Then $A_0 = A \cup f(u_1, \dots, u_m) \rightarrowtail f(u_1, \dots, u_m)^\rho$ is satisfiable.

By definition of A_0, the β_0 are the α verifying the reduction formula $f(u_1, \dots, u_m) \rightarrowtail f(u_1, \dots, u_m)^\rho$, with $\rho = (x_1 \rightarrowtail x'_1) \dots (x_k \rightarrowtail x'_k)$. We have $Dom(\alpha) = Var(A) \cup \{x_1, \dots, x_k\}$. The domain of β_0 is $Dom(\alpha) \cup \{x'_1, \dots, x'_k\}$. Then $\beta_0 = \alpha\ [Dom(\alpha)]$ and by definition of the reduction formula, the $\beta_0 x'_i$ are such that $t[\beta_0 x'_1]_{p_1} \dots [\beta_0 x'_k]_{p_k}$ is the first reduced form of $\alpha f(u_1, \dots, u_n)$ in any outermost rewriting chain starting from $\alpha f(u_1, \dots, u_n)$, having an outermost rewriting position at a non variable position of $f(u_1, \dots, u_n)$.
Then, by definition of the outermost strategy, the $\beta_0 t_0$ represent any possible outermost reduced form of αt just before the reduction occurs at a non variable occurrence of $f(u_1, \dots, u_n)$. Thus, outermost termination of the $\beta_0 t_0$ implies outermost termination of the αt.

Then t_0 is narrowed in all possible ways into terms v_i at positions p_i with substitutions σ_i, provided p_i and σ_i satisfy the outermost narrowing requirements, as defined in Definition 4.5. We now show that if $\beta_0 t_0$ is reducible, then there exist β_i satisfying A' such that outermost termination of the $\beta_i v_i$ implies outermost termination of $\beta_0 t_0$.
We have $\beta_0 t_0 \rightarrow^{p, l \rightarrow r}_{Outermost} t'$ and $p \in \mathcal{P}os_{\mathcal{F}}(t_0)$ since $t_0 = t^\rho$.
By Lifting Lemma, there is a term v and substitutions β and $\sigma = \sigma_0 \wedge \bigwedge_{j \in [1..k]} \overline{\sigma_j}$, corresponding to each rewriting step $\alpha t_0 \rightarrow^{p, l \rightarrow r}_{Outermost} t'$, such that:

1. $t_0 \rightsquigarrow^{p, l \rightarrow r, \sigma}_{Outermost} v$,
2. $\beta v = t'$,
3. $\beta \sigma_0 = \beta_0[\mathcal{Y} \cup Var(l)]$
4. β satisfies $\bigwedge_{j \in [1..k]} \overline{\sigma_j}$.

where σ_0 is the most general unifier of $t_0|_p$ and l and $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 t_0|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all position p' which are prefix positions of p in t_0.
These narrowing steps are effectively produced by the rule **Narrow**, applied in all possible ways. So a term βv is produced for every outermost rewriting branch starting from $\beta_0 t_0$. Then outermost termination of the βv implies outermost termination of $\beta_0 t_0$.
We prove that β satisfies $A' = A_0 \wedge \sigma_0 \bigwedge_{j \in [1..k]} \overline{\sigma_j}$ like in the innermost case.

□

LEMMA (STOPPING LEMMA). *Let $(\{t\}, A, C)$ be a node of any proof tree, with A satisfiable, and giving the node $(\emptyset, A', C')$ by application of an inference rule. Then for every ground substitution α satisfying A, αt S-terminates.*

PROOF. The only rule giving the node $(\emptyset, A', C')$ is **Stop**. When **Stop** is applied, then

—either $TERMIN(S, t)$ and then αt S-terminates for every ground substitution α,

—or $(t_{ref} > t)$ is satisfiable. Then, for every ground substitution α satisfying A, $\alpha t_{ref} \succ \alpha t$. By induction hypothesis, αt S-terminates.

□

C. THE USABLE RULES

To prove Lemma 6.2, we need the next three lemmas. The first two ones are pretty obvious from the definition of the usable rules.

LEMMA C.1. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. Then, every symbol $f \in \mathcal{F}$ occurring in t is such that $Rls(f) \subseteq \mathcal{U}(t)$.*

PROOF. We proceed by structural induction on t.

