A. Bouhoula and F. Jacquemard

Automatic Verification of Sufficient Completeness for Specifications of Complex Data Structures

Research Report LSV-05-17
August 2005

Laboratoire Spécification et Vérification

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Automatic Verification of Sufficient Completeness for Specifications of Complex Data Structures*

Adel Bouhoula[1]**. and Florent Jacquemard[2]

[1] École supérieure des communications de Tunis, Tunisia. bouhoula@planet.tn
[2] INRIA Futurs & LSV UMR CNRS ENS Cachan, France. florent.jacquemard@inria.fr

Abstract. We present a new procedure for testing sufficient completeness for conditional and constrained term rewriting systems in presence of constrained axioms for constructors. Such axioms allow to specify complex data structures like e.g. sets or sorted lists. Our approach is based on tree grammars with constraints, a formalism which permits an exact representation of languages of ground constructor terms in normal form. The procedure is sound and complete and has been successfully used for checking the sufficient completeness of several specifications where related former techniques fail.

Keywords: Sufficient Completeness, Algebraic Specifications, Inductive Theorem Proving, Conditional and Constrained Term Rewriting.

1 Introduction

Sufficient completeness [Gut75] is a fundamental property of algebraic specifications. It expresses that, given a term rewriting system (TRS) $\mathcal{R}$ and a set $\mathcal{C}$ of distinguished operators called *constructors*, every ground term can be rewritten to a constructor term, *i.e.* a term built only from constructors of $\mathcal{C}$. Intuitively, constructor terms represent values and the others symbols (called *defined functions*) represent functions defined over these values by $\mathcal{R}$. The sufficient completeness of $\mathcal{R}$ means that these functions are completely defined. This property is strongly related to inductive theorem proving, and in particular to *ground reducibility*, the property that all ground instances (instances without variables) of a given term are reducible by a given TRS [KNZ87,KNRZ91].

Sufficient Completeness is undecidable in general [Gut78]. However, decidability results have been obtained for restricted cases [GH78,HH82,Der83,Kou85,Com86,LLT90,KNRZ91]. Tree automata with constraints have appeared to be a well suited framework in this context of decision of sufficient completeness and related properties, for a survey, see [CGJ+02].

The above results are concerned with unconditional rewriting only. In the context of conditional specifications, the art is less developed, probably because the problem is much harder. A procedure has proposed in [BR90] for checking completeness for a very small class of conditional rewrite systems. [Bou96] addresses the same problem for parametrized conditional specifications. However, the completeness of this procedure assume that the axioms for defined functions are left-linear and that there are no axioms for constructors. In the opposite case, the efficiency of the procedure is very limited (see the examples of Sections 5 and 7). In [BJ01], tree automata techniques are used to check sufficient completeness of specifications with axioms between constructors. This technique has been generalized to membership equational logic [BJM00] and another extension has been proposed recently [HCM05]. The approaches of [BJ01,BJM00] admit rewrite rules for constructors They work by transforming the initial specification in order to get rid of these axioms. However, they are limited to axioms for constructors which are non-constrained and *left-linear* (see the example of Section 7).

* This work has been partially supported by INRIA Lorraine and by a grant SSHN of the French institute for cooperation in the French embassy in Tunisia.

** This works has been started while the first author was staying at Laboratoire Spécification and Vérification, UMR CNRS ENS Cachan.

In this paper, we present an effective method for testing sufficient completeness of conditional and constrained rewrite systems with axioms for constructors which can be constrained and non left linear. Such rules permit the axiomatization of complex data structures like *e.g.* sorted lists, as illustrated in Section 7. To our knowledge, no previous work was able to check the sufficient completeness of this kind of specifications.

Our method is based on the incremental construction of a *pattern tree*. The idea is roughly that sufficient completeness of a TRS $\mathcal{R}$ is ensured as long as every term which contain only one defined function symbol, at the top position, can be rewritten (at the top), under the assumption that $\mathcal{R}$ is terminating. The pattern tree constructed is hence labelled by constrained terms of the form $f(t_1, \ldots, t_n) \, [\![c]\!]$ where f is a defined symbol and $t_1, \ldots, t_n$ are constructor terms (c is a constraint), and every node in the tree is obtained from its parent by instantiation of a variable with a constructor term. Since it is sufficient to restrict the rewriting to the top position, the positions of variables to be instantiated can be limited to positions of left hand sides of rules of $\mathcal{R}$. This ensures the termination of the procedure. We call such positions *induction positions*. Moreover, still under the hypothesis that the rewrite system is terminating, the direct (constructor) subterms $t_1, \ldots, t_n$ of the term to be tested can be assumed irreducible ($\mathcal{R}$-normal forms). This suggests the use of constrained tree grammars, which are well suited for the finite representation of sets of normal forms [CGJ+02], for the above incremental instantiation.

The criteria for the verification of sufficient completeness during the construction of the pattern tree is that all the leaves are *strongly ground reducible* by $\mathcal{R}$. This latter sufficient condition for the ground reducibility by $\mathcal{R}$ requires in particular that the conditions of candidate rules of $\mathcal{R}$ (for reducibility of ground instances) are inductive consequences of $\mathcal{R}$. Therefore, our procedure rely on a system for inductive theorem proving, and it has been designed to fit with a general framework defined recently in [BJ05] for that purpose, which is also based on normal form constrained tree grammars.

Let us summarize below a few arguments in favor of our approach. 1) It handles axioms for constructors even with constraints. 2) It is sound for ground confluent and terminating specifications and it is also complete, without restriction. Moreover, it works directly on the given specification, at the opposite of *e.g.* [BJ01] – see remark above. 3) The choice of constrained tree grammar is crucial for the proof of completeness, and moreover eased considerably the formal proofs of our procedure (both for correctness and completeness). A reason why this formalism seems particular relevant both for sufficient completeness checking and inductive theorem proving (in the lines of [BJ05]) is that it provides a finite representation of ground constructor terms in normal form which is *exact*. Many other approaches rely on representations like *cover sets* [Kap94,Bou96]) which may be over approximating (*i.e.* they may represent also some reducible terms) in presence of axioms for constructors (for an illustration, see the examples in Sections 5 and 7). 4) If the specification is not sufficiently complete, the procedure stops (finding at least one leave which is not strongly ground reducible) and returns the set of patterns alongs with cnstraints on which a function is not defined, as an hint for the rewrite rules which must be added to the system in order to make it sufficiently complete. It is also possible to learn the conditions of such a rule from the failure of the strongly ground reducibility test. 5) The method has been successfully used for checking sufficient completeness for several specifications where related techniques fail.

The paper is organized as follows. In Section 2, we briefly introduce the basic concepts about constrained and conditional term rewriting. Constrained tree grammars are presented in Section 3, and Section 4 introduces sufficient completeness and the method of [BJ05] for inductive theorem proving. The procedure for checking sufficient completeness is defined in Section 6, where its correctness and completeness are proved. Finally, Sections 5 and 7 present examples of applications of this procedure to specifications of respectively integers and sorted lists.

2 Preliminaries

The reader is assumed familiar with the basic notions of term rewriting [DJ90] and mathematical logic. Notions and notations not defined here are standard.

Terms and substitutions. We assume given a many sorted signature $(\mathcal{S}, \mathcal{F})$ (or simply $\mathcal{F}$, for short) where $\mathcal{S}$ is a set of *sorts* and $\mathcal{F}$ is a finite set of function symbols, each symbol f is given with a profile $f : S_1 \times \ldots \times S_n \to S$ where $S_1, \ldots, S_n, S \in \mathcal{S}$ and n is the *arity* of f. We assume moreover that the signature $\mathcal{F}$ comes in two parts, $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ where $\mathcal{C}$ a set of *constructor symbols*, and $\mathcal{D}$ is a set of *defined symbols*. Let $\mathcal{X}$ be a family of sorted variables. We sometimes denote variables with sort exponent like x^S in order to indicate that x has sort $S \in \mathcal{S}$.

The set of well-sorted terms over $\mathcal{F}$ (resp. constructor well-sorted terms) with variables in $\mathcal{X}$ will be denoted by $\mathcal{T}(\mathcal{F}, \mathcal{X})$ (resp. $\mathcal{T}(\mathcal{C}, \mathcal{X})$). The subset of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ (resp. $\mathcal{T}(\mathcal{C}, \mathcal{X})$) of variable-free terms, or *ground* terms, is denoted $\mathcal{T}(\mathcal{F})$ (resp. $\mathcal{T}(\mathcal{C})$). We assume that each sort contains at least one ground term. The sort of a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is denoted $sort(t)$.

