Termination of Priority Rewriting - Extended version

Isabelle Gnaedig

Termination of priority rewriting

Isabelle GNAEDIG

LORIA-INRIA
BP 239 F-54506 Vandœuvre-lès-Nancy Cedex
Phone: + 33 3 54 95 84 21
Fax: + 33 3 54 95 84 01
e-mail: Isabelle.Gnaedig@loria.fr

Abstract. Introducing priorities on rules increases the expressive power of rewriting and helps to limit computations. Priority rewriting is used as a computation model for the functional paradigm. Termination of priority rewriting then warrants that programs give a result. We describe in this paper an inductive proof method for termination of priority rewriting, lying on an inductive mechanism on the termination property. It works by generating proof trees, using abstraction and narrowing. As it specifically handles priorities on the rules, our technique allows proving termination of rewrite systems that would diverge without priorities.

1 Introduction

In [4, 3], priority rewriting systems (PRS's in short) have been introduced. A PRS is a rewrite system (RS in short) with a partial ordering on rules, determining a priority between some of them. Considering priorities on the rewrite rules to be used can be very useful for an implementation purpose to reduce the indeterminism of computations, or to enable divergent system to terminate, and for a semantical purpose, to increase the expressive power of rewriting. Priority rewriting is used as a computation model for functional programming [12], and is underlying in the functional strategy, used for example in Lazy ML [1], Clean [5], or Haskell [9].

But priority rewriting is delicate to handle. First, the priority rewriting relation is not always decidable, because a term rewrites with a given rule only if in the redex, there is no reduction leading to another redex, reducible with a rule of higher priority.

A way to overcome the undecidability is to force evaluation of the terms in reducing subterms to strong head normal form via some strategy [12]. But normalization can lead to non-termination. In [11], a decidable innermost definition of rewriting using PRS's has been proposed.

Second, the semantics of a PRS is not always clearly defined. In [4], a semantics is proposed, lying on the notion of unique sound an complete set of closed instances of the rules of the PRS, and it is showed that bounded -a weaker property than termination- PRS's have a semantics. In [15], a fixed point based technique is proposed to compute the semantics of a PRS. It is also proved that for a bounded PRS with finitely many rules, the set of successors of any term is finite and computable. In [11], a logical semantics of PRS's, more closed to the notion of equational logic is given. A particular class of PRS's is proved sound and complete with respect to the initial algebra, provided every priority rewriting sequence from every ground term terminates.

Then the termination problem of the priority rewriting relation naturally arises, either to warrant that it has a semantics, or to ensure that rewriting computations always give a result. Surprisingly, it seems not to have been much investigated until now.

To our knowledge, it has only been specifically addressed in [11], where the use of reduction orderings is extended with an instantiation condition on rules linked with the priority order. Let us cite also [13], discussing a normalizing strategy of PRSs, i.e a strategy giving only finite derivations for terms having a normal form with usual rewriting.

Our purpose here is to consider termination of priority rewriting from an operational point of view, with the concern of warrantying a result for every computation. So it seems us interesting to focus on the innermost priority rewriting of [11], because it is decidable, easy to manipulate, and the innermost strategy is often used in programming contexts where priorities on rules are considered.

Obviously, a PRS is terminating if the underlying RS is. So usual rewriting termination proof techniques can be used for priority rewriting. Here, we propose to be finer in considering non (innermost) terminating RS's, that precisely become terminating using priorities on rules.

We use an inductive method, whose principle has already been applied for proving termination of rewriting (without priorities) under strategies [8]. The idea is to prove on the ground term algebra that every derivation starting from any term terminates, supposing that it is true for terms smaller than the starting terms for an induction ordering.

The background is given in Section 2. In Section 3, we present the inductive proof principle of our approach. Section 4 develops the basic concepts of the inductive proof mechanism based on abstraction and narrowing, and the involved constraints. Section 5 presents the proof procedure and states the general theorem with its conditions of application. We conclude in Section 6.

2 The background

We assume that the reader is familiar with the basic definitions and notations of term rewriting given for instance in [2, 6, 14]. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is the set of terms built from a given finite set $\mathcal{F}$ of function symbols f having arity $n \in \mathbb{N}$, and a set $\mathcal{X}$ of variables denoted $x, y \ldots$. $\mathcal{T}(\mathcal{F})$ is the set of ground terms (without variables). The terms 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. Let p and p' be two positions. The position p' is said to be (a strict) suffix of p if $p' = p\lambda$, where λ is a (non-empty) sequence of integers. For a position p of a term t, we note $t|_p$ the subterm of t at position p, and $t[s]_p$ the term obtained in replacing by s the subterm at position p in t.

A substitution is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F}, \mathcal{X})$, written $\sigma = (x = t, \ldots, y = u)$. 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 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 $l \to r$, such that $l \notin \mathcal{X}$ and $Var(r) \subseteq Var(l)$. 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 rule, and is called a *defined function symbol* otherwise. The set of defined function symbols is denoted $\mathcal{D}_R$ ($\mathcal{R}$ is omitted when

there is no ambiguity). In this paper, we only consider finite sets of function symbols and 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 is 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. The reflexive transitive closure of the rewriting relation induced by $\mathcal{R}$ is denoted by $\stackrel{*}{\rightarrow}^{\mathcal{R}}$. The innermost rewriting relation consists of always rewriting at the lowest possible positions.

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 infinitely 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 $\mathcal{T}(\mathcal{F}, \mathcal{X})$, $f(\ldots t \ldots) \succ t$.

For $\mathcal{F}$ and $\mathcal{X}$ finite, if $\succ$ is monotone and has the subterm property, then it is noetherian [10]. If, in addition, $\succ$ is stable under substitution (for any substitution σ, any pair of terms $t, t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $t \succ t'$ implies $\sigma t \succ \sigma t'$), then it is called a simplification ordering.

A priority rewrite system (PRS in short) is a pair $(\mathcal{R}, >)$ of an underlying rewrite system $\mathcal{R}$ and a partial ordering $>$ on the rules of $\mathcal{R}$. A rule r_1 has a higher priority than a rule r_2 iff $r_1 > r_2$, which is written $\downarrow^{r_1}_{r_2}$.

