A proof of weak termination providing the right way to terminate

Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner

A proof of weak termination providing the right way to terminate *

Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner

LORIA-INRIA & LORIA-CNRS
BP 239 F-54506 Vandœuvre-lès-Nancy Cedex
Phone: + 33 3 83 58 17 00
Fax: + 33 3 83 27 83 19
e-mail: fissore@loria.fr, gnaedig@loria.fr, Helene.Kirchner@loria.fr

Abstract. We give an inductive method for proving weak innermost termination of rule-based programs, from which we automatically infer, for each successful proof, a finite strategy for data evaluation. We first present the proof principle, using an explicit induction on the termination property, to prove that any input data has at least one finite evaluation. For that, we observe proof trees built from the rewrite system, schematizing the innermost rewriting tree of any ground term, and generated with two mechanisms: abstraction, schematizing normalization of subterms, and narrowing, schematizing rewriting steps. Then, we show how, for any ground term, a normalizing rewriting strategy can be extracted from the proof trees, even if the ground term admits infinite rewriting derivations.

1 Introducing the problem

In the context of programming in general, termination is a key property that warrants the existence of a result for every evaluation of a program. For rule-based programs, written in languages like ASF+SDF [19], Maude [4], Cafe-OBJ [12], or ELAN [3], data evaluation consists in exploring rewriting derivations of an input term. Strong termination, expressing that every rewriting derivation terminates, often does not hold. When for any term, there is at least one terminating derivation, the rewrite system is said to be weakly terminating. This is an interesting property for languages like ELAN , whose strategies can express that the result of the program evaluation on a data is *one of its possible* finite evaluations, or *the first* one. Weak termination then warrants a result for such evaluation strategies.

Analyzing termination also allows choosing the good way to evaluate data. Indeed, if the program is strongly terminating, a depth-first evaluation can be used, while if the program is only weakly terminating, a breadth-first algorithm, often much more costly, is necessary in general. In the second case, if there is

* ——- The original publication is available at www.springerlink.com —— https://link.springer.com/chapter/10.1007/978-3-540-31862-0_26

a way to find terminating branches, the breadth-first technique can be avoided, which yields a considerable gain for program executions. This is what we propose.

Specific methods for proving termination of rewriting under strategies have been studied. Let us cite [2] and [13, 9] for the innermost strategy, [10] for the outermost strategy, and [8, 20] for local strategies on operators. All these works tackle the problem of strong termination. Here, we consider the weak innermost termination problem, i.e. we prove that among all innermost rewriting derivations starting from any term, one of them is finite. We focus on the innermost rewriting strategy, consisting in rewriting always at the lowest possible positions, since it is most often used as a built-in mechanism in evaluation of rule-based languages and functional languages.

Like the previously cited methods, the approach presented here also gives a way to prove weak termination of standard rewriting. But to our knowledge, it is the only approach able to handle term rewriting systems (TRSs in short) that are not strongly but only weakly innermost terminating. Moreover, our method is *constructive* in the sense that the proof gives the strategy to follow to obtain one of the finite derivations.

The weak termination property has been studied from several perspectives. For instance, B. Gramlich proved that weak termination can imply strong termination [16]. He also established conditions on TRSs for the property to be preserved by the union operation on TRSs [17]. J. Goubault-Larrecq proposed a proof of weak termination of typed Lambda-Sigma calculi in [15].

In order to illustrate the main ideas of our method on a running example, let us consider the following TRS:

$$f(g(x), s(0)) \to f(g(x), g(x)) \tag{1}$$
$$f(g(x), s(y)) \to f(h(x, y), s(0)) \tag{2}$$
$$g(s(x)) \to s(g(x)) \tag{3}$$
$$g(0) \to 0 \tag{4}$$
$$h(x, y) \to g(x). \tag{5}$$

Obviously, $\mathcal{R}$ is not terminating, nor even, because of the rule (2), innermost terminating. For instance, the following innermost infinite sequence is possible in $\mathcal{R}$: $\underline{f(g(f(0,0)), s(0))} \to^{(2)} f(\underline{h(f(0,0),0)}, s(0)) \to^{(5)} \underline{f(g(f(0,0)), s(0))} \ldots$ However, $\mathcal{R}$ is weakly innermost terminating ; in particular, the cycle above can be avoided by using the rule (1) instead of (2).

We first propose in this paper a method based on the same inductive principle as [9, 8, 10], where we study strong termination: we use an explicit induction on the termination property, but to prove here that every element t of a given set of terms T weakly innermost terminates, i.e. there is at least one finite innermost rewriting derivation starting from t. The general proof principle relies on the simple idea that for establishing weak innermost termination of a ground term t, it is enough to suppose that subterms of t weakly innermost terminate, and that rewriting the context leads to at least one terminating chain. Iterating this process until a non-reducible context is obtained establishes weak innermost termination of t.

Directly using the termination notion on terms has also been proposed in [14], for inductively proving well-foundedness of binary relations, among which path orderings. The approach differs from ours in that it works on general relations, that can then be used on TRSs, whereas we directly handle the termination proof of a given TRS.

From the proof of weak termination of a given TRS, we then extract for any given ground term, a rewriting strategy to compute one of its normal form, even if the ground term admits infinite rewriting derivations. To some extent, our method has similarities with [18], where an automaton is built for normalization according to a needed-redex strategy in the case of orthogonal rewrite systems.

In Section 2, the background is presented. Section 3 introduces the basic concepts of the inductive proof mechanism. In Section 4, our method is formally described with inference rules and a strategy to apply them. Finally, in Section 5, a strategy is proposed to reach an innermost normal form from a given term, using information of the proof establishing weak termination.