A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is identified as usual to a function from its set of *positions* (strings of positive integers) $\mathcal{P}os(t)$ to symbols of $\mathcal{F}$ and $\mathcal{X}$. The subset of non-variable positions $p \in \mathcal{P}os(t)$ such that $t(p) \in \mathcal{F}$ is denoted $\mathcal{P}os_{\mathcal{F}}(t)$. We note Λ the empty string (root position). The *subterm* of t at position p is denoted by $t|_p$. The result of replacing $t|_p$ with s at position p in t is denoted by $t[s]_p$. This notation is also used to indicate that s is a subterm of t and in this case, the position p may be omitted. We note $var(t)$ the set of variables occurring in t. A term t is *linear* if every variable of $var(t)$ occurs exactly once in t.

A substitution is a finite mapping $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where $x_1, \ldots, x_n \in \mathcal{X}$ and $t_1, \ldots t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. As usual, we identify substitutions with their morphism extension to terms. A variable renaming is a (well) sorted bijective substitution which maps variables to variables. A substitution σ is *grounding* for a term t if the domain of σ contains all the variables of t and the codomain of σ contains only ground terms. We use postfix notation for substitutions application and composition. We write $\mathrm{mgu}(s, t)$ to denote the most general unifier of terms s and t.

Constraints for terms and clauses. We assume given a constraint language $\mathcal{L}$, which is a finite set of predicate symbols with a recursive Boolean interpretation in the domain of ground constructor terms of $\mathcal{T}(\mathcal{C})$. Typically, $\mathcal{L}$ contains the syntactic equality $. \approx .$ (syntactic disequality $. \not\approx .$) and some (recursive) simplification ordering $. \prec .$ on ground constructor terms. *Constraints* on the language $\mathcal{L}$ are Boolean combinations of atoms of the form $P(t_1, \ldots, t_n)$ where $P \in \mathcal{L}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{X})$. By convention, an empty combination is interpreted to true.

We may extend the application of substitutions from terms to constraints in a straightforward way, and therefore define a solution for a constraint c as a (constructor) substitution σ grounding for all terms in c and such that $c\sigma$ is interpreted to true. The set of solutions of the constraint c is denoted $sol(c)$. A constraint c is *satisfiable* if $sol(c) \neq \emptyset$ (and *unsatisfiable* otherwise).

A *constrained term* $t \, [\![c]\!]$ is a linear term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ together with a constraint c, which may share some variables with t. Note that the assumption that t is linear is not restrictive, since any non linearity may be expressed in the constraint, for instance $f(x, x) \, [\![c]\!]$ is semantically equivalent to $f(x_1, x_2) \, [\![c \wedge x_1 \approx x_2]\!]$. A *literal* is an equation $s = t$ or an oriented equation $s \to t$ between two terms. We consider *clauses* of the form $\Gamma \Rightarrow L$ where Γ is a conjunction of literals and L is a literal. We find convenient to see clauses themselves as terms on a signature extended by the predicate symbols $=$, and $\to$ and the connective $\wedge$ and $\Rightarrow$. This way, we can define a *constrained clause* as a constrained term.

Conditional constrained rewriting. A *conditional constrained rewrite rule* is a constrained clause of the form $\Gamma \Rightarrow l \to r \, [\![c]\!]$ such that Γ is a conjunction of equations $u = v$, called the *condition* of the rule, the terms l and r (called resp. left- and right-hand side) are linear and have the same sort, and c is a constraint. When the condition Γ is empty, it is called a *constrained rewrite rule*. A set of conditional constrained, resp. constrained, rules is called a *conditional constrained* (resp. *constrained*) *rewrite system*.

Let $\mathcal{R}$ be a conditional constrained rewrite system. A term $t[l\sigma] \, [\![d]\!]$ rewrites to $t[r\sigma] \, [\![d]\!]$ by a rule $\rho \equiv \Gamma \Rightarrow l \to r \, [\![c]\!] \in \mathcal{R}$, written $t[l\sigma] \, [\![d]\!] \to t[r\sigma] \, [\![d]\!]$ if the substitution σ is such that $d \wedge \neg c\sigma$ is unsatisfiable and $u\sigma \downarrow_{\mathcal{R}} v\sigma$ for all $u = v \in \Gamma$. The rewrite relation induced by the rules of a (conditional) constrained system $\mathcal{R}$, and its transitive and the reflexive transitive closures, are respectively denoted $\xrightarrow[\mathcal{R}]{}$, $\xrightarrow[\mathcal{R}]{+}$ and $\xrightarrow[\mathcal{R}]{*}$, and $u \downarrow_{\mathcal{R}} v$ stands for $\exists w, \; u \xrightarrow[\mathcal{R}]{*} w \xleftarrow[\mathcal{R}]{*} v$.

Note the semantical difference between conditions and constraints in rewrite rules. The validity of the condition is defined wrt the system $\mathcal{R}$ whereas the interpretation of constraint is fixed and independent from $\mathcal{R}$.

A constrained term $s\,[\![c]\!]$ is *reducible* by $\mathcal{R}$ if there is some $t\,[\![c]\!]$ such that $s\,[\![c]\!] \xrightarrow[\mathcal{R}]{} t\,[\![c]\!]$. Otherwise $s\,[\![c]\!]$ is called $\mathcal{R}$-*irreducible*, or an $\mathcal{R}$-*normal form*. A constrained term $t\,[\![c]\!]$ is *weakly reducible* by $\mathcal{R}$ if it contains as a subterm an instance of a left hand side of a rule of $\mathcal{R}$. A constrained term $t\,[\![c]\!]$ is *ground reducible* (resp. *ground irreducible*) if $t\sigma$ is reducible (resp. irreducible) for every irreducible solution σ of c grounding for t.

The system $\mathcal{R}$ is terminating if there is no infinite sequence $t_1 \xrightarrow[\mathcal{R}]{} t_2 \xrightarrow[\mathcal{R}]{} \ldots$, $\mathcal{R}$ is *ground confluent* if for any ground terms $u, v, w \in \mathcal{T}(\mathcal{F})$, $v \xleftarrow[\mathcal{R}]{*} u \xrightarrow[\mathcal{R}]{*} w$, implies then $v \downarrow_{\mathcal{R}} w$, and $\mathcal{R}$ is *ground convergent* if it is both ground confluent and terminating.

Constructor specification. We assume from now on given a conditional constrained rewrite system $\mathcal{R}$ of the form $\mathcal{R}_{\mathcal{D}} \uplus \mathcal{R}_{\mathcal{C}}$ where: $\mathcal{R}_{\mathcal{D}}$ is contains conditional constrained rules of the form $\Gamma \Rightarrow f(\ell_1, \ldots, \ell_n) \to r\,[\![c]\!]$ such that $f \in \mathcal{D}$, $\ell_1, \ldots, \ell_n \in \mathcal{T}(\mathcal{C}, \mathcal{X})$ and $\mathcal{R}_{\mathcal{C}}$ contains constrained rewrite rules containing only constructor symbols from $\mathcal{C}$.

3 Constrained Tree Grammars

Constrained tree grammars permit an exact representation of the set of $\mathcal{R}$-irreducible ground terms for a given constrained rewrite system $\mathcal{R}$, *i.e.* the initial model of $\mathcal{R}$ for inductive theorem [see TATA, ch. 4]. In our approach, they are used in the procedure for checking sufficient completeness. to generate incrementally a relevant set of terms, by means of non terminal replacement following production rules,

Definition 1. *A* constrained tree grammar $\mathcal{G} = (Q, \Delta)$ *is given by a finite set Q of non-terminals of the form $\llcorner u \lrcorner$, where u is a linear term of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and a finite set Δ of production rules of the form $\llcorner t \lrcorner := f(\llcorner u_1 \lrcorner, \ldots, \llcorner u_n \lrcorner)\,[\![c]\!]$ where $f \in \mathcal{F}$, $\llcorner t \lrcorner, \llcorner u_1 \lrcorner, \ldots, \llcorner u_n \lrcorner \in Q$ and c is a constraint.*

The non-terminals are always considered modulo variable renaming. In particular, we assume that the term $f(u_1, \ldots, u_n)$ is linear.

Term generation, language. We associate to a given constrained tree grammar $\mathcal{G} = (Q, \Delta)$ a finite set of new unary constraint predicates of the form $.{:}\llcorner u \lrcorner$, where $\llcorner u \lrcorner \in Q$ (modulo variable renaming), called *membership constraints*. The production relation between constrained terms $\vdash_{\mathcal{G},x}$, or $\vdash_x$ or $\vdash$ for short when $\mathcal{G}$ is clear from context, is defined by:

$$t[x]\,[\![x{:}\llcorner u \lrcorner \wedge d]\!] \vdash_x t[f(y_1, \ldots, y_n)]\,[\![y_1{:}\llcorner u_1 \lrcorner \wedge \ldots \wedge x_n{:}\llcorner x_n \lrcorner \wedge c \wedge d\sigma]\!]$$

if there exists $\llcorner u \lrcorner := f(\llcorner u_1 \lrcorner, \ldots, \llcorner u_n \lrcorner)\,[\![c]\!] \in \Delta$ such that $f(u_1, \ldots, u_n) = u\sigma$ (we assume that the variables of $u_1, \ldots, u_n$ and c do not occur in the constrained term $t[x]\,[\![x{:}\llcorner u \lrcorner \wedge d]\!]$) and $x_1, \ldots, x_n$ are fresh variables, The reflexive transitive and transitive closures of the relation $\vdash$ are respectively denoted $\vdash^*$ and $\vdash^+$.

