ICOT Technical Report: TR-0754

TR-0754

Inductive Theorem Proving based on Term Rewriting

by
A. Ohsuga (Toshiba)

March, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
(03)3456-3191～5
Telex ICOT J32964
Minato-ku Tokyo 108 Japan

Institute for New Generation Computer Technology

Inductive Theorem Proving based on Term Rewriting

Akihiko Ohsuga
Systems & Software Engineering Laboratory
Toshiba Corporation
70, Yanagi-cho, Saiwai-ku
Kawasaki, Kanagawa 210, Japan
ohsuga@ssel.toshiba.co.jp

Abstract

An equation which is valid in the initial model of an equational theory is called an inductive theorem. In this paper, we present an inductive theorem proving method that is an extension of the so called inductive completion procedures. Compared with these procedures, our method can handle unorientable axioms and theorems and, therefore, does not fail. The method has been implemented on *Metis*, an experimental system for term rewriting techniques.

1 Introduction

Musser and Goguen showed that the correctness of an inductive theorem T in an equational theory E is equivalent to the consistency of an extended axiom system E' obtained by adding T to an axiom set of E as a new axiom[5, 10]. Moreover, they suggested that the Knuth-Bendix completion procedure[9] can be used to prove consistency of the new system E'. Their method has only one kind of inconsistency, i.e. true=false. Huet and Hullot proposed a method that finds inconsistencies more easily by decomposing function symbols into constructors and defined symbols[8]. Dershowitz pointed out that an equation T is valid in the initial model defined by R if and only if no equality between two distinct irreducible ground terms follows from R and T[3]. Fribourg presented a linear proof method which restricts the numbe of critical pairs to be generated[4]. These proof methods are called inductive completion procedures.

In spite of the fact that the inductive completion procedure works efficiently, it sometimes fails if an equation that cannot be oriented in the given ordering is generated. Furthermore, it sometimes loops, generating an infinite number of critical pairs. In this paper, we address both problems by presenting a new inductive completion procedure which is based on Fribourg's linear method[4]. As for the first problem, we employ orientation-free rewrite rules. With these rewrite rules, the method can obtain a ground convergent rewriting system which corresponds to the given equational theory whenever it terminates. Since inductive completion needs confluence only on the set of ground terms, the extended method is still refutationally complete. It is well known that the termination problem of Knuth-Bendix procedure is undecidable. Then, for the second problem, we introduce a criterion, called the recursive pair, to detect cases where the procedure fails to terminate.

This criterion is based on Hermann's crossed pair which examines the structure of critical pairs to find infinite loops in the Knuth-Bendix completion[6].

The method is refutationally complete; that is, it refutes any equation which is not an inductive theorem. We have implemented it on *Metis*, an experimental system for term rewriting techniques. Several experiments with *Metis* illustrate how the method works.

2 Preliminaries

In this section, we introduce the terminology and notation used in this paper. We assume that the reader is familiar with the concepts of term rewriting systems (see[7]).

We deal with finite sequences of the following two kinds of symbols (and parentheses and commas for ease of reading); a finite set $\mathcal{F}$ of *function symbols*, and a denumerable set $\mathcal{V}$ of *variables*. We will denote the set of all terms constructed from $\mathcal{F}$ and $\mathcal{V}$ by $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and the set of all the ground terms constructed from $\mathcal{F}$ by $\mathcal{T}(\mathcal{F})$. A term that is constructed from $\mathcal{F}$ is called a ground term. We assume that $\mathcal{F}$ contains at least one constant, therefore the set of ground terms is not empty.

The notation t/p represents a subterm of t at position p. $t[s]$ represents a term t containing s as its subterm. In this context, $[s]$ represents a certain occurrence of s in $t[s]$. Thus, $t[s']$ denotes the term obtained by replacing the occurrence of s in $t[s]$ with s'.

$t\theta$ denotes the result of applying a substitution θ to a term t, and is called an *instance* of t. $t\theta$ is called a *ground instance* of t if $t\theta$ is ground.

Let $\sim$ be a binary relation on terms. We say that $\sim$ is monotonic with respect to the term structure if $t \sim u$ implies $c[t] \sim c[u]$, for all terms t, u, and contexts $c[\,]$. $\sim$ is monotonic with respect to instantiation if $t \sim u$ implies $t\theta \sim u\theta$, for all terms t, u, and substitutions θ. A relation that satisfies both properties is called *monotonic*. An *ordering* is an irreflexive and transitive binary relation. An ordering $\rightsquigarrow$ is *well-founded* if there is no infinite sequence $t_1 \rightsquigarrow t_2 \rightsquigarrow \cdots$. Ordering $\rightsquigarrow$ on terms is called *reduction ordering* if it is well-founded and monotonic. *Simplification ordering* is reduction ordering that satisfies the subterm property: $c[t] \rightsquigarrow t$ for all contexts $c[\,]$ and terms t. A *strong simplification ordering* is a simplification ordering that is total on ground terms.