Definition 1. *[11] Let $\mathcal{R}$ be a PRS on $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A term s is IP-reducible and (IP-) rewrites to t at position p with the rule $l \rightarrow r$, and the substitution σ which is written $s \rightarrow^{IP}_{p,l\rightarrow r,\sigma} t$ iff:*

- *$s \rightarrow_{p,l\rightarrow r,\sigma} t$*
- *no proper subterm of the redex $s|_p$ is IP-reducible*
- *$s|_p$ is not IP-reducible by any rule in $\mathcal{R}$ of higher priority than $l \rightarrow r$.*

Example 1. With the rewrite system

$$\begin{array}{l} f(g(x)) \rightarrow b\ (1) \\ \big\downarrow\ g(a) \rightarrow c\ (2) \\ \big\downarrow\ g(a) \rightarrow d\ (3) \end{array}$$

on the term $f(g(a))$, Rule (1) should apply, but this would not be an innermost rewriting step. So Rule (2) applies, but Rule (3) does not, because (2) > (3).

A RS $\mathcal{R}$ IP-terminates if and only if every IP-derivation of the rewriting relation induced by $\mathcal{R}$ is finite. If $t \stackrel{*}{\rightarrow}^{IP} t'$ with t' IP-irreducible, then t' is called a(n) (IP-) normal form of t and denoted by $t{\downarrow}$. Note that given t, $t{\downarrow}$ may be not unique.

3 Inductively proving innermost termination of priority rewriting

We prove termination of IP-rewriting by induction on the ground terms. Working on ground terms is not a restriction since the algebraic semantics of rule-based languages is often initial. Moreover, in [11], to warrant stability by substitution of the innermost rewriting relation, the rules without highest priority only can reduce ground terms.

For proving that a priority rewrite system on $\mathcal{T}(\mathcal{F})$ IP-terminates, we reason with a local notion of termination on terms: a term t of $\mathcal{T}(\mathcal{F})$ is said to be IP-terminating for a PRS $\mathcal{R}$ if every IP-rewriting chain (or derivation) starting from t is finite.

For proving that a term t of $\mathcal{T}(\mathcal{F})$ is IP-terminating, we proceed by induction on $\mathcal{T}(\mathcal{F})$ with a noetherian ordering $\succ$, assuming the property for every t' such that $t \succ t'$. To warrant non emptiness of $\mathcal{T}(\mathcal{F})$, and a basis for the induction, we assume that $\mathcal{F}$ contains at least one constructor constant. The main intuition is to observe rewriting derivations starting from a ground term $t \in \mathcal{T}(\mathcal{F})$ which is any instance of a pattern $g(x_1, \ldots, x_m) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, for some defined function symbol $g \in \mathcal{D}$, and variables $x_1, \ldots, x_m$. Proving the property of IP-termination on ground terms amounts to proving that every ground instance of the patterns $g(x_1, \ldots, x_m)$ is terminating.

Rewriting derivations are simulated, using a lifting mechanism, by a proof tree developed from $g(x_1, \ldots, x_m)$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, for every $g \in \mathcal{D}$, by alternatively using narrowing and an abstraction mechanism. Narrowing schematizes the rewriting possibilities of terms, abstraction simulates the reduction of subterms in the derivations until these subterms become normal forms. It expresses the application of the induction hypothesis on these subterms, for which, as they are supposed to be IP-terminating, a normal form exists.

The schematization of ground rewriting derivations is achieved through constraints. The nodes of the developed proof trees are 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 steps. Each node in an abstract tree schematizes the set of ground instances of the current term, which 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.

For a term t of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ occurring in a proof tree issued from a reference term $t_{ref} = g(x_1, \ldots, x_m)$,

- first, some subterms $\theta t|_j$ of θt are supposed to be IP-terminating for every θ solution of the constraint associated to t, by the induction hypothesis, if $\theta t_{ref} \succ \theta t|_j$ for the induction ordering $\succ$. So the $t|_j$ are replaced in t by *abstraction variables* X_j representing respectively any of their normal forms. Reasoning by induction allows us to only suppose the existence of the normal forms *without explicitly computing them*. If the ground instances of the resulting term are IP-terminating (either if the induction hypothesis can be applied to them, or if they can be proved IP-terminating by other means, which we will present later), then the ground instances of the initial term are IP-terminating. Otherwise,
- the resulting term $u = t[X_j]_{\{i_1,\ldots,i_p\}}$ (where $i_1, \ldots, i_p$ are the abstraction positions in t) is narrowed in all possible ways into terms v, with an appropriate narrowing relation corresponding to IP-rewriting u according to the possible instances of the X_j.

Then IP-termination of the ground instances of t is reduced to IP-termination of the ground instances of the terms v. Now, if $\theta t_{ref} \succ \theta v$ for every ground substitution θ that is a solution of the constraint associated to v, by the induction hypothesis, θv is supposed to be IP-terminating. Otherwise, the process is iterated on v, until we get a term t' such that either $\theta t_{ref} \succ \theta t'$, or $\theta t'$ can be proved IP-terminating.

This technique was inspired from the one we proposed for proving the innermost termination of classical rewrite systems in [8]. We now detail the concepts needed to formalize and automate it.

4 Abstraction, narrowing, 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.

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 2 (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 is 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 is 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 $\succ_{\mathcal{P}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to satisfy C is that $t \succ_{\mathcal{P}} t'$ for every constraint $t > t'$ of C, and $\succ_{\mathcal{P}}$ is stable under substitution.

Simplification orderings fulfill such a condition. So, in practice, it is sufficient to find a simplification ordering $\succ_{\mathcal{P}}$ such that $t \succ_{\mathcal{P}} t'$ for every constraint $t > t'$ of C.

The ordering $\succ_{\mathcal{P}}$, defined on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, can then be seen as an extension of the induction ordering $\succ$ on $\mathcal{T}(\mathcal{F})$. For convenience sake, $\succ_{\mathcal{P}}$ will also be written $\succ$.

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, or the Lexicographic Path Ordering. This is often sufficient for the constraints considered here: 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.

4.1 Abstraction

Let us define the abstraction variables more formally.

Definition 3. *Let* $\mathcal{N}$ *be a set of variables disjoint from* $\mathcal{X}$. *Symbols of* $\mathcal{N}$ *are called* abstraction variables. *Substitutions and instantiations are extended to* $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$ *in the following way: for any substitution* σ *(resp. instantiation* θ*) such that* $Dom(\sigma)$ *(resp.* $Dom(\theta)$*) contains a variable* $X \in \mathcal{N}$, σX *(resp.* θX*) is in* IP*-normal form.*

Definition 4 (term abstraction). *The term* $t[t|_j]_{j \in \{i_1, \ldots, i_p\}}$ *is said to be* abstracted *into the term* u *(called* abstraction *of* t*) at positions* $\{i_1, \ldots, i_p\}$ *iff* $u = t[X_j]_{j \in \{i_1, \ldots, i_p\}}$, *where the* $X_j, j \in \{i_1, \ldots, i_p\}$ *are fresh distinct abstraction variables.*

IP-termination on $\mathcal{T}(\mathcal{F})$ is in fact proved by reasoning on terms with abstraction variables, i.e. on terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$. Ordering constraints are extended to pairs of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$. When subterms $t|_j$ are abstracted by X_j, we state constraints on 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 $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$, and $X \in \mathcal{N}$, but we will see later how they are combined with the substitutions used for the narrowing process.