2 Notations

We assume that the reader is familiar with the basic definitions and notations of term rewriting given for instance in [7]. $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the set of terms built from a given finite set $\mathcal{F}$ of function symbols having an arity $n \in \mathbb{N}$, and a set $\mathcal{X}$ of variables denoted $x, y \ldots$. $\mathcal{T}(\mathcal{F})$ is the set of ground terms (without variables). The terms composed of a symbol of arity 0 are called constants; $\mathcal{C}$ is the set of constants of $\mathcal{F}$. Positions in a term are represented as sequences of integers. The empty sequence ϵ denotes the top position. The notation $t|_p$ stands for the subterm of t at position p. The term $u[t_j]_{j\in\{i_1..i_k\}}$ denotes the term u in which the subterms $u|_j$ have been replaced by t_j respectively.

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 $\mathcal{T}(\mathcal{F},\mathcal{X})$. We identify a substitution $\sigma = (x \mapsto t) \ldots (y \mapsto u)$ with the finite conjunction of equations $(x = t) \wedge \ldots \wedge (y = u)$. The result of applying σ to a term $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ is written $\sigma(t)$ or σt. The domain of σ, denoted $Dom(\sigma)$ is the finite subset of $\mathcal{X}$ such that $\sigma x \neq x$. A ground substitution or instantiation is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F})$. The composition of substitutions σ_1 followed by σ_2 is denoted $\sigma_2\sigma_1$.

Given a set $\mathcal{R}$ of rewrite rules or term rewriting system on $\mathcal{T}(\mathcal{F},\mathcal{X})$, a function symbol in $\mathcal{F}$ is called a constructor if it does not occur in $\mathcal{R}$ at the top position of the left-hand side of a rule, and is called a defined function symbol otherwise. The set of constructors of $\mathcal{F}$ for $\mathcal{R}$ is denoted by $\mathcal{C}ons_{\mathcal{R}}$, the set of defined function symbols of $\mathcal{F}$ for $\mathcal{R}$ is denoted by $\mathcal{D}ef_{\mathcal{R}}$ ($\mathcal{R}$ is omitted when there is no ambiguity). The rewriting relation induced by $\mathcal{R}$ is denoted by $\rightarrow_{\mathcal{R}}$ ($\rightarrow$ if there is no ambiguity on $\mathcal{R}$). We note $s \rightarrow_{p,l\rightarrow r,\sigma} t$ (or $s \rightarrow^{p,l\rightarrow r,\sigma} t$ where either p or $l \rightarrow r$ or σ may be omitted) if s rewrites into t at position p with the rule $l \rightarrow r$ and the substitution σ, i.e. $s = s[l\sigma]_p$ and $t = s[r\sigma]_p$. The reflexive transitive closure of the rewriting relation induced by $\mathcal{R}$ is denoted by $\rightarrow_R^*$. Given a term

t, we call normal form of t, denoted by $t{\downarrow}$, any irreducible term, if it exists, such that $t \rightarrow_R^* t{\downarrow}$.

An ordering $\succ$ on $\mathcal{T}(\mathcal{F},\mathcal{X})$ is said to be noetherian iff there is no infinite decreasing derivation (or chain) for this ordering. It is $\mathcal{F}$-stable iff for any pair of terms t, t' of $\mathcal{T}(\mathcal{F},\mathcal{X})$, for any context $f(\ldots\ \ldots)$, $t \succ t'$ implies $f(\ldots t \ldots) \succ f(\ldots t' \ldots)$. It has the subterm property iff for any t of $\mathcal{T}(\mathcal{F},\mathcal{X})$, $f(\ldots t \ldots) \succ t$. Notice that, for $\mathcal{F}$ and $\mathcal{X}$ finite, if $\succ$ is $\mathcal{F}$-stable and has the subterm property, then it is noetherian [6]. If, in addition, $\succ$ is stable by substitution (for any substitution σ, any pair of terms $t, t' \in \mathcal{T}(\mathcal{F},\mathcal{X})$, $t \succ t'$ implies $\sigma t \succ \sigma t'$), then it is called a simplification ordering. Let t be a term of $\mathcal{T}(\mathcal{F})$; like for standard rewriting, we say that t weakly (resp. strongly) (innermost) terminates if and only if at least one (resp. every) (innermost) rewriting derivation starting from t is finite. Obviously, strong (innermost) termination implies weak (innermost) termination. An innermost rewriting normal form of t is also denoted by $t{\downarrow}$, when there is no ambiguity.

3 Induction and constraints

For proving that the terms t of $\mathcal{T}(\mathcal{F})$ weakly innermost terminate, we proceed by induction on $\mathcal{T}(\mathcal{F})$ with a noetherian ordering $\succ$, assuming that for any t' such that $t \succ t'$, t' weakly innermost terminates. To warrant non emptiness of $\mathcal{T}(\mathcal{F})$, we assume that $\mathcal{F}$ contains at least a constructor constant.

The main intuition is to observe the rewriting derivation tree starting from any ground term $t \in \mathcal{T}(\mathcal{F})$ which is any instance of a term $g(x_1,\ldots,x_m) \in \mathcal{T}(\mathcal{F},\mathcal{X})$, for some defined function symbol $g \in \mathcal{D}ef$, and variables $x_1,\ldots,x_m$. Proving weak innermost termination on ground terms amounts to prove that all these rewriting derivation trees have at least one finite branch.

Each rewriting derivation tree is 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 $g \in \mathcal{D}ef$, by alternatively using two main concepts: narrowing and abstraction. More precisely, narrowing schematizes all innermost rewriting possibilities of terms. The abstraction process simulates the innermost normalization of subterms in the derivations. It consists in replacing these subterms by special variables, denoting one of their possible innermost normal forms, without computing them. This abstraction step is performed on subterms that can be assumed weakly innermost terminating by induction hypothesis.

