Mechanizing inductive reasoning

Emmanuel Kounalis and Michaël Rusinowitch
CRIN, 54506 Vandœuvre les Nancy, BP239 (France)
e-mail: {kounalis,rusi}@loria.fr

Abstract

Automating proofs by induction is important in many computer science and artificial intelligence applications, in particular in program verification and specification systems. We present a new method to prove (and disprove) automatically inductive properties. Given a set of axioms, a well-suited induction scheme is constructed automatically. We call such a scheme a test-set. Then, for proving a property, we just instanciate it with terms from the test-set and apply pure algebraic simplification to the result. This method avoids completion and explicit induction. However it retains their positive features, namely the completeness of the former and the robustness of the latter.

1 Introduction

Inductive reasoning consists in performing inferences in domains where there exists a natural well-founded relation on the objects. It is fundamental when proving properties of numbers, data-structures or programs axiomatized by a set of conditional axioms. As opposed to deductive theorems, inductive theorems are usually valid only in some particular models of the axioms, for instance Herbrand models or the initial model, which fits nicely the semantics of data-type specifications, logic and functional programming.

As everybody knows from his experience, it might be difficult, not only to find an appropriate well-founded relation to support inductive inferences, but also to guess suitable induction hypothesis. Two main approaches have been proposed to overcome these difficulties. The first applies explicit induction arguments on the structure of terms [1,3,2,4,14]. The second one involves a proof by consistency: this is the *inductionless induction* method [10,5,6]. However, both methods have many limitations either on the theorems to be proved or on the underlying theory. For instance, explicit induction techniques is unable to provide us automatically with induction schemes, and cannot help to disprove false conjectures. On the other hand, the inductionless induction technique often fails where explicit induction succeeds. Moreover, there does not exist any realistic inductionless induction procedure for conditional theories.

In this paper, we present an alternative proof system for automatizing inductive reasoning in theories defined by conditional axioms. We show how to prove (and disprove) equations and more generally clauses in the initial model and Herbrand models respectively. Our method combines the full power of explicit induction and inductionless induction. It is refutationally complete in the following sense : any positive clause which is not valid in the initial model will be disproved in finite time, provided that no negative literals are introduced by the procedure. This method relies on the notion of *test-set* (which, in essence, is a finite description of the initial model) and applies only pure algebraic simplification. The key-idea of the simplification strategy is to use axioms, previously proved conjectures, and instances of the conjecture itself as soon as they are smaller than the currently examined proposition with respect to a well-founded relation. This last point captures the notion of Induction Hypothesis in the proof by induction paradigm. The refutational aspect of our procedure requires a convergence property of the axiomatization and, also, suitable test-sets. The convergence can be obtained either by a Knuth-Bendix like procedure [9] or semantic techniques specific to hierarchical axiomatizations(see [12] and section 5.2 of this paper). On the other side, building a test-set requires itself some theorem proving. Whereas the computation of test-sets is generally undecidable, in the last section, we propose a method to obtain test-sets in conditional theories over a free set of constructors. Our method can also be viewed as a **real automatization** of explicit induction: indeed the test-set computation yields automatically induction schemes which are well-adapted to the axioms. In addition, we show how the method applies to proofs of properties of some recursive programs and elementary arithmetic.

2 Overview of our approach: an example

Before discussing the technical details of the method we propose for mechanizing proofs of inductive theorems, we first describe our inference system on a simple exam-

ple, namely positive integers with cut-off and *gcd* functions and the less predicate. The arrow $\rightarrow$ just indicates how to apply a (conditional) equation for simplification:

(1) $x - 0 \rightarrow x$
(2) $0 - x \rightarrow 0$
(3) $s(x) - s(y) \rightarrow x - y$
(4) $(0 < s(x)) \rightarrow tt$
(5) $(x < 0) \rightarrow ff$
(6) $x < y = tt \Rightarrow s(x) < s(y) \rightarrow tt$
(7) $x < y = ff \Rightarrow s(x) < s(y) \rightarrow ff$
(8) $x < y = tt \Rightarrow gcd(x, y) \rightarrow gcd(y - x, y)$
(9) $x < y = ff \Rightarrow gcd(x, y) \rightarrow gcd(x - y, x)$
(10) $gcd(x, 0) \rightarrow x$
(11) $gcd(0, x) \rightarrow x$

Consider the conjectures:

(12) $x - x = 0$
(13) $x < x = ff$
(14) $x < s(x) = tt$
(15) $x < y = tt \vee x < y = ff$
(16) $x < y = ff \vee y < z = ff \vee x < z = tt$
(17) $gcd(x, x) = x$
(18) $gcd(x, y) = gcd(y, x)$
(19) $x < s(x) = ff$
(20) $x < y = ff \vee y < x = tt$

Except the two last ones all these propositions are valid in the standard arithmetic:note that 13,15 and 16 state that $<$ is a total ordering on integers, and 18 is the commutativity of *gcd*. This suggests to prove them by induction. With our method the first step consists in computing a test-set (see def. 4.1). By using techniques of section 6,we get the test-set $\{0, s(x), tt, ff\}$. The next step consists in replacing variables of the conjecture by the elements of the test-set and checking these instances by using pure simplification. *The simplification strategy may use axioms, previously proved conjectures, and instances of the conjecture itself as soon as they are smaller (w.r.t. a noetherian relation which contains the rewriting relation) than the currently examined proposition. This last point captures the notion of Induction Hypothesis in the proof by induction paradigm (see th. 4.1,4.2).* For the equation 12, two instances need to be checked: $0 - 0 = 0$ and $s(x) - s(x) = 0$. The first one reduces immediately to a trivial identity. For the second one consider the reduction (notice the use of 12 as an induction hypothesis): $s(x) - s(x) \rightarrow_3 x - x \rightarrow_{12} 0$ For 13, the only non-trivial instance is $s(x) < s(x) = ff$. However, $s(x) < s(x) \rightarrow_{13,7} ff$. For the last derivation, we have used an induction hypothesis to satisfy the condition of 7 (cf th. 4.1). For 14, the same argument can be employed. For 15, there are four instances by terms from the test-set. The only non-trivial case is $s(x) < s(y) = tt \vee s(x) < s(y) = ff$. By using case rewriting (cf def. 3.3), we can split this formula into the conjunction:

(21) $\neg x < y = tt \vee tt = tt \vee s(x) < s(y) = ff$
(22) $x < y = tt \vee s(x) < s(y) = ff$

Now, 21 is trivial and 22 is split again in:

(23) $x < y = tt \vee \neg x < y = ff \vee ff = ff$
(24) $x < y = tt \vee x < y = ff \vee s(x) < s(y) = ff$

23 is trivial again and 24 is subsumed by the initial conjecture: this is the induction step (cf th. 4.2). The 16,17,18 are proved exactly in the same way. Consider now 19. To disprove it, we are going to use the convergence properties of the initial system. Note that the saturation technique (see section 5) can prove the convergence property for 1-7 and th. 5.1 for the whole axiomatization. The instances to be considered are : $0 < s(0) = ff, s(x) < s(s(x)) = ff$ The first one reduces to $tt = ff$ whose members are irreducible and different. By th. 4.3 the conjecture is false. For 20, consider the following instance $0 < s(x) = ff \vee s(x) < 0 = tt$. It can be reduced to $tt = ff \vee ff = tt$ and therefore 20 is not valid.

3 Preliminaries

3.1 Definitions and notations

Let F be a signature of function symbols, and X a set of variables. We shall denote by $T(F, X)$ the set of terms built from F and X. We write $s[t]_n$ to mean that the term t is a subterm of s at position n. The set of ground terms is denoted by $T(F)$. A conditional equation is an equation or an expression of one of the types: $e_1 \wedge \ldots \wedge e_n \Rightarrow e$ or $e_1 \wedge \ldots \wedge e_n \Rightarrow$ or $\Rightarrow$ where $e, e_1, \ldots e_n$ are equations, $e_1, \ldots e_n$ are *conditions*, a positive literal e in a conditional equation is a *conclusion* and $\Rightarrow$ is the empty clause. In this paper, axiomatizations are built from conditional equations and goals to be proved are clauses (i.e. disjunction of equational literals, since $=$ is the only predicate, here [1]). Given a binary relation $\rightarrow$, $\rightarrow^*$ denotes its reflexive transitive closure. Given two binary relations, R, S, RoS denotes their composition. A relation R is noetherian if there is no infinite sequence $t_1 \; R \; t_2 \; R \; \ldots$. In the following, we suppose given a *reduction ordering* $\succ$ on the set of terms, that is, a transitive irreflexive relation which is noetherian, monotonic ($s \succ t$ implies $w[s] \succ w[t]$) and stable ($s \succ t$ implies $s\sigma \succ t\sigma$). A reduction ordering can be extended to literals by comparing the multiset of their members with the multiset extension of $\succ$. Formula are compared by using the multiset extension of this last ordering to the multiset of their atomic subformulas. Since there is no ambiguity, all these extensions will also be denoted by $\succ$. An equation $s = t$ will be written $s \rightarrow t$ if for all ground substitution $s\sigma \succ t\sigma$; in that case we say that the equation is *orientable*.