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

4.2 Narrowing

After abstracting the current term t into $t[X_j]_{j\in\{i_1,\ldots,i_p\}}$, we test whether the possible ground instances of $t[X_j]_{j\in\{i_1,\ \ldots,i_p\}}$ are reducible, according to the possible values of the instances of the X_j. This is achieved by innermost narrowing $t[X_j]_{j\in\{i_1,\ldots,i_p\}}$, with the priority rewrite system.

In a first time, to schematize innermost rewriting on ground terms, we need to refine the usual notion of narrowing. In fact, with the usual innermost narrowing relation, if a position p in a term t is a narrowing position, no suffix position of p can be a narrowing position as well. However, if we consider ground instances of t, we can have rewriting positions p for some instances, and p' for other instances, such that p' is a suffix of p. So, when using the narrowing relation to schematize innermost rewriting of ground instances of t, the narrowing positions p to consider depend on a set of ground instances of t, which is defined by excluding the ground instances of t that would be narrowable at some suffix position of p. For instance, with the RS $R = \{g(a) \to a, f(g(x)) \to b\}$, the innermost narrowing positions of the term $f(g(X))$ are 1 with the narrowing substitution $\sigma = (X = a)$, and ϵ with any σ such that $\sigma X \neq a$.

Let σ be a substitution on $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$. In the following, we identify σ with the equality formula $\bigwedge_i (x_i = t_i)$, with $x_i \in \mathcal{X} \cup \mathcal{N}$, $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$. Similarly, we call *negation* $\overline{\sigma}$ of the substitution σ the formula $\bigvee_i (x_i \neq t_i)$. The negation of Id means that no substitution can be applied.

Definition 5. *A substitution σ is said* to satisfy *a constraint $\bigwedge_j \bigvee_{i_j} (x_{i_j} \neq t_{i_j})$, iff for every ground instantiation θ, $\bigwedge_j \bigvee_{i_j} (\theta\sigma x_{i_j} \neq \theta\sigma t_{i_j})$. 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, and $\bigwedge_j \bigvee_{i_j} (x_{i_j} \neq t_{i_j})$ the constraint to be satisfied by σ_0.*

Definition 6 (Innermost narrowing). *A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$* innermost narrows *into a a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$ 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 $t \leadsto^{Inn}_{p,l\to r,\sigma} t'$ 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 suffix position p' of p in t.

Notice that 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. So, the narrowing substitutions we consider are restricted to the variables of the narrowed term t.

Now, we have to see how to simulate the IP-rewriting steps of a given term following the possible instances of its variables, by narrowing it with the rules, considering their priority. Unlike for simulating rewriting without priorities, where the narrowing process only depends of the term to be rewritten and of the rule considered, simulating IP-rewriting of ground instances of a term with a given rule requires to consider the narrowing steps with the rules having a higher priority. This also requires to use negations of substitutions. Let us consider the following example:

$$\left\downarrow \begin{array}{l} f(g(x), y) \to a \ (1) \\ f(x, h(y)) \to b \ (2) \\ f(x, y) \to c \quad (3). \end{array} \right.$$

The term $f(x, y)$ innermost narrows into a with the first rule and the most general unifier $(x = g(x'))$, then into b with the second rule, the most general unifier $(y = h(y'))$ and the constraint $x \neq g(x')$, and finally with the third rule into c with the most general unifier equal to Id and the constraint $x \neq g(x') \wedge y \neq h(y')$. So, applying the rules the one after the other, with the current narrowing most general unifier we have to cumulate the negation of the most general unifiers of the previous constrained substitutions, ignoring their constraint part.

If the narrowing substitutions $\sigma_0 \wedge \bigwedge_{j \in [1..k]} \overline{\sigma_j}$ of the previous rules have a constraint part coming from the innermost mechanism of Definition 6, this constraint part is also ignored by the priority mechanism. Indeed, the constraint part is defined from σ_0, and has no meaning for the negation of σ_0. With the PRS:

$$\left|\begin{array}{ll} f(g(h(x))) \to a & (1) \\ h(a) \to b & (2) \\ f(g(x)) \to c & (3) \end{array}\right.$$

the term $f(x)$ innermost narrows with Rule (1) and $\sigma_1 = (x = g(h(x')) \wedge x \neq a)$, Rule (2) does not apply, and Rule (3) applies with $\sigma_3 = (x = g(x'') \wedge x \neq g(h(x')))$.

If on the contrary, the constraint part of a substitution is due to the priority mechanism, the negation of this substitution by the innermost mechanism also only considers the most general unifier of the substitution. With the PRS:

$$\left|\begin{array}{ll} f(g(h(x,y)),z) \to a & (1) \\ f(x,y) \to b & (2) \\ h(a,x) \to a & (3) \\ h(x,b) \to b & (4) \end{array}\right.$$

the term $f(x, y)$ innermost narrows into a with Rule (1) and the constrained substitution $(x = g(h(x', y')) \wedge x' \neq a \wedge y' \neq b)$, because $h(x', y')$ narrows with Rule (3) and the substitution $(x' = a)$, and with Rule (4) and the substitution $(y' = b \wedge x' \neq a)$.

The term $f(x, y)$ also innermost narrows into b with Rule (2) and the substitution $(Id \wedge x \neq g(h(x', y')))$.

Definition 7 (Innermost priority narrowing). *Let $\mathcal{R}$ be a priority rewrite system. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$ IP-*narrows *into a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$ at the non-variable position p of t, using the rule $l \to r \in \mathcal{R}$ with the constrained substitution $\sigma = \sigma_0 \wedge \bigwedge_{j \in [1..k]} \overline{\sigma_j} \bigwedge_{i \in [1..n]} \overline{\sigma_0^i}$, which is written $t \rightsquigarrow^{IP}_{p,l \to r,\sigma} t'$ 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, $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 t|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all suffix position p' of p in t, and $\sigma_0^1, \ldots, \sigma_0^n$ are the most general unifiers of $t|_p$ with the left-hand sides of the rules having a greater priority than $l \to r$.