The schematization of ground rewriting derivation trees 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 a set of ground terms: all 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 rele-

vant sets of ground terms. The latter actually allow controlling the narrowing process, well known to easily diverge.

Unlike [9, 8, 10], where, for proving strong termination, all branches of the proof trees have to be considered, we only develop here the relevant branches that warrant termination of one rewriting derivation for any ground term.

We now introduce the necessary concepts to formalize and automate the technique sketched above.

3.1 Ordering constraints and abstraction

The induction ordering $\succ$ is constrained along the proof by imposing constraints 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 $t \succ_{\mathcal{P}} u$ induces $\theta t \succ \theta u$, for every θ solution of the constraint associated to u.

So inequalities of the form $t > u_1, \ldots, u_m$ are accumulated, which are called *ordering constraints*. Any ordering $\succ_{\mathcal{P}}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X})$ satisfying them and which is stable by substitution fulfills the previous requirement on ground terms. The ordering $\succ_{\mathcal{P}}$, defined on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, can then be seen as an extension of the induction ordering $\succ$, defined on $\mathcal{T}(\mathcal{F})$. For convenience, $\succ_{\mathcal{P}}$ is also written $\succ$.

It is important to remark that, for establishing the inductive termination proof, it is sufficient to decide whether there exists such an ordering.

Definition 1 (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 C of this form is undecidable. But a sufficient condition for an ordering $\succ$ to satisfy C is that $\succ$ is stable by substitution and $t \succ t'$ for any constraint $t > t'$ of C.

Other constraints are introduced by the abstraction mechanism. To abstract a term u at positions $i_1, \ldots, i_p$, where the $u|_j$ are supposed to have a normal form $u|_j{\downarrow}$, we replace the $u|_j$ by abstraction variables X_j representing respectively one of their possible innermost normal forms. Let us define these special variables more formally.

Definition 2 (NF-variable). *Let $\mathcal{N}$ be a set of new variables disjoint from $\mathcal{X}$. Symbols of $\mathcal{N}$ are called* NF-variables*. Substitutions and instantiations are extended to $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$ in the following way. Let $X \in \mathcal{N}$; for any substitution σ (resp. instantiation θ) such that $X \in Dom(\sigma)$, σX (resp. θX) is in normal form, and then $\mathcal{V}ar(\sigma X) \subseteq \mathcal{N}$.*

Definition 3 (term abstraction). *The term u is said to be* abstracted *into the term u' (called* abstraction *of u) at positions $\{i_1, \ldots, i_p\}$ iff $u' = u[X_j]_{j\in\{i_1,\ldots,i_p\}}$, where the $X_j, j \in \{i_1, \ldots, i_p\}$ are fresh distinct NF-variables.*

Weak termination on $\mathcal{T}(\mathcal{F})$ is proved by reasoning on terms with abstraction variables, i.e. on terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$. Ordering constraints are extended to pairs of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$. When subterms t_i are abstracted by X_i, 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_i. 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.

3.2 Narrowing

After abstraction of 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 $t[X_j]_{j\in\{i_1,\ldots,i_p\}}$.

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 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 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 TRS $R = \{g(a) \to a, f(g(x)) \to b\}$, 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 $\bigvee_i (x_i \neq t_i)$.

Definition 4. *A substitution σ is said* to satisfy *a constraint $\bigwedge_j \bigvee_{i_j} (x_{i_j} \neq t_{i_j})$, iff for all 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 5 (innermost narrowing). *A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$* innermost narrows *into a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$ at the non-variable position p, 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 and l at position p, and $\sigma_j, j \in [1..k]$ are all most general unifiers of $\sigma_0 t$ and a left-hand side of rule of $\mathcal{R}$, at suffix positions of p.

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.

3.3 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 is applied to a current term u if the narrowing substitution σ effectively corresponds to a rewriting step of ground instances of u, i.e. if σ is *compatible* with the abstraction constrained formula A associated to u (i.e. σA is satisfiable). Else, the narrowing step is useless. So the narrowing constraint attached to the narrowing step is added to the abstraction constraints initially of the form $t\downarrow = X$. This motivates the introduction of abstraction constrained formulas.

Definition 6. *An* abstraction constrained formula *(ACF in short) is a formula* $\bigwedge_i (t_i\downarrow = t'_i) \wedge \bigwedge_j \bigvee_{k_j} (u_{k_j} \neq v_{k_j})$*, where* $t_i, t'_i, u_{k_j}, v_{k_j} \in \mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$.

Definition 7. *An* abstraction constrained formula $A = \bigwedge_i (t_i\downarrow = t'_i) \wedge \bigwedge_j \bigvee_{k_j} (u_{k_j} \neq v_{k_j})$ *is* satisfiable *iff there exists at least one instantiation θ such that* $\bigwedge_i (\theta t_i\downarrow = \theta t'_i) \wedge \bigwedge_j \bigvee_{k_j} (\theta u_{k_j} \neq \theta v_{k_j})$*. The instantiation θ is then said to satisfy the ACF A and is called solution of A.*

Applying a constrained substitution $\sigma = \sigma_0 \wedge \bigwedge_j \bigvee_{i_j} (x_{i_j} \neq t_{i_j})$ to an ACF A gives a formula σA obtained by applying σ_0 to A and then by adjoining the disequality part to the result.

An ACF A is attached to each term u in the proof trees; its solutions characterize the interesting ground instances of this term, that are the θu such that θ is a solution of A. When A has no solution, the current node of the proof tree does not represent any ground term. Such nodes are then irrelevant for the weak termination proof. So we have the choice between generating only the relevant nodes of the proof tree, by testing satisfiability of A at each step, or stopping the proof on a branch on an irrelevant node, by testing unsatisfiability of A. These are both facets of the same question, but in practice, they lead to different solutions.