Definition 2. *The language $L(\mathcal{G}, \llcorner u \lrcorner)$ is the set of ground terms t* generated *by a constrained tree grammar $\mathcal{G}$ starting with the non-terminal $\llcorner u \lrcorner$, i.e. such that $x\,[\![x{:}\llcorner u \lrcorner]\!] \vdash^* t\,[\![c]\!]$ where c is satisfiable.*

Given $Q' \subseteq Q$, we note $L(\mathcal{G}, Q') = \bigcup_{\llcorner u \lrcorner \in Q'} L(\mathcal{G}, \llcorner u \lrcorner)$ and $L(\mathcal{G}) = L(\mathcal{G}, Q)$.
Given a constrained tree grammar $\mathcal{G} = (Q, \Delta)$, we can now define the interpretation of a membership constraint $t{:}\llcorner u \lrcorner$ where $\llcorner u \lrcorner \in Q$ by: $sol(t{:}\llcorner u \lrcorner) = \{\sigma \mid t\sigma \in L(\mathcal{G}, \llcorner u \lrcorner)\}$.

Normal form grammar. For every constrained rewrite system $\mathcal{R}_\mathcal{C}$, there exists a constrained tree grammar $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C}) = (Q_{\rm NF}(\mathcal{R}_\mathcal{C}), \Delta_{\rm NF}(\mathcal{R}_\mathcal{C}))$ which generates the language of ground $\mathcal{R}_\mathcal{C}$-normal forms. The construction of $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C})$ is a generalization of the one of [CJ03]. Intuitively, it corresponds to the complementation and completion of a tree grammar for $\mathcal{R}_\mathcal{C}$-reducible terms (such a grammar does essentially pattern matching of left hand side of rewrite rules), where every subset of non terminals (for the complementation) is represented by the most general unifiers of its elements.

Let $\mathcal{L}(\mathcal{R}_\mathcal{C})$ be the set containing the subterms of left hand side of constrained rules of $\mathcal{R}_\mathcal{C}$ and the strict subterms of left hand side of unconstrained rules of $\mathcal{R}_\mathcal{C}$ and one variable x^S for each sort $S \in \mathcal{S}$ and let $Q_{\rm NF}(\mathcal{R}_\mathcal{C})$ be the set of non terminals of the form $\llcorner \mathrm{mgu}(t_1, \ldots, t_n) \lrcorner$ where $\{\llcorner t_1 \lrcorner, \ldots, \llcorner t_n \lrcorner\}$ is a maximal subset of $\mathcal{L}(\mathcal{R}_\mathcal{C})$ such that $t_1, \ldots, t_n$ are unifiable. The set of transitions $\Delta_{\rm NF}(\mathcal{R}_\mathcal{C})$ contains every $\llcorner t \lrcorner := f(\llcorner u_1 \lrcorner, \ldots, \llcorner u_n \lrcorner) [\![\neg c]\!]$ such that $f \in \mathcal{F}$ with profile $S_1, \ldots, S_n \to S$ and $\llcorner u_1 \lrcorner, \ldots, \llcorner u_n \lrcorner \in Q_{\rm NF}(\mathcal{R}_\mathcal{C})$, $u_1, \ldots, u_n$ have respective sorts $S_1, \ldots, S_n$, t is the mgu of the set: $\{u \mid \llcorner u \lrcorner \in Q_{\rm NF}(\mathcal{R}_\mathcal{C}) \text{ and } u \text{ matches } f(u_1, \ldots, u_n)\}$, $c \equiv \bigvee_{l \to r [\![e]\!] \in \mathcal{R}_\mathcal{C},\ t = l\theta} e\theta$.

Example 1. Let *Int* be a sort for relative integers and assume a set $\mathcal{C}$ of constructor symbols containing $0 : Int$ and the unary predecessor and successor symbols $p, s : Int \to Int$. and let $\mathcal{R}_\mathcal{C} = \{s(p(x)) \to x,\ p(s(x)) \to x\}$.

The constrained tree grammar $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C})$ constructed as above has three non terminals $\llcorner s(x) \lrcorner$, $\llcorner p(x) \lrcorner$ and $\llcorner x^{Int} \lrcorner$, the latter is denoted $\llcorner 0 \lrcorner$ below, and the following production rules:

$$\llcorner 0 \lrcorner ::= 0 \qquad \llcorner s(x) \lrcorner ::= s(\llcorner 0 \lrcorner) \qquad \llcorner p(x) \lrcorner ::= p(\llcorner 0 \lrcorner)$$
$$\llcorner s(x) \lrcorner ::= s(\llcorner s(x) \lrcorner) \qquad \llcorner p(x) \lrcorner ::= p(\llcorner p(x) \lrcorner)$$

◇

Lemma 1. *The language $L(\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C}), Q_N F(\mathcal{R}_\mathcal{C}))$ is the set of $\mathcal{R}_\mathcal{C}$-irreducible terms of $\mathcal{T}(\mathcal{C})$.*

The proof of Lemma 1 can be found in [BJ05].

We shall consider below the normal form grammar $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C})$ associated to $\mathcal{R}_\mathcal{C}$ and we call a constrained term $t [\![c]\!]$ *decorated* if $c = x_1{:}\llcorner u_1 \lrcorner \wedge \ldots \wedge x_n{:}\llcorner u_n \lrcorner \wedge d$ where $\{x_1, \ldots, x_n\} = var(t)$, and $\llcorner u_1 \lrcorner, \ldots, \llcorner u_1 \lrcorner \in Q_{\rm NF}(\mathcal{R}_\mathcal{C})$ with $sort(u_i) = sort(x_i)$ for all $i \in [1..n]$, and d does not contain membership constraints.

4 Sufficient Completeness

We shall now define the problem we are interested in this paper and its relations with inductive theorem proving.

Sufficient completeness.

Definition 3. *The TRS $\mathcal{R}$ is sufficiently complete iff for all $t \in \mathcal{T}(\mathcal{F})$ there exists s in $\mathcal{T}(\mathcal{C})$ such that $t \xrightarrow[\mathcal{R}]{*} s$.*

The procedure we propose in Section 6 for checking sufficient completeness is based on an equivalent definition using the notion of sufficient completeness of the defined function symbols.

Definition 4. *A function symbol $f \in \mathcal{D}$ is* sufficiently complete *wrt $\mathcal{R}$ iff for all $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C})$, there exists $s \in \mathcal{T}(\mathcal{C})$ such that $f(t_1, \ldots, t_n) \xrightarrow[\mathcal{R}]{+} s$.*

Property 1. A TRS $\mathcal{R}$ is sufficiently complete iff every defined symbol $f \in \mathcal{D}$ is sufficiently complete wrt $\mathcal{R}$.

Proof. By induction on the number of occurrences of symbols of $\mathcal{D}$ in a given term $t \in \mathcal{T}(\mathcal{F})$. □

Inductive theorems. A clause C is a *deductive theorem* of $\mathcal{R}$ (denoted $\mathcal{R} \models C$) if it is valid in any model of $\mathcal{R}$, and it is an *inductive theorem* of $\mathcal{R}$ (denoted $\mathcal{R} \models_{\mathcal{I}nd} C$) iff for all substitution σ grounding for C, $\mathcal{R} \models C\sigma$. This definition is generalized to constrained clauses as follows.

Definition 5. *A constrained clause $C [\![c]\!]$ is an* inductive theorem *of $\mathcal{R}$ (denoted $\mathcal{R} \models_{\mathcal{I}nd} C [\![c]\!]$) if for all substitutions $\sigma \in sol(c)$ we have $\mathcal{R} \models C\sigma$.*

Inductive theorem proving with constrained tree grammars. In [BJ05] we develop a new approach for automated inductive theorem proving for specifications with constrained axioms for constructors and conditional and constrained axioms for defined functions. This procedure is sound and refutationally complete. The framework of [BJ05] is the same as in this paper, and is in particular based on a constrained tree grammar which describes exactly the set of ground constructor terms in normal form. In [BJ05], the grammar is used for the generation of subgoals during the proof by induction. In particular, the procedure of [BJ05] permits to prove conjectures with membership constraints, as defined in Section 3. Therefore, it can be called to discharge proofs obligations during the procedure for checking completeness defined in Section 6.