4.3 Cumulating constraints

Abstraction constraints have to be combined with the narrowing substitutions to characterize the ground terms schematized by the current term t in the proof tree. Indeed, a narrowing branch on the current term u with narrowing substitution σ represents a rewriting branch for any ground instance of σu.

In addition, σ has to satisfy the constraints on variables of u, already set in A. So, σ, considered as the narrowing constraint attached to the narrowing branch, is added to A. This leads to the introduction of abstraction constraint formulas.

Definition 8. *An* abstraction constrained formula *(ACF in short) is a formula* $\bigwedge_i(t_i\downarrow = t'_i) \wedge \bigwedge_j(x_j = u_j)$*, where* $x_j \in \mathcal{X} \cup \mathcal{N}$*,* $t_i, t'_i, u_j, \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$*.*

Definition 9. *An abstraction constrained formula* $A = \bigwedge_i(t_i\downarrow = t'_i) \wedge \bigwedge_j(x_j = u_j)$ *is* satisfiable *iff there is at least one instantiation* θ *such that* $\bigwedge_i(\theta t_i\downarrow = \theta t'_i) \wedge \bigwedge_j(\theta x_j = \theta u_j)$*. The instantiation* θ *is then said to satisfy the ACF* A *and is called solution of* A*.*

For a better readability on examples, we can propagate σ into A (by applying the substitution part of σ 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 u in the proof trees; the ground substitutions solutions of A define the instances of the current term u, for which we are observing IP-termination. When A has no solution, the current node of the proof tree represents no ground term. Such nodes are then irrelevant for the proof. Detecting and suppressing them during a narrowing step allows us to control the narrowing mechanism, well known to easily diverge. 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.

Checking the satisfiability of A is in general undecidable, but it is often easy in practice to exhibit an instantiation satisfying it. Automatable sufficient conditions, lying in particular on the characterization of normal forms, are also under study. The unsatisfiability of A is also undecidable in general, but simple automatable sufficient conditions can be used [8], as to test whether A contains equalities $t\downarrow = u$, where u is reducible. In the following, we present the procedure exactly simulating the rewriting trees, i.e. dealing with the satisfiability of A.

5 The IP-termination procedure

We are now ready to describe the inference rules defining our mechanism. They transform a set T of 3-tuples (U, A, C) where $U = \{t\}$ or $\emptyset$, t is the current term whose ground instances have to be proved IP-terminating, A is an abstraction constraint formula, C is a conjunction of ordering constraints.

- The first rule abstracts the current term t at given positions $i_1, \ldots, i_p$ into $t[X_j]_{j\in\{i_1,\ldots,i_p\}}$. The constraint $\bigwedge_{j\in\{i_1,\ldots,i_p\}} t_{ref} > t|_j$ is set in C. The abstraction constraint $\bigwedge_{j\in\{i_1,\ldots,i_p\}} t|_j\downarrow = X_j$ is added to the ACF A. We call this rule **Abstract**.
  The abstraction positions are chosen so that the abstraction mechanism captures the greatest possible number of rewriting steps: then we abstract all of the greatest possible subterms of $t = f(t_1, \ldots, t_m)$. More concretely, we try to abstract $t_1, \ldots, t_m$ and, for each $t_i = g(t'_1, \ldots, t'_n)$ that cannot be abstracted, we try to abstract $t'_1, \ldots, t'_n$, and so on. In the worst case, we are driven to abstract leaves of the term, which are either variables, or constants.
  Note also that it is not useful to abstract non-narrowable subterms of $\mathcal{T}(\mathcal{F}, \mathcal{N})$. Indeed, by Definition 3, every ground instance of such subterms is in IP-normal form.
- The second rule narrows the resulting term u in all possible ways in one step, with all possible rewrite rules of the rewrite system $\mathcal{R}$, and all possible substitutions, into terms $v_1, \ldots, v_q$, according to Definition 7. This step is a branching step, creating as states as there are narrowing possibilities. The substitution σ is integrated to A. This is the **Narrow** rule.

Table 1. Inference rules for IP-termination

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

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

Narrow: $$\frac{\{t\},\ A,\ C}{\{v_i\},\ A \wedge \sigma,\ C}$$

if if $t \rightsquigarrow^{IP}_{\sigma} v_i$ and $A \wedge \sigma$ is satisfiable

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

if $(C \wedge H_C(t))$ is satisfiable.

and $H_A(t) = \begin{cases} \top & \text{if } t \text{ is in } \mathcal{T}(\mathcal{F},\mathcal{N}) \text{ and is not narrowable} \\ t\downarrow = X & \text{otherwise.} \end{cases}$

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

– We finally have a **Stop** rule halting the proof process on the current branch of the proof tree, when the ground instances of the current term can be stated as IP-terminating. This happens when the whole current term u can be abstracted, i.e. when the induction hypothesis is applied to it, when $u \in \mathcal{T}(\mathcal{F},\mathcal{N})$ and is not narrowable, or when $u \in \mathcal{T}(\mathcal{F},\mathcal{N})$ is narrowable with substitutions σ such that $A \wedge \sigma$ is not satisfiable.

Let us note that the inductive reasoning can be completed as follows. When the induction hypothesis cannot be applied to a term u, it may be possible to prove IP-termination of every ground instance of u in another way. Let $IPT(u)$ be a predicate that is true iff every ground instance of u is IP-terminating. In the previous first and third steps of the inductive reasoning, we then associate the alternative predicate $IPT(u)$ to the condition $t > u$. It is true in particular when $u \in \mathcal{T}(\mathcal{F},\mathcal{N})$ and is not narrowable, as said above. Otherwise, we can use the notion of usable rule, as in [8].

The inference rules are given in Table 1. They use a reference term $t_{ref} = g(x_1, \ldots, x_m)$, where $x_1, \ldots, x_m \in \mathcal{X}$ and $g \in \mathcal{D}$ (if g is a constant, then $t_{ref} = g$).

We generate the proof trees of $\mathcal{R}$ by applying, for each defined symbol $g \in \mathcal{D}$, the inference rules using the reference term $t_{ref} = g(x_1, \ldots, x_m)$ on the initial set of 3-tuples $\{(\{t_{ref} = g(x_1, \ldots, x_m)\}, \top, \top)\}$, with a specific strategy S, repeating the following steps: first, apply **Abstract**, and then try **Stop**. Then try all possible applications of **Narrow**. Then, try **Stop** again.