Checking 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, qfef in short), whose satisfiability has been addressed in [5], that provides rules to transform any disunification problem into a solved form. Testing 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.

Unsatisfiability of A is also undecidable in general, but simple sufficient conditions can be used, very often applicable in practice. They rely on reducibility, unifiability, narrowing and constructor tests, and can be found in [11].

So both satisfiability and unsatisfiability checks need to use sufficient conditions. But in the first case, the proof process stops with failure as soon as 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. In the approach followed below, narrowing and abstraction are applied without checking the satisfiability of abstraction constraints, and the process stops as soon as they are detected to be unsatisfiable.

4 Inference rules for inductive termination proofs

We are now ready to describe the different steps of our mechanism on a term t, with initial empty constraints conjunctions A, C. It consists in iterating the three following steps.

The first step abstracts the current term u at given positions $i_1, \ldots, i_p$. The constraints $t > u|_{i_1}, \ldots, u|_{i_p}$ are set, allowing to suppose, by induction, the existence of irreducible forms for $u|_{i_1}, \ldots, u|_{i_p}$. Then, $u|_{i_1}, \ldots, u|_{i_p}$ are abstracted into abstraction variables $X_{i_1}, \ldots, X_{i_p}$ (or $X_1, \ldots, X_p$ for simplifying the indices). The abstraction constraint $u|_{i_1}\downarrow = X_1, \ldots, u|_{i_p}\downarrow = X_p$ is added to the ACF A. This is the *abstract* step. The abstraction positions are chosen so that the abstraction mechanism captures the greatest possible number of rewriting steps: we abstract the greatest possible subterms of $u = f(u_1, \ldots, u_m)$. Note also that it is not useful to abstract non narrowable subterms: their ground instances are always in normal form, since the variables of these subterms are NF-variables.

The second step innermost narrows the resulting term in one step with all possible rewrite rules of the rewrite system $\mathcal{R}$, and all possible substitutions σ, into terms v, according to Definition 5. This step is a branching step, creating as many states as narrowing possibilities. The substitution σ is integrated to A, as explained after Definition 7. 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 weakly innermost terminating, which happens when the induction hypothesis applies on it.

The previously presented steps are performed by inference rules that transform 3-tuples (T, A, C) where T is a set of terms of $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{N})$, containing the current term whose weak innermost termination has to be proved: this is either a singleton or the empty set, A is an ACF and C is a conjunction of ordering constraints stated by the abstract steps.

Before giving the corresponding inference rules, let us notice that the inductive reasoning can be completed in the following way. When the induction

hypothesis cannot be applied on a term u, it is sometimes possible to prove weak innermost termination of every ground instance of u by another way. Let $WT(u)$ be a predicate that is true iff every ground instance of u weakly innermost terminates. In the first (resp. third) previous step of the induction reasoning, we then associate the alternative predicate $WT(u|_{i_j})$ (resp. $WT(u)$) to the condition $t > u|_{i_j}$ (resp. $t > u$). For establishing that $WT(u)$ is true, in some cases, the notion of usable rules [1] can be used. This approach is fully developed in [13].

Table 1. Inference rules for the weak innermost termination proof

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

where u is abstracted into u' at the positions $i_1, \ldots, i_p \neq \epsilon$
if $C \wedge H_C(u|_{i_1}) \ldots \wedge H_C(u|_{i_p})$ is satisfiable
where $H_C(u|_j)_{j \in \{i_1,\ldots,i_p\}} = \begin{cases} \mathit{true} & \text{if } WT(u|_j) \\ t_{ref} > u|_j & \text{otherwise.} \end{cases}$

Narrow: $$\frac{\{u\},\quad A,\quad C}{\{v\},\quad \sigma A,\quad C} \text{ if } u \leadsto^{Inn}_{\sigma} v$$

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

if $(C \wedge H_C(u))$ is satisfiable or **A is unsatisfiable**
where $H_C(u) = \begin{cases} \mathit{true} & \text{if } WT(u) \text{ or } \textbf{A is unsatisfiable} \\ t_{ref} > u & \text{otherwise.} \end{cases}$

The termination proof procedure is described by the set of rules given in Table 1. These rules must be applied on the initial pairs $(\{t_{ref} = g(x_1, \ldots, x_m)\}, \top, \top)$, where g is a defined symbol, with the strategy S

$$(\textbf{Abstract}; \text{dk}(\textbf{Narrow}); \textbf{Stop})\ ^*$$

where ";" denotes the sequential application of rules, "dk" the application of a rule in all possible ways and "*" the iterative application of a strategy, until it is not possible anymore. The process stops if no inference rule applies anymore.

There are two cases for the behavior of the termination proof procedure. The strategy applied to the initial state $(\{t_{ref}\}, \top, \top)$ terminates if the rules do not apply anymore and all states are of the form $(\emptyset, A, C)$. Otherwise, the strategy does not terminate if there is an infinite number of applications of **Abstract** and **Narrow**.

A branch of the derivation tree is said to be successful if it is ended by an application of **Stop**, i.e. if its final state is of the form $(\emptyset, A, C)$.

Thus, the inductive weak termination proof is successful if there is at least one successful branch corresponding to each possible ground term. Let us develop this point.

In fact, branching, produced by **Narrow**, can generate different states with narrowing substitutions $\sigma_1, \ldots \sigma_n$. These substitutions can be compared (see [11]). For σ_i and σ_j, three situations may occur: σ_i is strictly less general than σ_j, which is noted $\sigma_i > \sigma_j$, (or σ_j is strictly less general than σ_i), σ_i and σ_j are equal up to a renaming, or else σ_i and σ_j are incomparable.