—If $t \in \mathcal{X} \cup \mathcal{X}_A$, the property is trivially true;

—if t is a constant a, $\mathcal{U}(t = a) = Rls(a) \cup_{l \to r \in Rls(a)} \mathcal{U}(r)$; the only symbol of t is a, and we have $Rls(a) \subseteq \mathcal{U}(t)$.

Let us consider a non-constant and non-variable term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$, of the form $f(u_1, \ldots, u_n)$. Then, by definition of $\mathcal{U}(t)$, we have $\mathcal{U}(t) = Rls(f) \cup_{i=1}^{n} \mathcal{U}(u_i) \cup_{l \to r \in Rls(f)} \mathcal{U}(r)$. Then, whatever g symbol of t, either $g = f$ and then $Rls(g) \subseteq \mathcal{U}(t)$, or g is a symbol occurring in some u_i and, by induction hypothesis on u_i, $Rls(g) \subseteq \mathcal{U}(u_i)$, with $\mathcal{U}(u_i) \subseteq \mathcal{U}(t)$. □

LEMMA C.2. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. Then $l \to r \in \mathcal{U}(t) \Rightarrow \mathcal{U}(r) \subseteq \mathcal{U}(t)$.*

PROOF. According to the definition of the usable rules, if a term t is such that $\mathcal{V}ar(t) \cap \mathcal{X} \neq \emptyset$, then $\mathcal{U}(t) = \mathcal{R}$, and then the property is trivially true. We will then suppose in the following that t does not contain any variable of $\mathcal{X}$.

Let $l \to r \in \mathcal{U}(t)$. By definition of $\mathcal{U}(t)$, since $\mathcal{V}ar(t) \cap \mathcal{X} = \emptyset$, among all recursive applications of the definition of $\mathcal{U}$ in $\mathcal{U}(t)$, there is an application $\mathcal{U}(t')$ of $\mathcal{U}$ to some term t' such that $\mathcal{U}(t') = Rls(g) \cup_i \mathcal{U}(t'|_i) \cup_{l' \to r' \in Rls(g)} \mathcal{U}(r')$, with $\mathcal{U}(t') \subseteq \mathcal{U}(t)$, and $l \to r \in Rls(g)$, with $g = top(l)$.

Since $l \to r \in Rls(g)$, by definition of $\mathcal{U}(t')$, we have $\mathcal{U}(r) \subseteq \cup_{l' \to r' \in Rls(g)} \mathcal{U}(r')$, and then $\mathcal{U}(r) \subseteq \mathcal{U}(t') \subseteq \mathcal{U}(t)$. □

LEMMA C.3. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. For any ground normalized substitution α and any rewrite chain $\alpha t \to^{p_1, l_1 \to r_1} t_1 \to^{p_2, l_2 \to r_2} t_2 \to \ldots \to^{p_n, l_n \to r_n} t_n$, the defined symbol of $t_k, 1 \leq k \leq n$ at a redex position of t_k is either a symbol of t or one of the $r_i, i \in [1..k]$.*

Proof. We proceed by induction on the length of the derivation. The property is obviously true for an empty derivation i.e. on αt.

Let us show the property for the first rewriting step $\alpha t \rightarrow^{p_1, l_1 \rightarrow r_1} t_1$. By definition of rewriting, $\exists \sigma : \sigma l_1 = \alpha t|_{p_1}$ and $t_1 = \alpha t[\sigma r_1]_{p_1}$. Let f be the redex symbol of t_1 at a position p, and let us show that f comes either from t or from r_1.

Since $t_1 = \alpha t[\sigma r_1]_{p_1}$, either p is a position of the context $\alpha t[]_{p_1}$, which does not change by rewriting, so we already have f as redex symbol of αt at position p. As α is normalized, p is a position of t, so f is a symbol of t.

Either p corresponds in t_1 to a non variable position of r_1, so f is a symbol of r_1.

Or p corresponds in t_1 to a position p_2 in σx, for a variable $x \in \mathcal{V}ar(r_1)$ at position q in r_1: we have $p = p_1 q p_2$. In this case, since $\mathcal{V}ar(r_1) \subseteq \mathcal{V}ar(l_1)$, we have $x \in \mathcal{V}ar(l_1)$, so σx is also a subterm of αt, and f occurs in αt at position $p' = p_1 q' p_2$, where q' is a position of x in l_1.