Let us clarify that if A is satisfiable, the transformed forms of A by **Abstract** and **Stop** are also satisfiable. Moreover, the first application of **Abstract** generates $A = (\bigwedge_i x_i\downarrow = X_i)$, always satisfied by the constructor constant supposed to exist in $\mathcal{F}$. Thus, with strategy S, it is useless to prove the satisfiability of A in the **Abstract** and **Stop** rules.

The process may not terminate if there is an infinite number of applications of **Abstract** and **Narrow** on the same branch of a proof tree. Nothing can be said in that case about IP-termination. The process stops if no inference rule applies

anymore. Then, when all branches of the proof trees end with an application of **Stop**, IP-termination is established.

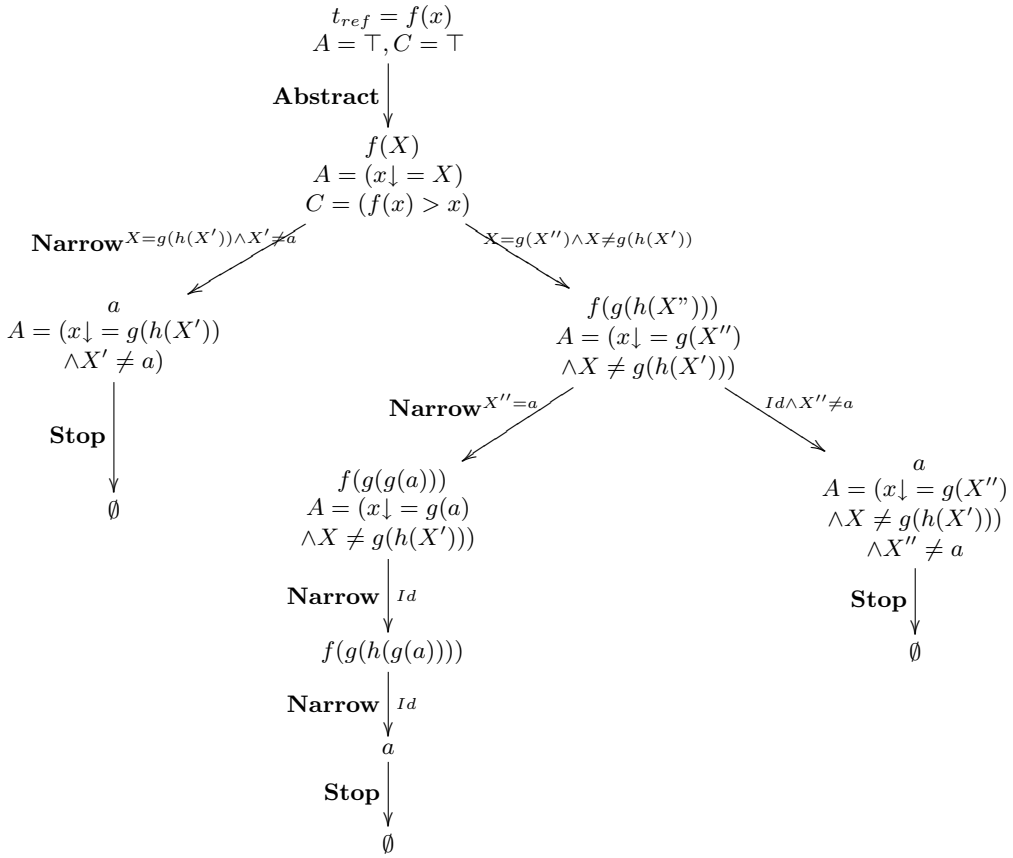
A finite proof tree is said to be *successful* if its leaves are states of the form $(\emptyset, A, C)$. We write $SUCCESS(g, \succ)$ if the application of S on $(\{g(x_1, \ldots, x_m)\}, \top, \top)$ gives a successful proof tree, whose sets C of ordering constraints are satisfied by the same ordering $\succ$.

Theorem 1. *Let $\mathcal{R}$ be a priority rewrite system on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ having at least one constructor constant. Every term of $\mathcal{T}(\mathcal{F})$ is IP-terminating iff there is a noetherian ordering $\succ$ such that for each symbol $g \in \mathcal{D}$, we have $SUCCESS(g, \succ)$.*

Example 2. Let us consider the following PRS, whose underlying RS is neither terminating, nor innermost terminating:

$$\left\downarrow \begin{array}{ll} f(g(h(x))) \to a & (1) \\ h(a) \to g(a) & (2) \\ f(g(x)) \to f(g(h(x))) & (3). \end{array} \right.$$

The proof tree of f is:



To lighten the figure, the sets A and C are not repeated on a branch, when they do not change. **Abstract** applies on $f(x)$ because the ordering constraint $f(x) > x$ is satisfiable by any noetherian ordering having the subterm property. Then, **Narrow** applies on $f(X)$ using Rules (1) and (3), according to Definition 7.

On the second branch, the term $f(g(h(X'')))$ narrows into $f(g(g(a)))$ with Rule (2) and $\sigma = (X'' = a)$, into a with Rule (1) and $\sigma = (Id \wedge X'' \neq a)$, but does not narrow with Rule (3): the negation of Id does not exist.

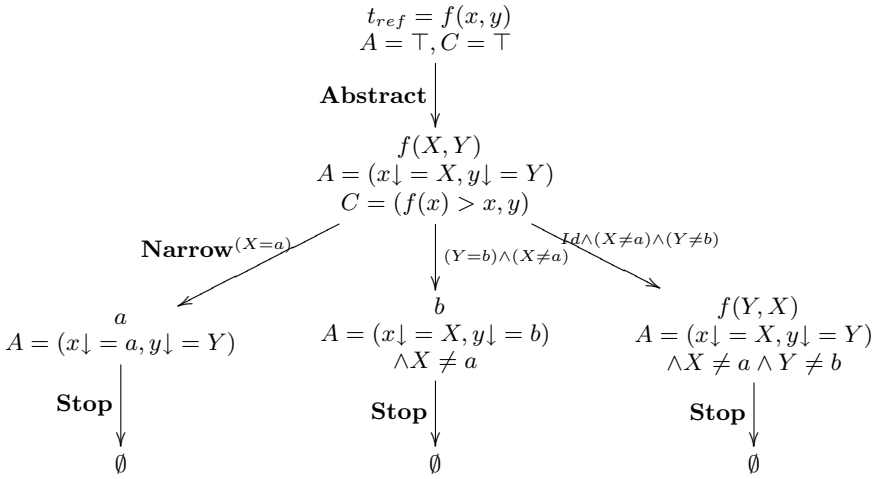
The set A after the **Abstract** step is trivially satisfied by the instantiation $\theta = (x = X = a)$. One can take $\theta = (x = g(h(g(a))), X' = g(a))$ for the next set A on the fist branch, $\theta = (x = g(a), X = X' = X'' = a)$ for the next set A on the second branch, and $\theta = (x = g(a), X = X' = a)$ for the last set A on the second branch.