States corresponding to substitutions that are more general than other ones then represent a set of ground instances that contains the other ones. So, for proving weak termination for all ground instances at a branching point, it is sufficient to prove weak termination only for the "most general states".

Note that the ignored states may schematize different rewriting steps than those we consider (at different positions, with different rewrite rules). So for the considered instances, if a "most general state" doesn't exclusively give rise to successful branches, we lose the possibility to test whether the other branches are successful. In practice, this case rarely occurs and the gain is greater in avoiding to consider redundant subsets of instances.

A branching node in a proof tree is a state, on which the Narrow rule applies. Let Σ be the set of narrowing substitutions (possibly with different rewrite rules) at a given branching node. Let Σ_0 be the reduced set from Σ such that $\sigma \in \Sigma_0$ iff $\sigma \in \Sigma$ and $\nexists\, \sigma' \in \Sigma$ such that $\sigma > \sigma'$ on $(Dom(\sigma) \setminus Var(l)) \cup (Dom(\sigma') \setminus Var(l'))$, where l and l' are the left-hand sides of rules respectively used to produce the narrowing substitutions σ and σ'. The set Σ_0 may yet contain equivalent (equal up to a renaming) substitutions which are marked as such. So for any two substitutions in Σ_0, either they are equivalent, or they are incomparable.

A proof tree is *weakly successful* if it is reduced to a state of the form $(\emptyset, A, C)$, or if at each branching node:

- for each class of equivalent substitutions, there exists at least one weakly successful subtree corresponding to a substitution in this class,
- all subtrees corresponding to incomparable substitutions are weakly successful.

So the strategy S can be optimized as follows: at each branching point of a proof tree, with set of substitutions Σ, we only develop the subtrees corresponding to Σ_0. Moreover, given two subtrees corresponding to equivalent substitutions, as soon as one of them is weakly successful, the other one is cut.

We write $SUCCESS(g, \succ)$ if the proof tree obtained by application on $(\{g(x_1, \ldots, x_m)\}, \top, \top)$, with the strategy S, of the inference rules whose conditions are satisfied by an ordering $\succ$, is weakly successful.

Theorem 1. *Let $\mathcal{R}$ be a TRS on a set $\mathcal{F}$ of symbols. If there exists an $\mathcal{F}$-stable ordering $\succ$ having the subterm property, such that for each defined symbol g, we have $SUCCESS(g, \succ)$, then every term of $\mathcal{T}(\mathcal{F})$ weakly innermost terminates.*

A formal description with a complete set of inference rules for describing the subtree cut process, and proofs of theorems are given in [11].

5 Finding a good derivation chain

As said previously, establishing weak termination of an undeterministic evaluation process warrants a result if a breadth-first strategy is adopted for this process. But such a strategy is in general very costly, and it is much better to have hints about the terminating derivations to compute them directly with a depth-first mechanism.

Our proof process, as it simulates the rewriting mechanism, gives complete information on a terminating rewriting branch. It allows extracting the exact application of rewrite rules that yields a normal form. To rewrite a term, it is enough to follow the rewriting scheme simulated by abstraction and narrowing in the proof trees.

We now formalize the use of the proof trees to compute a normal form for any term.

Definition 8. *Let $\mathcal{R}$ be a TRS proved weakly terminating with Theorem 1. The strategy tree ST_f associated to $f \in \mathcal{D}ef_{\mathcal{R}}$ is the proof tree obtained from the initial state $(\{f(x_1, \ldots, x_m)\}, \top, \top)$.*

Definition 9. *Let $\mathcal{R}$ be a TRS proved weakly terminating with Theorem 1. Let $ST = \{ST_f | f \in \mathcal{D}ef_{\mathcal{R}}\}$ be the set of strategy trees of $\mathcal{R}$ and $s = f(s_1, \ldots, s_m) \in \mathcal{T}(\mathcal{F})$. Normalizing s with respect to ST into $norm_{ST}(s)$ is defined in the following way:*

- *if $f \in \mathcal{C}ons_{\mathcal{R}}$, then $norm_{ST}(f(s_1, \ldots, s_n)) = f(norm_{ST}(s_1), \ldots, norm_{ST}(s_n))$,*
- *if $f \in \mathcal{D}ef_{\mathcal{R}}$, then normalizing s with respect to ST into $norm_{ST}(s)$ is performed by following the steps in the strategy tree ST_f of f, where $t = g(t_1, \ldots, t_n)$ is any term of the transformation chain of s with respect to ST and $u = g(u_1, \ldots, u_n)$ is the corresponding term in ST_f:*
  - *if the step is* **Abstract**, *and abstracts u at positions $i_1, \ldots, i_p$, then $t \mapsto t[t'_1]_{i_1} \ldots [t'_p]_{i_p}$,*
    where $t'_j = \begin{cases} t|_{i_j}\downarrow & \text{if } WT(u|_{i_j}) \\ norm_{ST}(t|_{i_j}) & \text{otherwise,} \end{cases}$
  - *if the step is* **Narrow** *with $g(u_1, \ldots, u_n) \leadsto^{Inn}_{p,l\to r,\sigma} u'$, then $g(t_1, \ldots, t_n) \mapsto t'$ where t' is defined by $g(t_1, \ldots, t_n) \to^{Inn}_{p,l\to r,\mu} t' = \mu u'$, with $\theta = \mu\sigma$ on $Var(g(u_1, \ldots, u_n))$ and $g(t_1, \ldots, t_n) = \theta g(u_1, \ldots, u_n)$ if μ exists,*
    $t' = g(t_1, \ldots, t_n)$ and the normalizing process stops, otherwise,
  - *if the step is* **Stop**, *then $g(t_1, \ldots, t_n) \mapsto t'$,*
    where $t' = \begin{cases} g(t_1, \ldots, t_n)\downarrow & \text{if } WT(g(u_1, \ldots, u_n)) \\ norm_{ST}(g(t_1, \ldots, t_n)) & \text{otherwise.} \end{cases}$

Given a TRS $\mathcal{R}$, the previous definition assumes that if the predicate WT has been used to prove termination of a particular term t during the termination proof of $\mathcal{R}$, one is able to find a normalizing strategy for t. A simple sufficient condition is that t is proved strongly terminating, which can be established in most cases, like for WT, with the usable rules. Under this assumption, the following theorem holds.