Example 2. Let us illustrate the principle of the procedure of [BJ05] on an example. We complete the specification of Example 1 with a sort *Bool* for Booleans, two constants of $\mathcal{C}$, $true, false : Bool$ and only one binary defined symbol $\leq$: $Int \times Int \to Bool$, in $\mathcal{D}$. Let $\mathcal{R}_\mathcal{D}$ be the following set of conditional rules:

$$\begin{array}{lll} 0 \leq 0 \to true, & s(x) \leq y \to x \leq p(y), & 0 \leq x = true \Rightarrow 0 \leq s(x) \to true, \\ 0 \leq p(0) \to false, & p(x) \leq y \to x \leq s(y), & 0 \leq x = false \Rightarrow 0 \leq p(x) \to false. \end{array}$$

We show that the following clause:

$$0 \leq x_1 = true \, [\![x_1 : \llcorner s(x) \lrcorner]\!] \tag{1}$$

is an inductive theorem of $\mathcal{R}$, following roughly the method of [BJ05] which is also based on the constrained tree grammar $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C})$. Applying the production rules of $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C})$ to (1), we obtain only two subgoals:

$$0 \leq s(x_1) = true \, [\![x_1 : \llcorner 0 \lrcorner]\!] \tag{2}$$

and

$$0 \leq s(x_1) = true \, [\![x_1 : \llcorner s(x) \lrcorner]\!] \tag{3}$$

The first subgoal (2) can further instantiated by $\mathcal{G}_{\rm NF}(\mathcal{R}_\mathcal{C})$ into $0 \leq s(0) = true$, and this equation rewrites by $\mathcal{R}_\mathcal{D}$ into the tautology $true = true$.

The second subgoal (3) can be simplified into the tautology $true = true$ using the clause (1), used as an induction hypothesis. The use of (1) as an induction hypothesis is possible because (1) is strictly smaller than (3). ◇

Strong ground irreducibility. An important issue in the incremental procedure of Section 6 is to detect whether all the ground instances of a term in construction are $\mathcal{R}_\mathcal{D}$-reducible. For this purpose, we use the following sufficient condition for ground reducibility, based on the notion of inductive theorem.

Definition 6. *A constrained term $t \, [\![c]\!]$ is* strongly ground reducible *wrt $\mathcal{R}$ if there are n rules $\Gamma_i \Rightarrow l_i \to r_i \, [\![c_i]\!]$ of $\mathcal{R}_\mathcal{D}$, with $i \in [1..n]$ and $n > 0$, such that $t = l_i\sigma_i$ and $c \wedge \neg c_i\sigma_i$ is unsatisfiable for each $i \in [1..n]$ and $\mathcal{R} \models_{\mathcal{I}nd} \Gamma_1\sigma_1 \, [\![c \wedge c_1\sigma_1]\!] \vee \ldots \vee \Gamma_n\sigma_n \, [\![c \wedge c_n\sigma_n]\!]$.*

Note that by definition, every strongly ground reducible constrained term is weakly reducible. Moreover, when $\mathcal{R}$ is ground convergent, every strongly ground reducible constrained term is ground reducible. This is proved in the proof of Theorem 2. The converse is not true, as shown by the following example.

Example 3. Let $\mathcal{R}_\mathcal{D} = \{ g(x, y) \to false \, [\![x \leq y]\!], \; g(s(x), y) \to false \, [\![s(x) > y]\!] \}$. Note that $\mathcal{R}$ is sufficiently complete. The term $t = g(x, y)$, where x and y have sort *Int* is ground reducible but not strongly ground reducible. ◇

5 Example: Integers

We continue with Examples 1 and 2. Since $\mathcal{R}_\mathcal{C}$ is terminating, in order to check the sufficient completeness of the symbol $\leq$ wrt $\mathcal{R}$, it is sufficient to consider the reductions (under $\mathcal{R} = \mathcal{R}_\mathcal{C} \uplus \mathcal{R}_\mathcal{D}$) of the terms of the form $t_1 \leq t_2$ where t_1 and t_2 are $\mathcal{R}_\mathcal{C}$-irreducible terms of $\mathcal{T}(\mathcal{C})$. By Lemma 1, it is equivalent to study the terms produced by $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_\mathcal{C})$ starting from terms of the form $x_1 \leq x_2 \, [\![x_1{:}n_1 \wedge x_2{:}n_2]\!]$ where n_1 and n_2 are non-terminal symbols of $Q_{\mathrm{NF}}(\mathcal{R}_\mathcal{C})$. For sake of simplicity, we shall denote such a term $n_1 \leq n_2$ below. There are nine such terms, and the derivations are described in Figure 1 and detailed as follows:

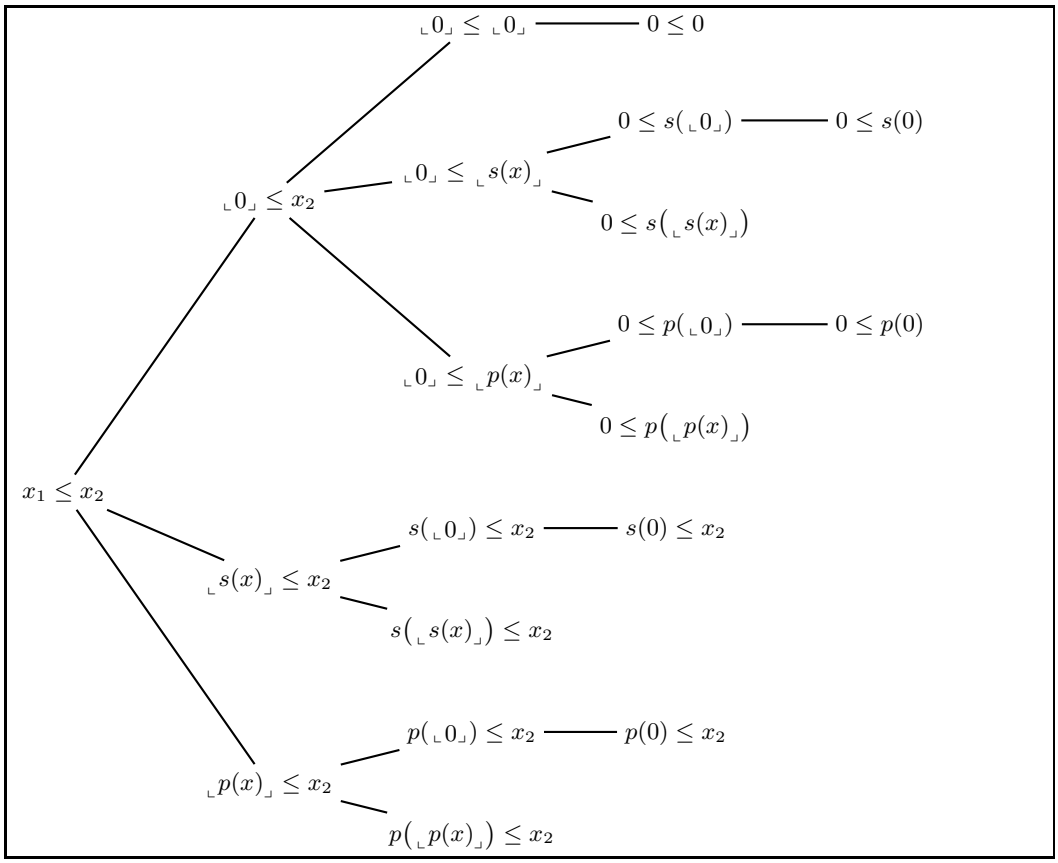

Fig. 1. Sufficient completeness of $\leq$.