In the proof tree of h, we just have an **Abstract**, a **Narrow** and a **Stop** step.

So the PRS is IP-terminating.

Example 3. Consider now the following PRS, which is also neither terminating nor innermost terminating:

$$\left\downarrow \begin{array}{ll} f(a,y)) \to a & (1) \\ f(x,b) \to b & (2) \\ f(x,y) \to f(y,x) & (3). \end{array}\right.$$



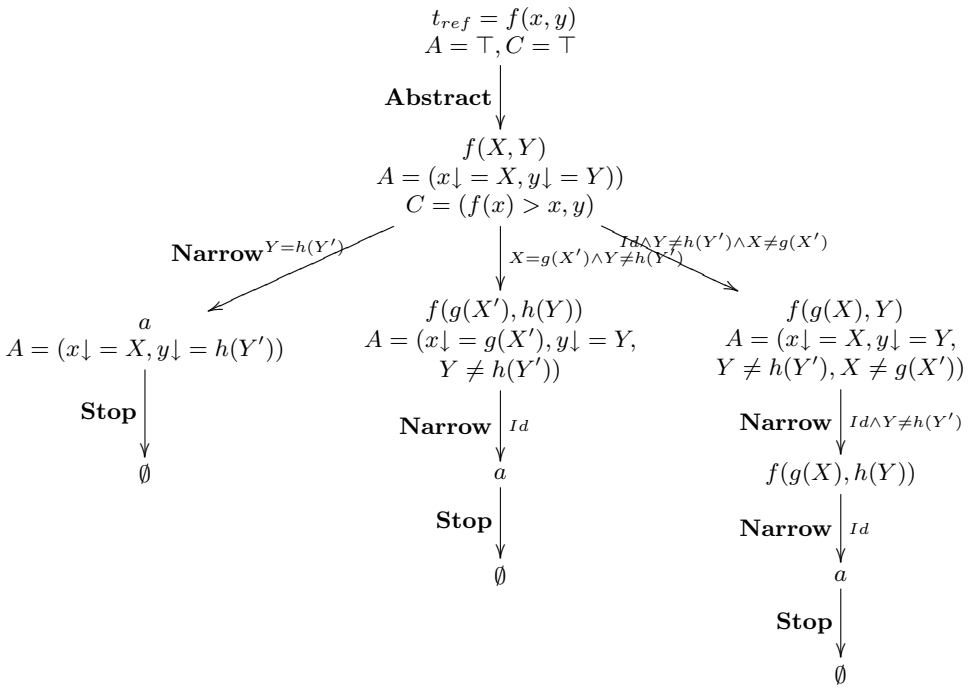
The rule **Stop** applies on $f(Y,X)$ because the term would only be narrowable with the substitution $\sigma = (Id \wedge Y \neq a \wedge X \neq b)$. But $A \wedge \sigma = (x\downarrow = X, y\downarrow = Y) \wedge X \neq a \wedge Y \neq b \wedge Y \neq a \wedge X \neq b)$ would be unsatisfiable: X and Y could only be of the form $f(X', Y')$, which is impossible since $f(X', Y')$ is always reducible.

The set A after the **Abstract** step is trivially satisfied by the instantiation $\theta = (x = X = y = Y = a)$. One can take $\theta = (x = y = Y = a)$ for the next set A on the fist branch, $\theta = (x = X = y = b)$ for the set A on the second branch, and $\theta = (x = X = b, y = Y = a)$ for the set A on the third branch.

Example 4. Let us consider the following PRS, whose underlying RS is also neither terminating, nor innermost terminating:

$$\left\downarrow \begin{array}{ll} f(x,h(y)) \to a & (1) \\ f(g(x),y) \to f(g(x),h(y)) & (2) \\ f(x,y) \to f(g(x),y) & (3). \end{array}\right.$$

The proof tree of the only defined symbol is:



Abstract applies on $f(x, y)$ because the ordering constraint $f(x) > x, y$ is satisfiable by any noetherian ordering having the subterm property. Then, **Narrow** applies on $f(X, Y)$ using Rules (1), (2), and (3), according to Definition 7.

On the second branch, the term $f(g(X'), h(Y))$ narrows into a with Rule (1) and $\sigma = Id$, so the other rules cannot apply: the negation of Id does not exist. This is coherent with the fact that Rule (1) has the highest priority, and applies for all possible instances of the term.

On the third branch, $f(g(X), Y)$ narrows into $f(g(X), h(Y))$ with Rule (2), but does not narrow with Rule (1), because the narrowing substitution would be $\sigma = (Y = h(Y'))$, and $A \wedge \sigma$ would be unsatisfiable. Indeed, we would have $Y \neq h(Y')$ and $Y = h(Y')$. The term $f(g(X), h(Y))$ does not narrow with Rule (3) either: it first narrows with Rule (2) and Id.

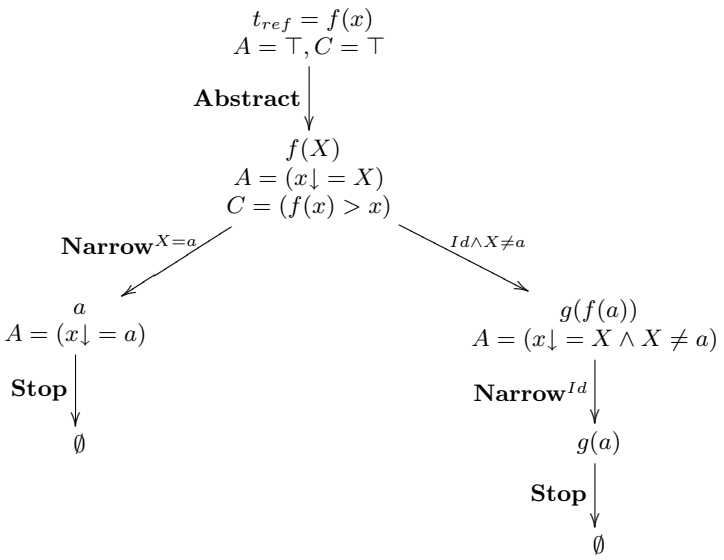
The set A after the **Abstract** step is trivially satisfied by the instantiation $\theta = (x = X = y = Y = a)$. One can take $\theta = (x = X = Y' = a, y = h(a))$ for the next set A on the fist branch, $\theta = (X' = y = Y = a, x = g(a))$ for the set A on the second branch, and $\theta = (x = X = y = Y = a)$ for the set A on the third branch.