Theorem 2. *Let $\mathcal{R}$ be a TRS proved weakly terminating with Theorem 1 and ST its set of strategy trees. Then for any term $t \in \mathcal{T}(\mathcal{F})$, $norm_{ST}(t)$ is an innermost normal form of t for $\mathcal{R}$.*

Let us come back to the TRS $\mathcal{R}$ presented in the introduction, built on $\mathcal{F} = \{f : 2, h : 2, g : 1, s : 1, 0 : 0\}$. We first prove that every ground term t of $\mathcal{T}(\mathcal{F})$ can be innermost normalized with $\mathcal{R}$, and then infer from this proof a strategy allowing normalization of any ground term of $\mathcal{T}(\mathcal{F})$.

Since the defined symbols of $\mathcal{R}$ are f, g, and h, we have to apply the inference rules to $f(x_1, x_2)$, $g(x_1)$ and $h(x_1, x_2)$. The proof trees, given in Table 2, show how the inference rules are applied, and provide the information needed to infer a strategy for normalizing any ground term. When **Narrow** applies, we specify the narrowing substitution, when it is useful for normalization, and in parentheses, the rewrite rule number used to narrow.

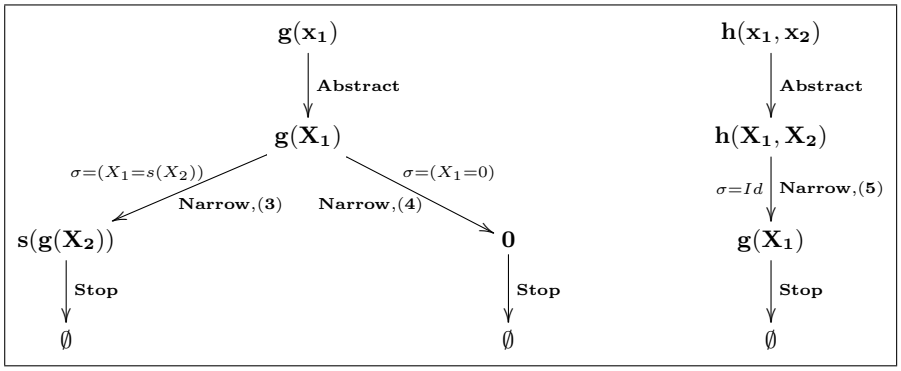
The subtree marked by $\odot$ in the proof tree of f is cut as soon as the subtree generated on the left from $\mathbf{f}(\mathbf{X_6}, \mathbf{s}(\mathbf{0}))$ with the same substitution (up to a renaming) $\sigma = (X_6 = g(X_7)) \wedge (X_7 \neq s(X_8) \wedge X_7 \neq 0)$ is successful.

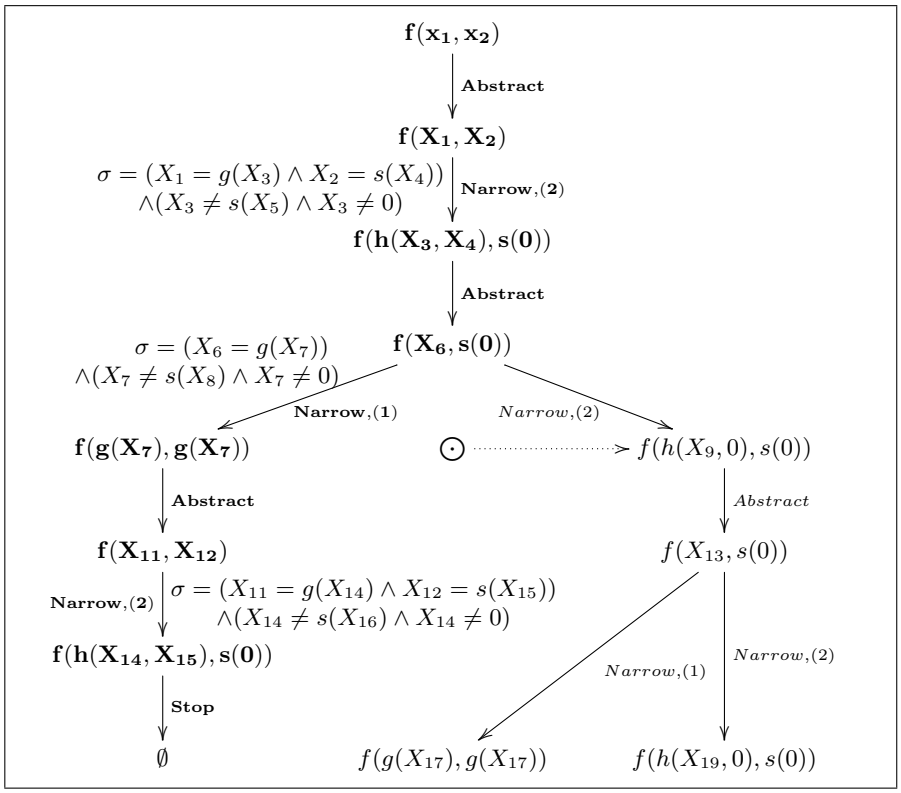
The final proof trees are **bold**. Since they are all successful, $\mathcal{R}$ is proved weakly innermost terminating on the ground term algebra. These proof trees are respectively the strategy trees $\mathbf{ST_g}, \mathbf{ST_h}$ and $\mathbf{ST_f}$, from which we can now infer a strategy normalizing any ground term t, according to Definition 9.