- $\llcorner 0 \lrcorner \leq \llcorner 0 \lrcorner$ is instantiated, using $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_\mathcal{C})$, into $0 \leq 0$ which is reduced into *true* by $\mathcal{R}_\mathcal{C}$.
- $\llcorner 0 \lrcorner \leq \llcorner s(x) \lrcorner$ is instantiated into $\llcorner 0 \lrcorner \leq s(\llcorner 0 \lrcorner)$ and $\llcorner 0 \lrcorner \leq s\big(\llcorner s(x) \lrcorner\big)$. The first term is further instantiated into $0 \leq s(0)$, which is reduced into *true* by $\mathcal{R}_\mathcal{D}$. The second term is instantiated into $0 \leq s\big(\llcorner s(x) \lrcorner\big)$, which is an abbreviation for $0 \leq s(x_2) \, [\![x_2{:}\llcorner s(x) \lrcorner]\!]$. It is reducible to *true*, since, as we have seen in Example 2, $0 \leq x_2 = \mathit{true} \, [\![x_2{:}\llcorner s(x) \lrcorner]\!]$ is an inductive theorem of $\mathcal{R}$.
- $\llcorner 0 \lrcorner \leq \llcorner p(x) \lrcorner$ is instantiated into $\llcorner 0 \lrcorner \leq p(\llcorner 0 \lrcorner)$ and $\llcorner 0 \lrcorner \leq p\big(\llcorner p(x) \lrcorner\big)$. The first term is further instantiated into $0 \leq p(0)$, which is reduced into *false* by $\mathcal{R}_\mathcal{D}$. The second term is instantiated into $0 \leq p\big(\llcorner p(x) \lrcorner\big)$, which is an abbreviation for $0 \leq p(x_2) \, [\![x_2{:}\llcorner p(x) \lrcorner]\!]$. It is reducible to *false*, since, similarly as in Example 2, we can show that $0 \leq x_2 = \mathit{false} \, [\![x_2{:}\llcorner p(x) \lrcorner]\!]$ is an inductive theorem of $\mathcal{R}$.
- $\llcorner s(x) \lrcorner \leq n_2$ (whatever n_2) is instantiated into $s(\llcorner 0 \lrcorner) \leq n_2$ and $s\big(\llcorner s(x) \lrcorner\big) \leq n_2$. Both terms are instances of the left hand side of the rule $s(x) \leq y \to x \leq p(y)$ of $\mathcal{R}_\mathcal{D}$.

– $\llcorner p(x) \lrcorner \leq n_2$ (whatever n_2) is instantiated into $p(\llcorner 0 \lrcorner) \leq n_2$ and $p(\llcorner p(x) \lrcorner) \leq n_2$. Both terms are instances of the left hand side of the rule $p(x) \leq y \rightarrow x \leq s(y)$ of $\mathcal{R}_{\mathcal{D}}$.

The proof of the completeness of $\leq$ fails with the methods of [BR90,Bou96]. Indeed, the following cover set for the sort *Int*: $\{0, s(x), p(x)\}$ is not relevant because, obviously, it does not describe exactly the set of $\mathcal{R}$-irreducible ground constructor terms. For instance $p(s(0))$ is an instance of $p(x)$ but is not irreducible. Note that the completeness of $\leq$ can be proved by [BJ01,BJM00] since the axioms for constructors are non-constrained and *left-linear*. However, we recall that these procedures do not work directly on the given specification but transform it in order to get rid of the axioms between constructors.

6 Verification of Sufficient Completeness

Sufficient completeness is undecidable for conditional rewrite systems. We propose in this section a complete procedure for testing this property, based on an oracle for verifying strongly ground reducibility, *i.e.* for inductive theorem proving. A procedure to solve such an oracle can be found in the framework presented in Section 4 [BJ05], which is also based on the constrained tree grammar $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}})$.

Pattern trees. The procedure checks the sufficient completeness of each defined symbol of $f \in \mathcal{D}$ by the incremental construction of a multi-rooted pattern tree called *pattern tree* of f and denoted $dtree(f)$. The nodes of $dtree(f)$ are labelled by decorated constrained terms of the form $f(t_1, \ldots, t_n)\,[\![c]\!]$ such that $t_i \in \mathcal{T}(\mathcal{C}, \mathcal{X})$ for every $i \in [1..n]$. Each root of $dtree(f)$ is labelled by a decorated term $f(x_1, \ldots, x_n)\,[\![x_1{:}\llcorner u_1 \lrcorner \wedge \ldots \wedge x_n{:}\llcorner u_n \lrcorner]\!]$ where $x_1, \ldots, x_n$ are distinct variables and $\llcorner u_1 \lrcorner, \ldots, \llcorner u_1 \lrcorner \in Q_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}})$. The successors of any internal node $dtree(f)$ are determined by the inference rules described in Figure 2.

Inference rules for sufficient completeness. The algorithm presented in Figure 2 for the construction of $dtree(f)$ operates incrementally, by mean of non terminal replacement following the production rules of $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}})$ applied to a decorated term labelling a leaf of the tree constructed so far, until the term obtained becomes strongly ground reducible. In order to ensure the termination of the algorithm, the replacement are limited to *induction variables*.

Definition 7. *The set $\mathcal{I}nd\mathcal{P}os(f, \mathcal{R})$ of* induction positions *of $f \in \mathcal{D}$ is the set of non-variable positions $p \neq \Lambda$ of left-hand sides of rules of $\mathcal{R}_{\mathcal{D}}$. Let t be a term of the form $f(t_1, \ldots, t_n)$ where $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{X})$. The set of* induction variables *of a term t, denoted $\mathcal{I}nd\mathcal{V}ar(t)$, is the subset of variables of $\mathcal{V}ar(t)$ which occur in t at an induction position of f.*

Intuitively, it is sufficient to consider induction variables only for the application of the production rules of $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}})$, because any ground instance of the term considered (the term labelling a leaf of $dtree(f)$) may be reduced by $\mathcal{R}$ at the root position only.

Let us now describe the inference rules of Figure 2 in a little more details.

Instantiation applies the production rules of the normal-form grammar $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}})$ to the decorated term $t\,[\![c]\!]$.

Strongly Ground Reducible Leaf checks a sufficient condition for the ground reducibility of $t\,[\![c]\!]$. In order to check that a conjunction of constraints is unsatisfiable, we use our procedure of [BJ05] for inductive theorem proving.

Irreducible Leaf produce a failure when none of the two above inferences applies. to a a leaf $t\,[\![c]\!]$. It means in this case that the symbol f is not sufficiently complete wrt $\mathcal{R}$. The term $t\,[\![c]\!]$ provides an hint on rule (exactly the left hand side and the constraint of this rule) which must be added to $\mathcal{R}$ in order to complete the specification of f. It is also possible to learn the conditions of such a rule from the failure of the strongly ground reducibility test.

Instantiation:	$\dfrac{t\,[\![c]\!]}{t'\,[\![c']\!]}$ `if` $t\,[\![c]\!]$ is not strongly ground reducible `where` $\begin{cases} x \in \mathcal{I}nd\mathcal{V}ar(t\,[\![c]\!]) \\ t\,[\![c]\!] \vdash_x t'\,[\![c']\!] \end{cases}$
Strongly Ground Reducible Leaf:	$\dfrac{t\,[\![c]\!]}{\textbf{success}}$ `if` $t\,[\![c]\!]$ is strongly ground reducible
Irreducible Leaf:	$\dfrac{t\,[\![c]\!]}{\textbf{failure}(t\,[\![c]\!])}$ `if` no other rule applies to $t\,[\![c]\!]$

Fig. 2. Inference rules for the construction of a pattern tree.

Termination, correctness, completeness.

Theorem 1 (Termination). *For every $f \in \mathcal{D}$, the size of dtree(f) is bounded.*

Proof. The number of rules of $\mathcal{R}_\mathcal{D}$ with the function symbol f at the top position is finite. It means that the set $\mathcal{I}nd\mathcal{P}os(f, \mathcal{R})$ is finite too. As a consequence, the size of non-ground terms with induction variables is also bounded, and the height of the pattern tree too, because since consecutive grafts in the same branch of the tree are labeled with deeper non-ground constrained terms. □

Theorem 2 (Soundness). *Assume that $\mathcal{R}$ is ground convergent. For each $f \in \mathcal{D}$, if all leaves of dtree(f) are* **success** *then f is sufficiently complete wrt $\mathcal{R}$.*

Proof. Let $f \in \mathcal{D}$ be a defined function symbol and suppose that all the leaves of *dtree*(f) are **success**. We want to prove that for all $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C})$, there exists $s \in \mathcal{T}(\mathcal{C})$ such that $f(t_1, \ldots, t_n) \xrightarrow[\mathcal{R}]{*} s$. Since, by hypothesis, $\mathcal{R}_\mathcal{C}$ is terminating, we may consider that $t_1 \ldots, t_n$ are $\mathcal{R}_\mathcal{C}$-irreducible (otherwise, they can be normalized under $\xrightarrow[\mathcal{R}_\mathcal{C}]{}$). By Lemma 1, it implies that there exist some non terminals $\llcorner u_1 \lrcorner, \ldots, \llcorner u_n \lrcorner$ of the grammar $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_\mathcal{C})$ such that:

$$f(x_1, \ldots, x_n)\,[\![x_1{:}\llcorner u_1 \lrcorner, \ldots, x_n{:}\llcorner u_n \lrcorner]\!] \vdash^* f(t_1, \ldots, t_n)\,[\![c]\!] \tag{4}$$

Note that the first term of the above derivation labels a root node of the pattern tree *dtree*(f). We proceed by induction on $t = f(t_1, \ldots, t_n)$ w.r.t. $\xrightarrow[\mathcal{R}]{}$, which is a well founded relation by hypothesis.