Moreover, as p is a redex position in t_1, then by definition of the innermost strategy, there is no suffix redex position of p in t_1. As $t_1|_p = \alpha t|_{p'}$, then similarly p' is a redex position in αt. As α is normalized, p' is a position of t, so f is a symbol of t.

Then, let us suppose the property true for any term of the rewrite chain $\alpha t \rightarrow^{p_1, l_1 \rightarrow r_1} t_1 \rightarrow \ldots \rightarrow^{p_k, l_k \rightarrow r_k} t_k$, i.e. any redex symbol f of t_k is also a symbol of t, or a symbol of one of the $r_i, i \in [1..k]$, and let us consider $t_k \rightarrow^{p_{k+1}, l_{k+1} \rightarrow r_{k+1}} t_{k+1}$.

By a similar reasoning than previously, we establish that any redex symbol f of t_{k+1} is also a symbol of t_k, or a symbol of r_{k+1}. We then conclude with the previous induction hypothesis. □

We are now able to prove Lemma 6.2.

Lemma 6.2. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. For any ground instance αt of t and any rewrite chain $\alpha t \rightarrow^{p_1, l_1 \rightarrow r_1} t_1 \rightarrow^{p_2, l_2 \rightarrow r_2} t_2 \rightarrow \ldots \rightarrow^{p_n, l_n \rightarrow r_n} t_n$, then $l_i \rightarrow r_i \in \mathcal{U}(t)$, $\forall i \in [1..n]$.*

Proof. If a variable $x \in \mathcal{X}$ occurs in t, then $\mathcal{U}(t) = \mathcal{R}$ and the property is trivially true. We then consider in the following that $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}_A)$, and then that α is a (ground) normalized substitution.
We proceed by induction on $\mathcal{T}(\mathcal{F}, \mathcal{X}_A)$ and on the length of the derivation.

The property is trivially true if αt is in normal form. For any $\alpha t \rightarrow^{p_1, l_1 \rightarrow r_1} t_1$, since α is normalized, p_1 corresponds in αt to a non-variable position of t. Let f be the symbol at position p_1 in t. Since f is the symbol at the redex position p_1 of αt with the rule $l_1 \rightarrow r_1$, then $l_1 \rightarrow r_1 \in Rls(f)$. Moreover, thanks to Lemma C.1, $Rls(f) \subseteq \mathcal{U}(t)$. Therefore, $l_1 \rightarrow r_1 \in \mathcal{U}(t)$.

Let us now suppose the property is true for any derivation chain starting from αt whose length is less or equal to k, and consider the chain: $\alpha t \rightarrow^{p_1, l_1 \rightarrow r_1} t_1 \rightarrow^{p_2, l_2 \rightarrow r_2} t_2 \rightarrow \ldots \rightarrow^{p_k, l_k \rightarrow r_k} t_k \rightarrow^{p_{k+1}, l_{k+1} \rightarrow r_{k+1}} t_{k+1}$. Let f be the symbol at position p_{k+1} in t_k. Since p_{k+1} is a redex position of t_k with the rule $l_{k+1} \rightarrow r_{k+1}$, then $l_{k+1} \rightarrow r_{k+1} \in Rls(f)$.

By Lemma C.3 with a derivation of length k, we have two cases:

—either the symbol f at position p_{k+1} in t_k is a symbol of t; then, thanks to Lemma C.1 on t, we get $Rls(f) \subseteq \mathcal{U}(t)$; henceforth $l_{k+1} \rightarrow r_{k+1} \in \mathcal{U}(t)$;

—or the symbol f at position p_{k+1} in t_k is a symbol of a $r_i, i \in [1..k]$; then, thanks to Lemma C.1 on r_i, we get $Rls(f) \subseteq \mathcal{U}(r_i)$; henceforth $l_{k+1} \to r_{k+1} \in \mathcal{U}(r_i)$; by induction hypothesis we have $l_i \to r_i \in \mathcal{U}(t)$ and, thanks to Lemma C.2, we have $\mathcal{U}(r_i) \subseteq \mathcal{U}(t)$. Henceforth $l_{k+1} \to r_{k+1} \in \mathcal{U}(t)$.