As an example, let us use the strategy to normalize the term $f(g(f(0,0)), s(0))$ following the steps of ST_f.

> **(Step 1 in $\mathbf{ST_f}$: Abstract)** The first step is **Abstract** at positions 1 and 2 by application of the induction hypothesis, and then we get $f(g(f(0,0)), s(0)) \mapsto f(norm_{ST}(g(f(0,\ 0))), norm_{ST}(s(0))$. Since s is a constructor, we have $norm_{ST}\ (s(0)) = s(norm_{ST}(0))$. Since 0 is a constructor constant, we have $norm_{ST}(0) = 0$, and finally $norm_{ST}(s(0)) = s(0)$. We now have to compute $norm_{ST}(g(f(0,\ 0)))$, by following the steps of ST_g.
>
> > **(Step 1 in $\mathbf{ST_g}$: Abstract)** The first step is **Abstract** at position 1 by application of the induction hypothesis, and then we get $g(f(0,0)) \mapsto g(norm_{ST}(f(0,0)))$. To compute $norm_{ST}(f(0,0))$, we have to follow the steps of ST_f.
> >
> > > **(Step 1 in $\mathbf{ST_f}$: Abstract)** The first step is **Abstract** at positions 1 and 2 by application of the induction hypothesis, and then we get $f(0,0) \mapsto f(norm_{ST}(0), norm_{ST}(0))$. Since 0 is a constant constructor, we have $norm_{ST}(0) = 0$, and then $f(0,0) \mapsto f(0,0)$.
> > >
> > > **(Step 2 in $\mathbf{ST_f}$: Narrow)** The second step is **Narrow** at the top position, with rule (2). The narrowing substitution σ is such that our current term $f(0,0)$ is not a ground instance of $\sigma f(X_1, X_2)$. Therefore $f(0,0) \mapsto f(0,0)$, and finally $norm_{ST}(f(0,0)) = f(0,0)$. We then come back to normalization of $g(f(0,0))$.
> >
> > **(Step 2 in $\mathbf{ST_g}$: Narrow)** Our current term is $g(f(0,0))$, and the second step of ST_g is **Narrow** at the top position, with rules (3) and (4). None of the narrowing substitutions σ is such that our current

Table 2. Proof trees for symbols g, h and f





term $g(f(0,0))$ is a ground instance of $\sigma g(X_1)$. Therefore $g(f(0,0)) \mapsto g(f(0,0))$, and finally $norm_{ST}(g(f(0,0))) = g(f(0,0))$. We then come back to normalization of our main term.

(Step 2 in $\mathbf{ST_f}$: Narrow) Our current term is $f(g(f(0,0)), s(0))$, and the current step in $\mathbf{ST_f}$ is **Narrow** at the top position with rule (2). The narrowing substitution σ is such that our current term is a ground instance of $\sigma f(X_1, X_2)$. So $f(g(f(0,0)), s(0)) \rightarrow^{\epsilon,(2)} f(h(f(0,0),0), s(0))$.

(Step 3 in $\mathbf{ST_f}$: Abstract) The current step in the proof tree is **Abstract** at position 1 thanks to the WT predicate, and more precisely thanks to the usable rules which give a strong terminating system. Then we have $h(f(0,0),0) \mapsto h(f(0,0),0)\downarrow$, and it suffices to rewrite $h(f(0,0),0)$ as long as a normal form is reached, which is guaranteed by the termination of the usable rules. Here we have $h(f(0,0),0) \rightarrow^{\epsilon,(5)} g(f(0,0))$. Finally we get $f(h(f(0,0),0), s(0)) \mapsto f(g(f(0,0)), s(0))$.

(Step 4 in $\mathbf{ST_f}$: Narrow) The current step in the tree is **Narrow** at the top position with rule (1). The narrowing substitution σ is such that our current term is a ground instance of $\sigma f(X_6, s(0))$. So $f(g(f(0,0)), s(0)) \rightarrow^{\epsilon,(1)} f(g(f(0,0)), g(f(0,0)))$.

(Step 5 in $\mathbf{ST_f}$: Abstract) The current step in the tree is **Abstract** at positions 1 and 2 thanks to the WT predicate, and then $f(g(f(0,0)), g(f(0,0))) \mapsto f(g(f(0,0))\downarrow, g(f(0,0))\downarrow)$. Since $g(f(0,0))$ is in normal form, we get $f(g(f(0,0)), g(f(0,0))) \mapsto f(g(f(0,0)), g(f(0,0)))$.

(Step 6 in $\mathbf{ST_f}$: Narrow) The current step of ST_f is **Narrow** at the top position, with rule (2). The narrowing substitution σ is such that our current term is a not a ground instance of $\sigma f(X_{11}, X_{12})$. Therefore the normalizing process stops on $f(g(f(0,0)), g(f(0,0)))$, which hence is a normal form of $f(g(f(0,0)), s(0))$.

For a more detailed development of this example, as well as for other examples, see [11].