Assume that t is $\mathcal{R}$-irreducible (base case of the induction). Let $s\,[\![d]\!]$ be the first term without induction variables occurring in the above grammar derivation 4, and let τ be the ground substitution of $sol(d)$ such that $s\tau = t$ (τ exists by Lemma 1). We shall show that $s\,[\![d]\!]$ is not strongly ground reducible by $\mathcal{R}$. It implies that the inference Irreducible leaf applies and *dtree*(f) contains a leaf labeled with **failure**, a contradiction. Indeed, otherwise, by definition, there exist n rules (with $n > 0$) of $\mathcal{R}_\mathcal{D}$ $\Gamma_i \Rightarrow l_i \to r_i\,[\![c_i]\!]$, with $i \in [1..n]$, and n substitutions σ_i, such that $s = l_i\sigma_i$ and $d \wedge \neg c_i\sigma_i$ is unsatisfiable for all $i \in [1..n]$ and $\mathcal{R} \models_{\mathcal{I}nd} \Gamma_1\sigma_1\,[\![d \wedge c_1\sigma_1]\!] \vee \ldots \vee \Gamma_n\sigma_n\,[\![d \wedge c_n\sigma_n]\!]$.

Since $\tau \in sol(d)$, then for all $i \in [1..n]$, $\tau \in sol(c_i\sigma_i)$ (otherwise, $d \wedge \neg c_i\sigma_i$ would be satisfiable). Therefore, there exists $k \in [1..n]$, such that $\mathcal{R} \models \Gamma_k\sigma_k\tau$. This implies that for each equation $u = v$ in $\Gamma_k\sigma_k\tau$, we have $u \downarrow_\mathcal{R} v$ because $\mathcal{R}$ is ground confluent, hence t can be rewritten by $\Gamma_k \Rightarrow l_k \to r_k\,[\![c_k]\!]$, a contradiction.

If t is $\mathcal{R}$-reducible, say $t \xrightarrow[\mathcal{R}]{} t'$, then we can apply the induction hypothesis to t'. □

As a corollary, since there are only two kind of leaves, we can conclude that if $\mathcal{R}$ is not sufficiently complete over $\mathcal{C}$ wrt $\mathcal{R}$, then the inference system will end with a failure.

Corollary 1 (Refutational Completeness). *Assume that $\mathcal{R}$ is ground convergent. If $\mathcal{R}$ is not sufficiently complete, then there exists $f \in \mathcal{D}$ such that dtree(f) contains a leaf of the form* **failure**.

Theorem 3 (Completeness). *If $\mathcal{R}$ is sufficiently complete then for each $f \in \mathcal{D}$, all leaves of dtree(f) are* **success**.

Proof. Assume that $\mathcal{R}$ is sufficiently complete and suppose that there exists a node $t\,[\![c]\!]$ in *dtree*(f), for some $f \in \mathcal{D}$, to which the inference Irreducible Leaf can be applied. It means, by definition, that $t\,[\![c]\!]$ does not contain any induction variable and is not strongly ground reducible.

We show first that $t\,[\![c]\!]$ is weakly reducible by $\mathcal{R}$. By construction, $t\,[\![c]\!]$ is decorated, hence by Lemma 1, there exists $\tau \in sol(c)$ such that for all $x \in var(t)$, $x\tau$ is $\mathcal{R}$-irreducible. Moreover, by construction, $t\tau$ has the form $f(t_1, \ldots, t_n)$ where $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C})$ are all $\mathcal{R}_\mathcal{C}$-irreducible. Hence, $t\tau$ is $\mathcal{R}_\mathcal{D}$-reducible at root position because $\mathcal{R}$ is sufficiently complete. Therefore, by definition, $t\tau$ is a ground instance of some left-hand side ℓ of a rule $\Gamma \Rightarrow \ell \to r\,[\![c']\!] \in \mathcal{R}_\mathcal{D}$, say $t\tau = \ell\theta$, and moreover, $\neg c'\theta$ is unsatisfiable. Since by hypothesis t does not contain any induction variable and by definition ℓ is linear, t is an instance of ℓ, say $t = \ell\sigma$ (with $\theta = \sigma\tau$). Also, $c \wedge \neg c'\sigma$ is unsatisfiable, otherwise, $\neg c'\theta$ would be satisfiable, because $\tau \in sol(c)$ and $\theta = \sigma\tau$. Hence the following subset $\mathcal{L}$ of $\mathcal{R}_\mathcal{D}$ is not empty:

$$\mathcal{L} = \left\{ \Gamma_i \Rightarrow \ell_i \to r_i\,[\![c_i]\!] \;\middle|\; i \in [1..n],\ t = l_i\sigma_i \text{ and } c \wedge \neg c_i\sigma_i \text{ is unsatisfiable} \right\}$$

By hypothesis, $t\,[\![c]\!]$ is not strongly ground reducible by $\mathcal{R}$ hence we have:

$$\mathcal{R} \not\models_{\mathcal{I}nd} \Gamma_1\sigma_1\,[\![c \wedge c_1\sigma_1]\!] \vee \ldots \vee \Gamma_n\sigma_n\,[\![c \wedge c_n\sigma_n]\!]$$

It means that for all $k \in [1..n]$ and all ground substitution $\tau \in sol(c \wedge c_k\sigma_k)$, we have $\mathcal{R} \not\models \Gamma_k\sigma_k\tau$. Hence $t\tau$ is not reducible at the root by a rule of $\mathcal{L}$. Assume now $t\tau$ is reducible at the root position by a rule $\Gamma \Rightarrow \ell \to r\,[\![d]\!] \in \mathcal{R} \setminus \mathcal{L}$. It means that t in an instance of ℓ, which contradicts the hypothesis that the above rule is not in $\mathcal{L}$. Hence $t\tau$ is not reducible at the root position. Moreover, $t\tau$ cannot be reducible at another position as we have seen above (by construction, $t\tau = f(t_1, \ldots, t_n)$ where $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C})$ are all $\mathcal{R}_\mathcal{C}$-irreducible). This contradicts the hypothesis that $\mathcal{R}$ is sufficiently complete. □

As a corollary, we can conclude that if the inference system fails then $\mathcal{R}$ is not sufficiently complete.

Corollary 2 (Soudness of Disproof). *For each $f \in \mathcal{D}$, if there exists a leaf of the form* **failure** *in dtree(f) then f is not sufficiently complete wrt $\mathcal{R}$.*

7 Example: Sorted Lists

We consider now a complete example of axiomatization of sorted lists without repetition. We assume three sorts in the signature: $\mathcal{S}$: *Bool* for Booleans, *Natb* for natural numbers bounded by a constant ∞ and *Set* for sorted lists, and we assume the following constructor symbols with their respective profiles:

$$\mathcal{C} = \left\{ \begin{array}{lll} true : Bool, & 0 : Natb, & \emptyset : Set \\ false : Bool, & s : Natb \to Natb, & ins : Natb \times Set \to Set \\ & \infty : Natb & \end{array} \right\}$$

and a constructor rewrite system for ordered lists,

$$\mathcal{R}_\mathcal{C} = \left\{ \begin{array}{l} ins(x, ins(y, z)) \to ins(x, z)\,[\![x \approx y]\!], \\ ins(x, ins(y, z)) \to ins(y, ins(x, z))\,[\![x \succ y]\!] \end{array} \right\}$$

The ordering constraint $\succ$ is interpreted as a reduction ordering total on ground terms. Note that $\mathcal{R}_\mathcal{C}$ is terminating thanks to the constraint of the second rule.

Let us complete this signature with the following defined function symbols:

$$\mathcal{D} = \big\{\in\!: \mathit{Natb} \times \mathit{Set} \to \mathit{Bool}, \mathit{sorted} : \mathit{Set} \to \mathit{Bool}, \mathit{min} : \mathit{Set} \to \mathit{Bool}\big\}$$

$$\mathcal{R}_{\mathcal{D}} = \left\{ \begin{array}{ll} x \in \emptyset \to \mathit{false}, & \\ x \in \mathit{ins}(y,z) \to \mathit{true}\,[\![x \approx y]\!], & \mathit{sorted}(\emptyset) \to \mathit{true}, \\ x \in \mathit{ins}(y,z) \to x \in z\,[\![x \not\approx y]\!], & \mathit{sorted}(\mathit{ins}(x,\emptyset)) \to \mathit{true}, \\ \mathit{sorted}(\mathit{ins}(y,z)) = \mathit{true} \Rightarrow \mathit{sorted}(\mathit{ins}(x,\mathit{ins}(y,z))) \to \mathit{true}\,[\![x \prec y]\!], & \\ \mathit{min}(\mathit{ins}(x,\emptyset) \to x, \quad \mathit{min}(\mathit{ins}(x,\mathit{ins}(y,z))) \to \mathit{min}(\mathit{ins}(x,z))\,[\![x \prec y]\!] & \end{array} \right\}$$

The induction positions of defined functions are respectively: $\mathcal{I}\mathit{ndPos}(\mathit{sorted},\mathcal{R}) = \{1,\ 1.2\}$, $\mathcal{I}\mathit{ndPos}(\in,\mathcal{R}) = \{2\}$, $\mathcal{I}\mathit{ndPos}(\mathit{min},\mathcal{R}) = \{1,\ 1.2\}$.