So the PRS is IP-terminating.

Example 5. Let us consider the following PRS, whose underlying RS is also neither terminating, nor innermost terminating:

$$\left\downarrow \begin{array}{ll} f(a) \to a & (1) \\ f(x) \to g(f(a)) & (2). \end{array}\right.$$

The proof tree of f is:



Note that in the four examples above, the irreducible constant of the algebra, and more generally the constructors, can be used in an automatic way to find a solution of A.

6 Conclusion

In this paper, we have proposed an inductive method for proving termination of the decidable innermost priority rewriting relation of C.K. Mohan [11]. This work is an extension to priority rewriting of an inductive approach given in [8] for proving innermost termination of rewriting. We have generalized the innermost narrowing relation introduced in [8], to model the IP-rewriting relation on ground terms. A lifting lemma involving priorities on the rules establishes correctness of this modelization, and then of Theorem 1. For details, see the appendix.

Constraints are crucial in our approach: ordering constraints warrant the applicability of the induction principle, abstraction constraints define the ground terms considered at each step of the proof, and help to contain the narrowing mechanism. As precised in Section 4, automatable sufficient conditions can be used to deal with them, which allows our termination proof procedure to be completely automatic in many cases.

As termination of the original priority rewriting relation of [3] warrants a semantics for this relation, one can think that IP-termination warrants a semantics for the IP-rewriting relation. This has to be investigated. We also plan to generalize our technique to the termination proof of other priority rewriting relations.

References

1. Lennart Augustsson. A compiler for lazy ml. In *LFP '84: Proceedings of the 1984 ACM Symposium on LISP and functional programming*, pages 218–227, New York, NY, USA, 1984. ACM.
2. Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, New York, NY, USA, 1998.
3. J. C. M. Baeten, J. A. Bergstra, J. W. Klop, and W. P. Weijland. Term-rewriting systems with rule priorities. *Theoretical Computer Science*, 67(2-3):283–301, 1989.

4. Jos C. M. Baeten, Jan A. Bergstra, and Jan Willem Klop. Term rewriting systems with priorities. In *Proceedings of the 2nd International Conference on Rewriting Techniques and Applications*, volume 256 of *Lecture Notes in Computer Science*, pages 83–94. Springer Verlag, 1987.
5. Home of Clean. http://clean.cs.ru.nl/index.html.
6. N. Dershowitz and D.A. Plaisted. Rewriting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 9, pages 535–610. Elsevier Science, 2001.
7. Isabelle Gnaedig. Termination of Priority Rewriting - Extended Version. Technical report, LORIA, 2007. Available at http://www.loria.fr/∼gnaedig/PAPERS/-REPORTS/IP-termin-extended.pdf.
8. Isabelle Gnaedig and Hélène Kirchner. Termination of Rewriting under Strategies. *ACM Transactions on Computational Logic*, 2007. To appear. Preliminary version available at http://www.loria.fr/∼gnaedig/PAPERS/REPORTS/S-termin-2007-preli.pdf.
9. Wiki homepage of Haskell. http://www.haskell.org/haskellwiki/Haskell.
10. J. B. Kruskal. Well-quasi ordering, the tree theorem and Vazsonyi's conjecture. *Trans. Amer. Math. Soc.*, 95:210–225, 1960.
11. Chilukuri K. Mohan. Priority rewriting: Semantics, confluence, and conditional. In *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications*, volume 355 of *Lecture Notes in Computer Science*, pages 278–291. Springer Verlag, 1989.
12. Rinus Plasmeijer and Marko van Eekelen. *Functional Programming and Parallel Graph Rewriting*. Addison Wesley, 1993.
13. Masahiko Sakai and Yoshihito Toyama. Semantics and strong sequentiality of priority term rewriting systems. *Theoretical Computer Science*, 208(1–2):87–110, 1998.
14. Terese. *Term Rewriting Systems*. Number 55 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.
15. Jaco van de Pol. Operational semantics of rewriting with priorities. *Theoretical Computer Science*, 200(1-2):289–312, 1998.

Appendix

This appendix contains the proof of the lifting lemma and of the theorem.

A The lifting lemma

For the proof of Theorem 1, we need the following lifting lemma.

Lemma 1 (Priority Innermost 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 IP-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 $\alpha s \rightarrow^{IP}_{p,l\rightarrow r} t'$, then there exist a term $s' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and substitutions $\beta, \sigma = \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j} \bigwedge_{i\in[1..n]} \overline{\sigma_0^i}$. such that:*

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

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 suffix position p' of p in s, and $\sigma_0^1, \ldots, \sigma_0^n$ are the most general unifiers of $s|_p$ with the left-hand sides of the rules having a greater priority than $l \rightarrow r$.

For the proof of the lemma, we need the two following propositions.

Proposition 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 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 $(\mu\sigma) = (\nu\sigma)[B]$. □

Proof (of Lemma 1).
In the following, we assume that $\mathcal{Y} \cap \mathcal{V}ar(l) = \emptyset$ for every $l \rightarrow r \in \mathcal{R}$.
If $\alpha s \rightarrow^{IP}_{p,l\rightarrow r} t'$, then there is 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:
$\mu(s|_p) = \alpha(s|_p)$ for $Dom(\tau) \subseteq \mathcal{V}ar(l)$ and $\mathcal{V}ar(l) \cap \mathcal{V}ar(s) = \emptyset$
$= \tau l$ by definition of τ
$= \mu l$ for $Dom(\alpha) \subseteq \mathcal{Y}$ and $\mathcal{Y} \cap \mathcal{V}ar(l) = \emptyset$,

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 is 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 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 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 suffix position p' of 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 is $i \in [1..k]$ 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 $l' \to r'$, which is impossible by definition of innermost reducibility of $\alpha(s)$ at position p. 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 suffix positions of p.