[1] we shall identify a conditional equation and its corresponding representation as a clause

3.2 Inductive theorems

Given a set of conditional equations Ax on the signature F, we recall that a Herbrand model of Ax is a model of Ax whose domain is $T(F)$ (axioms for equality are implicitly assumed to be valid, too). A formula $\mathcal{F}$ is a *deductive theorem of* Ax if it is valid in any model of Ax. This will be denoted by $Ax \models \mathrm{F}$. The notion of inductive theory can be related to several kinds of models: Herbrand models or initial models (i.e least Herbrand models). These relations are discussed in [11]. We have chosen to study the initial and Herbrand model approaches:

Definition 3.1 *Let H be a set of conditional equations on the signature F. A clause e is an* inductive theorem of H *iff for any ground substitution σ, $H \models e\sigma$.*

For clauses, validity in all Herbrand models differs, in general, from validity in the initial model. However these two notions of validity coincide for unconditional equations as it is proved in [11].

3.3 Conditional rewriting

The idea of rewriting is to impose a direction when using equations in proofs. This direction is indicated by an arrow when it is independent from the instanciation: $l \rightarrow r$ means that, we can replace l by r in some context. When an instance of a conditional equation is orientable and has a valid conditional part, it can be applied as a rule. The conditions are checked by a recursive call to the theorem-prover. Termination of such calls is ensured by requiring the conditions to be smaller (w.r.t. the reduction ordering $\succ$) than the conclusion:

Definition 3.2 *Let H be a set of clauses. Let A be a term or a clause, and n an occurrence of A. Then $A[s\sigma]_n \rightarrow_H A[t\sigma]_n$ if σ is a substitution and there is a conditional equation $c \Rightarrow s = t$, in H such that $s\sigma \succ t\sigma$ and $H \models c\sigma$ and $(s = t)\sigma \succ c\sigma$..*

A term A is reducible w.r.t. $\rightarrow_H$, if there is a term B such that $A \rightarrow_H B$. Otherwise we say that A is H-irreducible. The system H will be qualified as *convergent* if for all ground terms a,b such that $H \models a = b$ there exists a ground term c such that $a \rightarrow_H{}^* c$ and $b \rightarrow_H{}^* c$. One can easily see that it is also equivalent to the property that every ground term possesses a unique irreducible form. Note that the conditional rewriting relation may be undecidable. The relation $\rightarrow_H$ will be extended to sets of clauses in a natural way: by definition, $S \cup \{c\} \rightarrow_H S \cup \{d\}$ whenever $c \rightarrow_H d$.

3.4 Case rewriting

Case reasoning is a very powerful technique which is the basis of many theorem proving strategies. It is a most important rule in the context of inductive theorem proving where case splitting arises naturally from induction hypothesis. We propose here a notion of case rewriting which is well-suited to inductive reasoning.

Definition 3.3 *Let H be a set of conditional equations and $c \Rightarrow s = t$ a conditional equation in H. Let $A[s\sigma]_n$ be a clause (where σ is a substitution) and let S be a set of clauses. The case rewriting rule can be stated as follows*

$$S \cup \{A[s\sigma]_n\} \rightarrow_H S \cup \{(c\sigma \vee A[s\sigma]_n), (\neg c\sigma \vee A[t\sigma]_n)\}$$

if $c\sigma$ is not a subclause of $A[s\sigma]_n$, n occurs in a maximal literal of A, $s\sigma \succ t\sigma$ and $(s = t)\sigma \succ c\sigma$

Let us denote $\rightarrow_H \cup \rightarrow_H$ by $\hookrightarrow_H$. The following propositions is the base for proving (or disproving) clausal theorems.