Following the construction of Section 3, the constrained tree grammar $\mathcal{G}_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}})$ which generates the set of $\mathcal{R}_{\mathcal{C}}$-irreducible ground constructor terms has the following set of non-terminals: $Q_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}}) = \{\llcorner x^{\mathit{Bool}}\lrcorner, \llcorner x^{\mathit{Natb}}\lrcorner, \llcorner x^{\mathit{Set}}\lrcorner, \llcorner \mathit{ins}(x_1,x_2)\lrcorner\}$, (for sake of clarity, $\llcorner x^{\mathit{Set}}\lrcorner$ is denoted $\llcorner\emptyset\lrcorner$ below) and the following set of production rules:

$$\Delta_{\mathrm{NF}}(\mathcal{R}_{\mathcal{C}}) = \left\{ \begin{array}{llll} \llcorner x^{\mathit{Bool}}\lrcorner & \vdash & \mathit{true}, & \llcorner x^{\mathit{Bool}}\lrcorner \vdash \mathit{false}, \\ \llcorner x^{\mathit{Natb}}\lrcorner & \vdash & 0, & \llcorner x^{\mathit{Natb}}\lrcorner \vdash \infty, \\ \llcorner x^{\mathit{Natb}}\lrcorner & \vdash & s(\llcorner x^{\mathit{Natb}}\lrcorner), & \llcorner\emptyset\lrcorner \vdash \emptyset, \\ \llcorner \mathit{ins}(x_1,x_2)\lrcorner & \vdash & \mathit{ins}(\llcorner x^{\mathit{Natb}}\lrcorner, \llcorner\emptyset\lrcorner), & \\ \llcorner \mathit{ins}(x_1,x_2)\lrcorner & \vdash & \mathit{ins}(\llcorner x^{\mathit{Natb}}\lrcorner, \llcorner \mathit{ins}(x_1,x_2)\lrcorner)\,[\![x^{\mathit{Natb}} < x_1]\!] & \end{array} \right\}$$

Let us now check the sufficient completeness of each of the symbols of $\mathcal{D}$ wrt $\mathcal{R}$. In the rest of the example, we shall use the same simplified notation for decorated terms as in Section 5, where the non-terminals are written inside the terms instead than in the constraints.

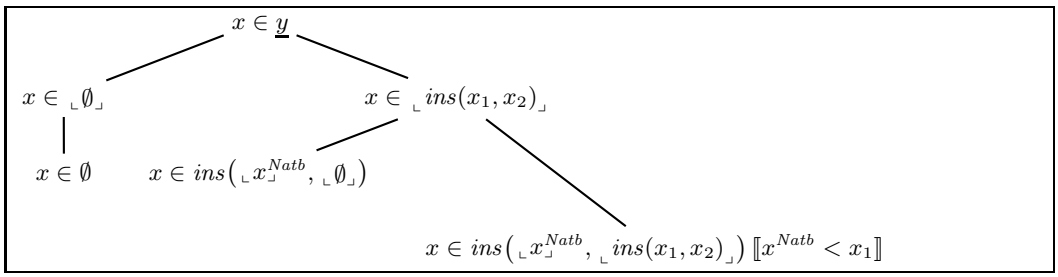

Fig. 3. The pattern tree of $\in$

The pattern tree of Figure 3 shows that the function $\in$ is completely defined. The term $x \in \emptyset$ is $\mathcal{R}_{\mathcal{D}}$-reducible and the terms $x \in \mathit{ins}(\llcorner x^{\mathit{Natb}}\lrcorner, \llcorner\emptyset\lrcorner)$, and $x \in \mathit{ins}(\llcorner x^{\mathit{Natb}}\lrcorner, \llcorner \mathit{ins}(x_1,x_2)\lrcorner)\,[\![x^{\mathit{Natb}} < x_1]\!]$, which are respective abbreviations for $x \in \mathit{ins}(y_1,y_2)\,[\![y_1{:}\llcorner x^{\mathit{Natb}}\lrcorner \wedge y_2{:}\llcorner\emptyset\lrcorner]\!]$ and

$$x \in \mathit{ins}(y_1,y_2)\,[\![y_1{:}\llcorner x^{\mathit{Natb}}\lrcorner \wedge\ y_2{:}\llcorner \mathit{ins}(x_1,x_2)\lrcorner \wedge x^{\mathit{Natb}} < x_1]\!]$$

are strongly ground reducible wrt $\mathcal{R}_{\mathcal{D}}$ since the formula $x = y_1 \wedge x \neq y_1$ is unsatisfiable.

The pattern tree in Figure 4 shows that the function *sorted* is completely defined. Note that the terms $\mathit{sorted}(\emptyset)$ and $\mathit{sorted}(\mathit{ins}(\llcorner x^{\mathit{Natb}}\lrcorner, \emptyset))$ (abbreviation for $\mathit{sorted}(\mathit{ins}(y_1,\emptyset))\,[\![y_1{:}\llcorner x^{\mathit{Natb}}\lrcorner]\!]$) are $\mathcal{R}_{\mathcal{D}}$-reducible and the terms:

$$\mathit{sorted}(\mathit{ins}(\llcorner x^{\mathit{Natb}}\lrcorner, \mathit{ins}(\llcorner y^{\mathit{Natb}}\lrcorner, \llcorner\emptyset\lrcorner)))\,[\![x^{\mathit{Natb}} < y^{\mathit{Natb}}]\!]$$

which is an abbreviation for:

$$\mathit{sorted}(\mathit{ins}(y_1,\mathit{ins}(y_2,y_3)))\,[\![y_1{:}\llcorner x^{\mathit{Natb}}\lrcorner \wedge y_2{:}\llcorner y^{\mathit{Natb}}\lrcorner \wedge y_3{:}\llcorner\emptyset\lrcorner \wedge x^{\mathit{Natb}} < y^{\mathit{Natb}}]\!]$$

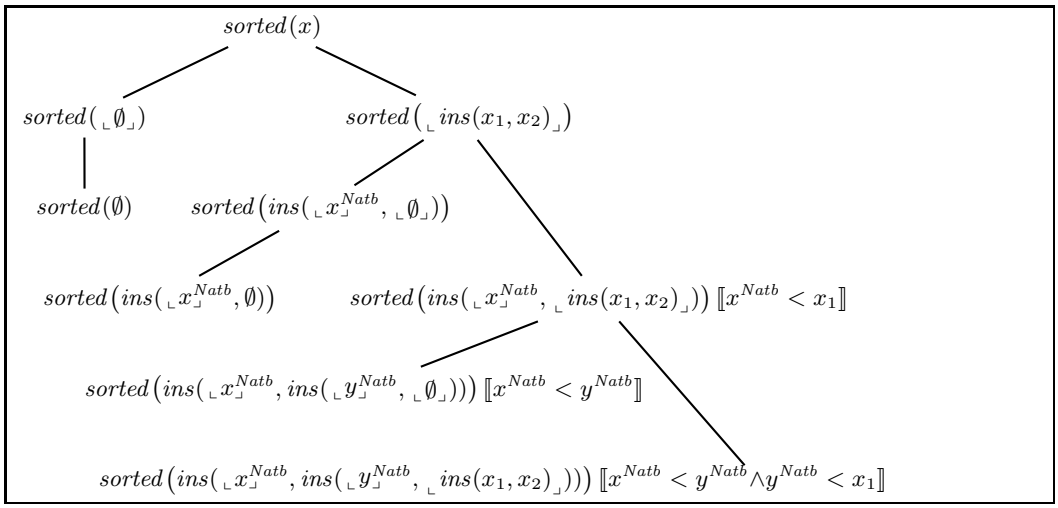

Fig. 4. The pattern tree of *sorted*

and:

$$sorted(ins(\llcorner x\lrcorner^{Natb}, ins(\llcorner y\lrcorner^{Natb}, \llcorner ins(x_1, x_2)\lrcorner)))\ [\![x^{Natb} < y^{Natb} \wedge y^{Natb} < x_1]\!]$$

which is an abbreviation for:

$$sorted(ins(y_1, ins(y_2, y_3)))\ \left[\!\!\left[\begin{array}{lll} y_1{:}\llcorner x\lrcorner^{Natb} \wedge & y_2{:}\llcorner y\lrcorner^{Natb} & \wedge\ y_3{:}\llcorner ins(x_1, x_2)\lrcorner \\ & \wedge\ x^{Natb} < y^{Natb} \wedge & y^{Natb} < x_1 \end{array} \right]\!\!\right]$$

are strongly ground reducible since the two following conjectures are inductive theorems of $\mathcal{R}$, and can be proved using the method of [BJ05].