Let us now suppose that there exist a rule $l' \to r' \in \mathcal{R}$ of higher priority than $l \to r$ and a substitution σ_0^i such that $\sigma_0^i(s|_p) = \sigma_0^i l'$. With a similar reasoning than previously, we get that $\alpha(s)$ is reducible at position p with the rule $l' \to r'$, which has higher priority than $l \to r$. This is impossible by definition of IP-reducibility of $\alpha(s)$ by $l \to r$ at position p. So the ground substitution β also satisfies $\bigwedge_{i\in[1..n]} \overline{\sigma_0^i}$ where $\sigma_0^1, \ldots, \sigma_0^n$ are the most general unifiers of $s|_p$ with the left-hand sides of rules having a greater priority than $l \to r$ (4).

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

□

B The *IP*-termination theorem

Theorem 1. *Let $\mathcal{R}$ be a priority rewrite system on $\mathcal{T}(\mathcal{F},\mathcal{X})$ having at least one constructor constant. Every term of $\mathcal{T}(\mathcal{F})$ is IP-terminating iff there is 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 IP-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.

The rule **Abstract** applies to give $f(X_1,\ldots,X_m), X_1,\ldots,X_m \in \mathcal{N}$. Indeed, by hypothesis, we have $IPT(x_i)$. Then **Stop** applies, because we also have $IPT(f(X_1,\ldots,X_m)$. 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 prove by induction on $\mathcal{T}(\mathcal{F})$ that any ground instance $\theta f(x_1,\ldots,x_m)$ IP-terminates for any term $f(x_1,\ldots,x_m) \in \mathcal{T}(\mathcal{F},\mathcal{X})$ with $f \in \mathcal{F}$. We use an abstraction lemma, a narrowing lemma, and a stopping lemma, which are given after this main proof.

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 $g(\theta x_1,\ldots,\theta x_m)$ is IP-terminating for any θ satisfying $A = \top$ if we have $SUCCESS$ $(h,\succ)$ for every defined symbol h. 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 state s 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 represented by s. When t is a reducible constant, the set of ground instances is reduced to t itself.

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

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

When the **Stop** inference rule applies on $(\{t\},A,C)$, by stopping lemma, every term of $G = \{\alpha t \mid \alpha \text{ satisfies } A\}$ is IP-terminating. Therefore, IP-termination is ensured for all terms in all sets G in the proof tree.

As the process is initialized with $\{t_{ref}\}$ and a set A of abstraction constraints satisfiable by any ground substitution, we get that $g(\theta x_1,\ldots,\theta x_m)$ is IP-terminating, for any $t_{ref} = g(x_1,\ldots,x_m)$, and any ground instance θ.

If f is a constructor, either it is a constant, which is irreducible, and then IP-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 just an application of **Abstract** and **Stop**. Indeed, $f(x_1, \ldots, x_m)$ always abstracts into $f(X_1, \ldots, X_m)$. Then **Stop** applies because $f(X_1, \ldots, X_m)$ is not narrowable and all its variables are in $\mathcal{N}$.

□

Lemma 2 (Abstraction lemma). *Let $(\{t\}, A, C)$ be a state of any proof tree, giving the state $(\{t' = t[X_j]_{j\in\{i_1,\ldots,i_p\}}\}, A', C')$ by application of* **Abstract**.

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

Proof. We prove that $\alpha t \xrightarrow{*}_S \beta t'$, where $\beta = \alpha \cup \bigcup_{j\in\{i_1,\ldots,i_p\}} X_j = \alpha t|_j\downarrow$.

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

- either $IPT(t|_j)$ is true, and then by definition of the predicate IPT, $\alpha t|_j$ is IP-terminating;
- or $t_{ref} > t|_j$ is satisfiable by $\succ$, and then, by induction hypothesis, $\alpha t|_j$ is IP-terminating.

So, $\alpha t|_j$ reduces to an IP-normal form $\alpha t|_j\downarrow$. Then, whatever the positions $i_1, \ldots, i_p$ in the term t, we have $\alpha t \xrightarrow{*}^{IP} \alpha t[\alpha t|_{i_1}\downarrow]_{i_1} \ldots [\alpha t|_{i_p}\downarrow]_{i_p} = \beta t'$.

Thus, $\alpha t \xrightarrow{*}^{IP} \beta t'$ 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 IP-termination of $\beta t'$ implies IP-termination of αt.

Finally, , β satisfies $A' = A \wedge t|_{i_1}\downarrow = X_{i_1} \ldots \wedge t|_{i_p}\downarrow = X_{i_p}$, 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 3 (Narrowing lemma). *Let $(\{t\}, A, C)$ be a state of any proof tree, giving the states $(\{v_i\}, A'_i, C'_i), i \in [1..l]$, by application of* **Narrow**. *For any ground substitution α satisfying A, if αt is IP-reducible, then, for each $i \in [1..l]$, there is β_i such that IP-termination of the $\beta_i v_i, i \in [1..l]$, implies IP-termination of αt. Moreover, β_i satisfies A'_i for each $i \in [1..l]$.*

Proof. For any rewriting step $\alpha t \rightarrow^{IP}_{p,l\rightarrow r} t'$, by Lifting Lemma, there is a term $v \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and substitutions $\beta, \sigma = \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j} \wedge \bigwedge_{i\in[1..n]} \overline{\sigma_0^i}$ such that:

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

where σ_0 is the most general unifier of $t|_p$ and l, $\sigma_j, j \in [1..k]$ are all the most general unifiers of $\sigma_0 t|_{p'}$ and a left-hand side l' of a rule of $\mathcal{R}$, for all suffix positions p' of p in s, $\sigma_0^i, i \in [1..n]$ are all the most general unifiers of $t|_p$ with the left-hand sides of the rules having greater priority than $l \rightarrow r$.

The narrowing steps are effectively produced in the proof tree by the **Narrow** rule, applied in all possible ways on t. Then, the narrowing step $t \rightsquigarrow^{IP}_{p,l\rightarrow r,\sigma} v$ is produced. So a term βv is produced for every IP-rewriting branch starting from αt. Then IP-termination of the βv implies IP-termination of αt.

Let us prove that β satisfies $A' = A \wedge \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j} \wedge \bigwedge_{i\in[1..n]} \overline{\sigma_0^i}$.

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 $A' = A \wedge \sigma_0 \wedge \bigwedge_{j\in[1..k]} \overline{\sigma_j} \wedge \bigwedge_{i\in[1..n]} \overline{\sigma_0^i}$.

□

Lemma 4 (Stopping lemma). *Let $(\{t\}, A, C)$ be a state of any proof tree, with A satisfiable, and giving the state $(\emptyset, A', C')$ by application of an inference rule. Then for every ground substitution α satisfying A, αt is IP-terminating.*

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

- either $IPT(t)$ and then αt is IP-terminating 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 is IP-terminating.

□