Proposition 3.1 *The case rewriting rule is sound (the derived set of clauses is logically equivalent to the initial set). The relation $\hookrightarrow_H$ is noetherian.*

4 How to prove and disprove inductive theorems

In this section, we propose general methods to prove (or disprove) automatically that clauses are inductive consequences of theories axiomatized by a set of conditional equations. These techniques allow us to replace inductive reasoning by pure simplification. Such a mechanization of inductive proofs is based on the notion of *test-set*, which, in essence, provides us with a finite description of the initial model.

4.1 Test sets

First, let us define the height of a term as the height of the tree representation of this term. The height of a set of conditional equations will be the maximal height of the terms occurring in this set. The height of an object x will be denoted by $|x|$.

Definition 4.1 *A* **test-set for a set of conditional equations H** *is a finite subset $S(H)$ of $T(F, X)$ such that the following properties hold:*

completeness: *for any H-irreducible ground term s, there exists a term t in $S(H)$ and a substitution σ such that $t\sigma = s$.*

soundness: *for any term t in $S(H)$ there exists an H-irreducible ground term s and a substitution σ such that $t\sigma = s$.*

transnormality: *every non-ground term in $S(H)$ has infinitely many ground instances which are H-irreducible.*

coveredness: *any non-ground term t in $S(H)$, $|t| \geq l$ where $l = |H| - 1$ if every variable in the left-hand side of the positive literals in H occurs once and*

$l = |H|$ if H contains a rule whose left-hand side has multiple occurrences of the same variable.

Definition 4.2 *A* **test-substitution w.r.t. S(H)** *is a substitution which applies every variable to an element of the test-set $S(H)$*

Example 4.1 *Let us come back to the introductory example. Let H be the set of axioms $1, 2, \ldots, 10$. As we pointed out, $S(H) = \{0, s(x), tt, ff\}$ may be considered as a test-set. Note that the four properties of the definition are verified.*

The construction of test-sets for equational theories is decidable and may be performed in relatively efficient way. The algorithm is based on pumping lemmas in tree languages and is detailed in [8]. Such an algorithm does not exist for conditional theories. However, in the last section, we shall give a method to derive test-sets in theories defined over a free set of constructors.

4.2 Inductive proofs by simplification

Our notion of induction refers to a noetherian ordering on ground terms, which contains the conditional rewriting relation. **We can use as an inductive hypothesis any instance of the theorem we want to prove, as soon as this instance is smaller (w.r.t. $\succ$)than the one that is currently considered.** In Theorem 4.1 we propose two rewriting relations which are sound, with regard to the use of induction hypothesis. Both of them allow to utilize the conjecture after a first *normal* simplification step has been applied. However, if the first step is conditional, we can also use the conjecture when attempting to satisfy the conditions of the rule (case a). The following notations will be useful:

Definition 4.3 *Let $u = v$ be an equation, then $\overset{u=v}{\longleftrightarrow}$ denotes the symmetric closure of $\rightarrow_{\{u=v\}}$ and $\rightarrow_{H/u=v}$ denotes the relation $(\overset{u=v}{\longleftrightarrow})^* o(\rightarrow_H) o(\overset{u=v}{\longleftrightarrow})^*$. Let H be a set of conditional equations, let A be a term and n an occurrence of A. Then we write $A[s\sigma]_n \rightarrow_{H[u=v]} A[t\sigma]_n$ if σ is a substitution and there is a conditional equation $a_1 = b_1 \wedge \cdots \wedge a_n = b_n \Rightarrow s = t$ in H, such that:*
1. $s\sigma \succ t\sigma$ and $(s = t)\sigma \succ (a_1 = b_1 \wedge \cdots \wedge a_n = b_n)\sigma$
2. $\forall i, \exists c \;\; a_i\sigma \rightarrow_{H \cup \{u=v\}}{}^* c$ and $b_i\sigma \rightarrow_{H \cup \{u=v\}}{}^* c$

The following theorem shows how to prove equations in the initial model of conditional theories.

Theorem 4.1 (see [9]) *Let H be a set of conditional equations, $S(H)$ a test-set, and $u = v$ an equation. We suppose that one of the following hypothesis is verified:*

a. *$\rightarrow_{H \cup \{u=v\}}$ is noetherian. In this case we define $\rightsquigarrow$ as the reflexive closure of the relation: $(\rightarrow_{H[u=v]}) o (\rightarrow_{H \cup \{u=v\}})^*$*

b. *$\rightarrow_{H/u=v}$ is noetherian. In this case we define $\rightsquigarrow$ as the reflexive closure of the relation: $(\rightarrow_H) o (\rightarrow_{H/u=v})^*$*

If, for all test-substitution ν there is a term α such that $u\nu \rightsquigarrow \alpha$ and $v\nu \rightsquigarrow \alpha$ then $u = v$ is an inductive theorem for H (and therefore is valid in the initial model of H by proposition 3.2).