□

PROPOSITION 6.3. *Let $\mathcal{R}$ be a rewrite system on a set $\mathcal{F}$ of symbols, and t a term of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{X}_A)$. If there exists a reduction ordering $\succ$ such that $\forall l \to r \in \mathcal{U}(t) : l \succ r$, then any ground instance of t is terminating.*

PROOF. As $\succ$ orients the rules used in any reduction chain starting from αt for any ground substitution α, by properties of the reduction orderings, $\succ$ also orients the reduction chains, which are then finite. □

D. A LEMMA SPECIFIC TO THE OUTERMOST CASE

LEMMA 7.5. *Let $(\{t_i\}, A_i, C_i)$ be the ith node of any branch of the proof tree obtained by applying the strategy $Strat{-}Rules(Outermost)$ on $(\{t_{ref}\}, \top, \top)$, and $\succ$ a noetherian ordering having the subterm property. If every reduction formula in A_i can be reduced to a formula $\bigwedge_j x_j = x'_j$, then we have:*

for every variable x of t_i in $\mathcal{X}$: $(t_{ref} > x)/A_i$ is satisfiable by $\succ$.

PROOF. The proof is made by induction on the number i of applications of the inference rules from $(\{t_{ref}\}, \top, \top)$ to the node $(\{t_i\}, A_i, C_i)$.

Let us prove that the property holds for $i = 0$. We have $t_0 = t_{ref}$ and then $Var(t_0) = \mathcal{V}ar(t_{ref})$. Consequently, for every $x \in Var(t_0)$, whatever the ground substitution α such that $\mathcal{V}ar(t_{ref}) \subseteq Dom(\alpha)$, αx is a subterm of αt_{ref}. The induction ordering $\succ$ satisfying the conditions of the rules before the application of these rules can be any noetherian ordering having the subterm property. We then have $\alpha t_{ref} \succ \alpha x$.

We now prove that if the property holds for $i - 1$, it also holds for i.

If the rule used at the ith step is **Stop**, then $Var(t_i) = \emptyset$, and then, the property is trivially verified.

If the rule used at the ith step is **Abstract**, as the rule **Abstract** replaces subterms in t_{i-1} by new variables of $\mathcal{X}_A$, then $(Var(t_i) \cap \mathcal{X}) \subseteq (Var(t_{i-1}) \cap \mathcal{X})$, so the property still holds.

If the rule used at the ith step is **Narrow** then, by hypothesis, the reduction renaming applied to t_{i-1} and giving a term t'_{i-1} just consists in a mere renaming of the variables of t_{i-1}. Let t_i be a term obtained by narrowing t'_{i-1} with the substitution σ.

Let $z \in \mathcal{V}ar(t_i)$, and α a substitution satisfying A_i. We show that $\alpha t_{ref} \succ \alpha z$. We have two cases.

Either z is a fresh variable introduced by the narrowing step. Let $x' \in \mathcal{V}ar(t'_{i-1})$ such that $z \in \mathcal{V}ar(\sigma x')$, and $x \in \mathcal{V}ar(t_{i-1})$ such that x' is a renaming of x. By hypothesis, every reduction formula in A_i can be reduced to a formula $\bigwedge_j x_j = x'_j$.

This is then the same for A_{i-1}. Moreover, since α satisfies A_i, then it satisfies in particular A_{i-1}. Then, by induction hypothesis, $\alpha t_{ref} \succ \alpha x$ and, since α satisfies $x = x'$, we also have $\alpha t_{ref} \succ \alpha x'$.
By hypothesis, σ contains the equality $x' = C[z]$, with $C[z]$ a (possibly empty) context of z. Moreover, by definition of the rule **Narrow**, $A_i = A_{i-1} \wedge R(t_{i-1}) \wedge \sigma$. So A_i contains the equality $x' = C[z]$.
Then, as α satisfies A_i, α is such that $\alpha x' = \alpha C[z]$. Since $\alpha t_{ref} \succ \alpha x'$, we have $\alpha t_{ref} \succ \alpha C[z]$ and then, by subterm property, $\alpha t_{ref} \succ \alpha z$.

Or $z \in \mathcal{V}ar(t'_{i-1})$; by the same reasoning as in the previous point for x', we have $\alpha t_{ref} \succ \alpha z$. $\square$