6 Conclusion and perspectives

In this paper, we have proposed a method to prove weak innermost termination of term rewriting systems by explicit induction on the termination property. To simulate the innermost rewriting derivations of any ground term, we generate proof trees issued from patterns $g(x_1, \ldots, x_m)$ where g is a defined function symbol, in using two mechanisms: abstraction, introducing variables that represent ground normal forms, and narrowing, schematizing rewriting on ground terms.

When all proof trees have a successful branch for all ground instances of the patterns, the weak innermost termination property of the rewrite system is proved. Then from these successful branches, a normalizing strategy can be inferred for any ground term. We show how to extract the relevant information from the proof trees to guide the innermost normalization process.

Proving weak termination of a program and deducing a normalizing strategy can be achieved at *compile-time*. Then, to evaluate a data at *run-time* with no

risk of non-termination, it suffices to follow the strategy described in Section 5, that states which rule to apply and at which position in the term, at each step of the normalization process. Henceforth, evaluation at run-time is made very efficient, since it always leads to a result, i.e. an irreducible term.

Up to our knowledge, this is the first method proposed to ensure weak termination of rewriting systems, allowing to find a finite evaluation for every term.

The important point to automate our proof principle is the satisfaction of the constraints at each step of the proof. On many examples, this is immediate: as the ordering constraints only express the subterm property, they are trivially satisfied by any simplification ordering. Otherwise, we can use automatic ordering constraint solvers. As for abstraction constraints, they can be managed with an unsatisfiability test, for which simple sufficient conditions exist, that are automated. Thus, in general, weak termination proof can be completely automatic.

As in our approach, the rewriting strategy is explicitly handled in the proof principle, the method should be easily applicable to other strategies, especially to the outermost strategy, and to local strategies on operators. This potentially leads to a new functionality for CARIBOO, a toolbox for proving termination under strategies [9].

References

1. T. Arts and J. Giesl. Proving innermost normalization automatically. Technical Report 96/39, Technische Hochschule Darmstadt, Germany, 1996.
2. T. Arts and J. Giesl. Proving innermost normalisation automatically. In *Proceedings 8th Conference on Rewriting Techniques and Applications, Sitges (Spain)*, volume 1232 of *Lecture Notes in Computer Science*, pages 157–171. Springer-Verlag, 1997.
3. Peter Borovanský, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Christophe Ringeissen. An overview of ELAN. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the second International Workshop on Rewriting Logic and Applications*, volume 15, `http://www.elsevier.nl/locate/entcs/volume15.html`, Pont-à-Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science. Report LORIA 98-R-316.
4. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187–243, 2002.
5. H. Comon. Disunification: a survey. In Jean-Louis Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 9, pages 322–359. The MIT press, Cambridge (MA, USA), 1991.
6. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
7. Nachum Dershowitz and Jean-Pierre Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990. Also as: Research report 478, LRI.

8. O. Fissore, I. Gnaedig, and H. Kirchner. Termination of rewriting with local strategies. In M. P. Bonacina and B. Gramlich, editors, *Selected papers of the 4th International Workshop on Strategies in Automated Deduction*, volume 58 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers B. V. (North-Holland), 2001.
9. O. Fissore, I. Gnaedig, and H. Kirchner. CARIBOO : An induction based proof tool for termination with strategies. In *Proceedings of the Fourth International Conference on Principles and Practice of Declarative Programming*, pages 62–73, Pittsburgh (USA), October 2002. ACM Press.
10. O. Fissore, I. Gnaedig, and H. Kirchner. Outermost ground termination. In *Proceedings of the Fourth International Workshop on Rewriting Logic and Its Applications*, volume 71 of *Electronic Notes in Theoretical Computer Science*, Pisa, Italy, September 2002. Elsevier Science Publishers B. V. (North-Holland).
11. O. Fissore, I. Gnaedig, and H Kirchner. Proving weak termination also provides the right way to terminate - Extended version. Technical report, LORIA, Nancy (France), March 2004. Available at http://www.loria.fr/∼gnaedig/PAPERS/REPORTS/wt-extended-2004.ps.
12. K. Futatsugi and A. Nakagawa. An overview of CAFE specification environment – an algebraic approach for creating, verifying, and maintaining formal specifications over networks. In *Proceedings of the 1st IEEE Int. Conference on Formal Engineering Methods*, 1997.
13. I. Gnaedig, H. Kirchner, and O. Fissore. Induction for innermost and outermost ground termination. Technical Report A01-R-178, LORIA, Nancy (France), September 2001.
14. Goubault-Larreck. Well-founded recursive relations. In *Proc. 15th Int. Workshop Computer Science Logic (CSL'2001)*, volume 2142 of *Lecture Notes in Computer Science*, Paris, 2001. Springer-Verlag.
15. J. Goubault-Larrecq. A proof of weak termination of typed lambda-sigma-calculi. In *Proceedings of the TYPES'96 Workshop*, volume 1512 of *Lecture Notes in Computer Science*, Aussois (France), 1998. Springer-Verlag.
16. Bernhard Gramlich. Relating innermost, weak, uniform and modular termination of term rewriting systems. In Andrei Voronkov, editor, *Proceedings of the 3rd International Conference on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 285–296, St. Petersburg, Russia, July 1992. Springer-Verlag.
17. Bernhard Gramlich. On termination and confluence properties of disjoint and constructor-sharing conditional rewrite systems. *Theoretical Computer Science*, 165(1):97–131, September 1996.
18. G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, I. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic*, chapter 11, pages 395–414. The MIT press, 1991.
19. P. Klint. A meta-environment for generating programming environments. *ACM Transactions on Software Engineering and Methodology*, 2:176–201, 1993.
20. S. Lucas. Termination of rewriting with strategy annotations. In A. Voronkov and R. Nieuwenhuis, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 669–684, La Habana, Cuba, December 2001. Springer-Verlag, Berlin.