Example 4.2 *Consider the following conditional axioms for integers with $+$, odd and even.*

(24) $x + 0 \rightarrow x$
(25) $x + s(y) \rightarrow s(x + y)$
(26) $even(0) \rightarrow tt$
(27) $even(s(0)) \rightarrow ff$
(28) $even(s(s(x))) \rightarrow even(x)$
(29) $even(x) = tt \Rightarrow odd(s(x)) \rightarrow tt$
(30) $even(x) = ff \Rightarrow odd(s(x)) \rightarrow ff$

Here the test-set is $\{0, s(0), s(s(z)), tt, ff\}$. Let us prove first the commutativity of $+$. We apply case b. of th. 4.1. We just consider the non-trivial case, which is an instantiation by the last scheme: $s(s(x)) + s(s(y)) = s(s(y)) + s(s(x))$. After simplification, we have to consider the goal: $s(s(s(x)) + y)) = s(s(s(y)) + x))$. Commutativity can be applied strictly inside *the equation, since it is supposed to be true for smaller instances (induction hypothesis). We get: $s(s(y + s(s(x)))) = s(s(x + s(s(y))))$ and then, simplification finishes the job. By assuming now the commutativity of $+$, we can prove in the same way : $odd(x + s(x)) = tt$. First, $odd(s(s(x)) + s(s(s(x)))) \rightsquigarrow odd(s(s(s(s(s(x+x)))))) \rightsquigarrow tt$. To justify the last rewriting step we need to prove as a lemma $even(s(s(s(s(x + x))))) = tt$ or its simplified form $even(x + x) = tt$. This is achieved by the same technique.*

It is straightforward to generalize the previous method to proving that clauses are inductive theorems. However, in this general situation, case analysis is crucial:

Theorem 4.2 *Let H be a set of conditional equations, $S(H)$ a test-set, and C a clause. If, for all test-substitution ν, $\{C\nu\} \hookrightarrow_H{}^* \{p_1, p_2 \cdots, p_n\}$, and every clause p_j is either a tautology (contains two complementary literals or an instance of $x=x$) or is subsumed by an axiom or contains an instance of C which is strictly smaller w.r.t. $\succ$ than $C\nu$, then C is an inductive theorem of H*

Example 4.3 *Let us prove now the transitivity of $<$ (see axioms in the introductory example): $x < y = ff \vee y < z = ff \vee x < z = tt$. The only non-trivial instance by a test-substitution among the eight of them is: $s(x) < s(y) = ff \vee s(y) < s(z) = ff \vee s(x) < s(z) = tt$. After three steps of case-rewriting, we get only one clause which is not a tautology, namely:*

$$x < y = ff \vee y < z = ff \vee x < z = tt \vee s(x) < s(y) = ff \vee s(y) < s(z) = ff \vee s(x) < s(z) = tt$$

This clause contains a subclause which is a strictly smaller instance of the one to be proved. Hence by th. 4.2, the proof is achieved. In the same way we could prove in the example 4.2 that $even(x) = tt \vee odd(x) = tt$.

4.3 Disproving inductive theorems

The notion of test-set is particularly useful for refuting inductive properties. The next definition provides us with criteria to reject such conjectures.

Definition 4.4 *We suppose that we are given a set of conditional equations* H, *and a test-set* $S(H)$. *Let* H' *be the set of positive literals of* H. *Then, a clause* $\neg e_1 \vee \ldots \vee \neg e_m \vee g_1 = d_1 \vee \ldots \vee g_n = d_n$ *is* **quasi-inconsistent** *with respect to* H *if there is a test-substitution* σ *such that, for all* $i \leq m$, $e_i\sigma$ *is an inductive theorem and for all* $j \leq n$ *at least one of the following is verified:*

- $g_j\sigma \not\equiv d_j\sigma$ *and* $g_j\sigma$ *and* $d_j\sigma$ *are irreducible by* H'.
- $g_j\sigma \succ d_j\sigma$ *and* $g_j\sigma$ *is irreducible by* H'.
- $g_j\sigma \prec d_j\sigma$ *and* $d_j\sigma$ *is irreducible by* H'.