$$(\dagger)\quad sorted(ins(y_2, y_3)) = true\ [\![y_2{:}\llcorner x\lrcorner^{Natb} \wedge y_3{:}\llcorner\emptyset\lrcorner]\!] \text{ and}$$

$$(\ddagger)\quad sorted(ins(y_2, y_3)) = true\ [\![y_2{:}\llcorner x\lrcorner^{Natb} \wedge y_3{:}\llcorner ins(x_1, x_2)\lrcorner \wedge x^{Natb} < x_1]\!]$$

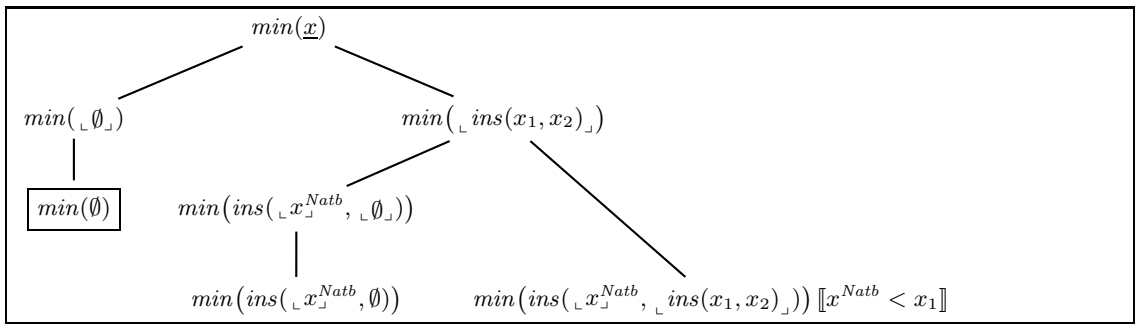

Fig. 5. The pattern tree of *min*

The function *min* is not completely defined, as shown by the pattern tree of Figure 5. The term $min(\emptyset)$ labeling the failing leaf is not reducible by $\mathcal{R}$. This suggest to add to $\mathcal{R}_{\mathcal{D}}$ an axiom for the simplification of $min(\emptyset)$, for instance: $min(\emptyset) \to \infty$. Note that the terms: $min(ins(\llcorner x\lrcorner^{Natb}, \emptyset))$ and $min\big(ins(\llcorner x\lrcorner^{Natb}, \llcorner ins(x_1, x_2)\lrcorner)\big)\ [\![x^{Natb} < x_1]\!]$ are $\mathcal{R}$-reducible.

The methods of [BR90,Bou96,BJ01,BJM00] cannot be applied to prove the sufficient completeness of *sorted* and *min* since the axioms for constructors are constrained and *non left-linear*. We could imagine a straightforward adaptation of methods based on cover sets to *constrained cover sets* for sorted lists, like $\{\emptyset, ins(x,\emptyset), ins(x, ins(y,z))\,[\![x < y]\!]\}$. This also fails. The reason is that this representation of $\mathcal{R}_{\mathcal{C}}$-irreducible ground constructor terms is still not exact. For instance $ins(0, ins(s(0), 0))$ is an instance of $ins(x, ins(y,z))\,[\![x < y]\!]$ but is not irreducible.

8 Conclusion

We have proposed a method for testing sufficient completeness of constrained and conditional rewrite systems with constrained rules for constructors. Our procedure uses a tree grammar with constraints which describes the exact set of ground constructor terms in normal form and relies on a method for inductive theorem proving based on the same framework [BJ05]. It is sound for ground confluent and terminating TRS and also complete, and has been successfully used manually for checking sufficient completeness of several specifications where related techniques fail. Moreover, in case of disproof, *i.e.* when the specification is not sufficiently complete, our procedure proposes candidates left hand sides and constraints and a hint for conditions of rewrite rule to complete it.

We are planning to implement the procedure presented on this paper, based on a forthcoming system for [BJ05] generalizing Spike [BR95] and on an efficient library for tree automata with constraints.

We are currently developing a technique for checking ground confluence for conditional constrained rewrite systems using constrained tree grammars in the same framework as in this paper and [BJ05].

References

[BJ01] Adel Bouhoula and Jean-Pierre Jouannaud. Automata-driven automated induction. *Information and Computation*, 169(1):1–22, 2001.

[BJ05] Adel Bouhoula and Florent Jacquemard. Automated induction for complex data structures. Research Report LSV-05-11, Laboratoire Spécification et Vérification, ENS Cachan, France, July 2005. 24 pages.

[BJM00] Adel Bouhoula, Jean-Pierre Jouannaud, and Jose Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236(1-2):35–132, 2000.

[Bou96] Adel Bouhoula. Using induction and rewriting to verify and complete parameterized specifications. *Theoretical Computer Science*, 170(1-2):245–276, 1996.

[BR90] Wadoud Bousdira and Jean-Luc Remy. On sufficient completeness of conditional specifications. In Stéphane Kaplan and Mitsuhiro Okada, editors, *CTRS*, volume 516 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 1990.

[BR95] Adel Bouhoula and Michaël Rusinowitch. Spike: A system for automatic inductive proofs. In *Proceedings of the 4th International Conference on Algebraic Methodology and Software Technology (AMAST)*, volume 936 of *Lecture Notes in Computer Science*, pages 576–577. Springer, 1995.

[CGJ+02] Hubert Comon, Max Dauchet Rémy Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. `http://www.grappa.univ-lille3.fr/tata`, 2002.

[CJ03] Hubert Comon and Florent Jacquemard. Ground reducibility is exptime-complete. *Information and Computation*, 187(1):123–153, 2003.

[Com86] Hubert Comon. Sufficient completness, term rewriting systems and "anti-unification". In Jörg H. Siekmann, editor, *CADE*, volume 230 of *Lecture Notes in Computer Science*, pages 128–140. Springer, 1986.

[Der83] N. Dershowitz. Well-founded orderings. In *Information Science Research Office, ATR-83-8478-3*, The Aerospace Corporation, El Segundo, California USA, 1983.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 243–320. 1990.

[GH78] John V. Guttag and James J. Horning. The algebraic specification of abstract data types. *Acta Inf.*, 10:27–52, 1978.

[Gut75] John V. Guttag. *The Specification and Application to Programming of Abstract Data Types*. Phd thesis, University of Toronto, Computer Science Department, 1975. Report CSRG-59.

[Gut78] John V. Guttag. Notes on type abstraction. In Friedrich L. Bauer and Manfred Broy, editors, *Program Construction*, volume 69 of *Lecture Notes in Computer Science*, pages 593–616. Springer, 1978.

[HCM05] Joe Hendrix, Manuel Clavel, and José Meseguer. A sufficient completeness reasoning tool for partial specifications. In *Proceedings ot the 16th International Conference on Term Rewriting and Applications (RTA)*, volume 3467 of *Lecture Notes in Computer Science*, pages 165–174. Springer, 2005.

[HH82] Gérard P. Huet and Jean-Marie Hullot. Proofs by induction in equational theories with constructors. *J. Comput. Syst. Sci.*, 25(2):239–266, 1982.

[Kap94] Deepak Kapur. An automated tool for analyzing completeness of equational specifications. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, volume Special Issue of *Software Engineering Notes*, pages 28–43. ACM Press, 1994.

[KNRZ91] Deepak Kapur, Paliath Narendran, Daniel J. Rosenkrantz, and Hantao Zhang. Sufficient-completeness, ground-reducibility and their complexity. *Acta Informatica*, 28(4):311–350, 1991.

[KNZ87] Deepak Kapur, Paliath Narendran, and Hantao Zhang. On sufficient-completeness and related properties of term rewriting systems. *Acta Informatica*, 24(4):395–415, 1987.

[Kou85] Emmanuel Kounalis. Completeness in data type specifications. In B. F. Caviness, editor, *European Conference on Computer Algebra (2)*, volume 204 of *Lecture Notes in Computer Science*, pages 348–362. Springer, 1985.

[LLT90] Azeddine Lazrek, Pierre Lescanne, and Jean-Jacques Thiel. Tools for proving inductive equalities, relative completeness, and omega-completeness. *Information and Computation*, 84(1):47–70, January 1990.