The next result shows that, when the set of axioms is convergent, a quasi-inconsistent clause cannot be inductively valid. This is proved by building a well-chosen ground instance of the clause, which is false in some Herbrand model of the axioms. In particular, if the clause is an equation then it is not valid in the initial model.

Theorem 4.3 *Let* H *be a convergent set of conditional equations and* $S(H)$ *a test set for* H. *If* C *is quasi-inconsistent w.r.t.* H *then* C *is not an inductive theorem of* H.

Example 4.4 *The axioms are as in example 4.2. (note that they satisfy the convergence property). Consider the conjecture* $even(x) = tt \vee odd(x) = ff$. *It is quasi-inconsistent as shown by the following instance:*

$$even(s(0)) = tt \vee odd(s(0)) = ff$$

The theorems 4.1,4.2 and 4.3 can be combined into an inductive theorem-proving procedure which is complete for positive clauses, in the sense that it will disprove every positive clause which is not an inductive theorem, provided that no negative literals are introduced by the procedure. However, in the general case, the procedure allows to disprove many false conjectures.

5 How to get convergence

Convergent systems of equations have the property that two terms are equal if and only if they simplify to identical ones. In this section, we recall several methods to obtain the convergence property which is crucial for disproving conjectures.

5.1 The saturation technique

The saturation technique generalizes Knuth and Bendix procedure [7] to conditional theories. It is based on a refutationally complete set of inference rules. These rules have been discussed in [13].

5.2 Hierarchical techniques

Hierarchical axiomatizations are natural tools for building structured specifications. They are obtained by incremental extensions of a base theory with new function definitions. For hierarchical axiomatizations [12] , ground confluence can be obtained by semantic methods. The next theorem underlies Plaisted's work [12]:

Theorem 5.1 *Let* H *be a convergent set of conditional equations on the signature* $F - \{f\}$, *and let* H' *be an extension of* H *with conditional equations where the symbol* f *occurs. Assume that* H' *has the same initial model than* H. *If for every ground term* $f(t_1, \ldots, t_n)$ *there exists* $t' \in T(F - \{f\})$ *such that* $f(t_1, \ldots, t_n)\ (\rightarrow_{H'})^*\ t'$ *then* H' *is convergent.*

Verification of inductive properties often involves the proof of some lemmas. Adding these lemmas to the initial axiomatization does not destroy the convergence property as stated in the following result:

Theorem 5.2 *If* H *is convergent and* C *is a conditional equation which is an inductive theorem of* H. *Then* $\rightarrow_{H \cup \{C\}}$ *is convergent.*

For instance, in the introductory example 1-11 and 12,13,14,17,18 is convergent.

6 How to get test-sets

In this section, we propose a method of constructing test-sets for conditional theories whose signature F can be partitionned into a set C of constructors and a set D of defined functions. Therefore, we assume that every left-hand side of an orientable instance of a conclusion has a symbol from D. This corresponds to the well-known principle of definition of [5]. In order to simplify our presentation we shall suppose that $D = \{f\}$.

Definition 6.1 *Let* CS *be the set* $\{g(x_1, \ldots, x_n); g \in C\}$, *a* **pattern tree** T *of* $f(x_1, \ldots, x_n)$, *where* $f \in D$ *is a tree whose nodes are terms. The root is* $f(x_1, \ldots, x_n)$. *Every successor of a node* s *is obtained by replacing a variable of* s *by an element of* S *whose variables have been renamed.*

Example 6.1 Let CS be $\{0, s(x)\}$. Here is a pattern tree of $x < y$:

$$\begin{array}{ccc} & x \leqslant y & \\ & \swarrow \quad \searrow & \\ x < 0 & & x < s(y) \\ & \swarrow \quad \searrow & \\ 0 < s(y) & & s(x) < s(y) \end{array}$$

In the following, we describe a procedure for deriving a pattern-tree such that a test-set can be extracted

from its leaves arguments. Hence, we suppose given a set of conditional equations defining a function f. To construct a suitable pattern tree of $f(x_1, \ldots, x_n)$, the next definition tells us how to identify the nodes to be expanded and the variables to be replaced.

Definition 6.2 *A term t is* **extensible at position** u *w.r.t. H if t/u is a variable and there is a rule $c \Rightarrow l \rightarrow r \in H$ such that l/u is a function symbol or a variable occurring more than once in l. A term t is* extensible *w.r.t. H if it is extensible at some position u.*

Definition 6.3 *Given a set H of conditional equations, we say that t is* **pseudo-reducible** *by H if there is a set of rules $\{c_1 \Rightarrow l_1 \rightarrow r_1, \ldots, c_n \Rightarrow l_n \rightarrow r_n\}$ in S such that $t/u_1 = l_1\sigma_1, \ldots, t/u_n = l_n\sigma_n$ and $C_1\sigma_1 \vee \ldots \vee C_n\sigma_n$ is an inductive theorem of H.*

Note immediately that if a term is pseudo-reducible, all its ground instances are reducible. Also, proving that a node is pseudo-reducible amounts to prove some inductive theorems. To avoid any vicious circle, either we can use a different method to prove these particular properties or we can use our method itself with a less refined test-set than the one we are trying to build. Let H be a set of conditional equations and let T be a pattern tree for $f(x_1, \ldots, x_n)$. If each leaf of T is either pseudo-reducible or not-extensible then we say T is complete for H. The tree in example 6.1 is complete for the axioms of the preliminary example. The previous definitions provides us with a procedure to derive complete pattern-trees. Starting from the tree $T = f(x_1, ..., x_n)$, we iterate the following operations:

- Select non-deterministically a leaf t which is extensible at some position u and not pseudo-reducible
- For any c in CS, rename c in c' with new variables and add $t[c']_u$ as a son of t.

When the previous procedure halts with success (as it did on all the examples we have tested), it provides immediately a test-set:

Theorem 6.1 *Given a set of conditional equations H, if there is a finite complete pattern-tree for $f(x_1, \ldots, x_n)$ whose all leaves are pseudo-reducible then a test-set for H can be computed.*

Given a finite complete pattern-tree, let G' be the set of its leaves arguments. We define G to be a subset of G' such that every element of G' has an instance in G and no element of G is an instance of another element of G. In the example 6.1, $G' = \{0, s(x), s(y)\}$ and $G = \{0, s(x)\}$.

7 Conclusion

We have presented new methods for inductive reasoning. These methods try to capture as much as possible the power of simplification. Moreover, when the axioms are convergent, test-sets give an efficient strategy to disprove theorems. We also feel that this method should generalize easily to the case of conditional equations with negative literals in the conditions.

References

[1] R. Aubin. Mechanizing structural induction. In *Theoretical Computer Science 9*, pp. 329–362, 1979.

[2] R.S. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, New York, 1979.

[3] R-M. Burstall. Proving properties of programs by structural induction. In *Computer Journal 12*, pp. 41–48, 1969.

[4] S.J. Garland and J.V. Guttag. An overview of LP, the Larch Prover. In N. Dershowitz, ed., *Proc. of the 3rd RTA Conf., USA*, pp. 137–151, LNCS 355, 1989.

[5] G. Huet and J-M. Hullot. Proofs by induction in equational theories with constructors. *J. of Computer and System Sciences*, 25(2):239–266, 1982.

[6] J.P. Jouannaud and E. Kounalis. Proof by induction in equational theories without constructors. In *Proc. of 1st Symp. on Logic In Computer Science*, pp. 358–366, Boston (USA), 1986.

[7] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In J. Leech, ed., *Computational Problems in Abstract Algebra*, pp. 263–297, Pergamon Press, Oxford, 1970.

[8] E. Kounalis. Pumping lemmas in tree languages. in *Mathematical Foundations of Computer Science*, 1990.

[9] E. Kounalis and M. Rusinowitch. A mechanization of conditional reasoning. In *First International Symp. on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, 1990.

[10] D.L. Musser. On proving inductive properties of abstract data types. In *Proc. 7th ACM POPL*, pp. 154–162, 1980.

[11] P. Padawitz. *Computing in Horn Clause Theories.* Springer-Verlag, 1988.

[12] D. Plaisted. Semantic confluence tests and completion methods. In *Journal Information and Control 65*, pp. 182–215, 1985.

[13] M. Rusinowitch. Theorem-proving with resolution and superposition. In *Proc. of the International Conference on Fifth Generation Computer Systems*, 1988.

[14] H. Zhang, D. Kapur, and M.S. Krishnamoorthy. A mechanizable induction principle for equational specifications. In E. Lusk and R. Overbeek, ed., *Proc. 9th CADE*, pp. 162–181, LNCS 310, 1988.