An *equation* is a pair of terms $l = r$. Given a set of equations E, the symmetric and monotonic closure of E is denoted by $\leftrightarrow_E$. That is, $t \leftrightarrow_E u$ if and only if t is $c[l\theta]$ and u is $c[r\theta]$ for some context $c[\,]$, substitution θ, and equation $l \doteq r$ in E ($l \doteq r$ denotes $l = r$ or $r = l$). The reflexive transitive closure of $\leftrightarrow_E$ is denoted by $\overset{*}{\leftrightarrow}_E$. Note that $\overset{*}{\leftrightarrow}_E$ is a congruence relation on terms.

Let E be a set of equations. $\mathcal{T}(\mathcal{F})/\overset{*}{\leftrightarrow}_E$ is called an *initial model* of E. An equation $g = d$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is called an *inductive theorem* of E if $g = d$ holds in the initial model of E. An equation $g = d$ is an inductive theorem of E if and only if $g\theta \overset{*}{\leftrightarrow}_E d\theta$ for all ground instances $g\theta = d\theta$ of $g = d$.

3 Orientation-free rewrite rules

First, we introduce orientation-free rewrite rules and extend the concept of reduction. Usually, a rewrite rule is an oriented pair of terms. In this paper, however,

we define a *rewrite rule* (or an *orientation-free rewrite rule*) as an unoriented pair of terms, written $l \Leftrightarrow r$. A set of such rewrite rules is called an *unoriented term rewriting system* (UTRS). Let R be a UTRS, and $\succ$ be an ordering on terms. A term t is said to be *reduced* to another term u by R and $\succ$, written $t \rightarrow_{R\succ} u$, if there exists a rewrite rule $l \Leftrightarrow r$ and a substitution θ such that t is $c[l\theta]$, u is $c[r\theta]$, and $t \succ u$ ($l \Leftrightarrow r$ denotes $l \Leftrightarrow r$ or $r \Leftrightarrow l$). The reflexive and transitive closure of $\rightarrow_{R\succ}$ is denoted by $\stackrel{*}{\rightarrow}_{R\succ}$ and the symmetric closure of $\stackrel{*}{\rightarrow}_{R\succ}$ is denoted by $\stackrel{*}{\leftrightarrow}_{R\succ}$. It is a routine to verify that $\stackrel{*}{\leftrightarrow}_{R\succ}$ is a congruence relation.

We say that UTRS is *terminating* with respect to $\succ$ if $\succ$ is well-founded. We say that UTRS R is *ground confluent* with respect to $\succ$ if, for all ground terms t, u_1, u_2 such that $u_1 \stackrel{*}{\leftarrow}_{R\succ} t \stackrel{*}{\rightarrow}_{R\succ} u_2$, there exists a ground term v such that $u_1 \stackrel{*}{\rightarrow}_{R\succ} v \stackrel{*}{\leftarrow}_{R\succ} u_2$. We say that a term t is *reducible* with respect to R and $\succ$ if there exists a term u such that $t \rightarrow_{R\succ} u$. A term t is said to be *irreducible* if t is not reducible. An irreducible term s such that $t \stackrel{*}{\rightarrow}_{R\succ} s$ is called an *irreducible form* of t with respect to R and $\succ$, written $t{\downarrow}$. A UTRS which is terminating and ground confluent is said to be *ground convergent*. If R is a ground convergent UTRS, then every ground term t has a unique irreducible form $t{\downarrow}$, called the *normal form* of t. From now on, we restrict $\succ$ to strong simplification ordering. Thus, any UTRS is always terminating.

In the following discussions, we assume that all variables appearing in given terms, equations, and rewrite rules are universally quantified. Thus, they do not have common variables. For notational simplicity, $\succ$ may be omitted if it is clear from the context.

4 Inference rules for ground completion

Our method mainly consists of two procedures. One is a ground completion to obtain a ground convergent UTRS R which corresponds to a given equational theory E. The other is a method of inductive theorem proving to check the consistency of T which is a set of equations we want to prove with respect to R. First, we discuss a ground completion procedure.

Together with the extension of reductions by UTRS, let us extend the definition of critical pairs[9] as well. Let $l_1 \Leftrightarrow r_1$ and $l_2 \Leftrightarrow r_2$ be rewrite rules and s be a non-variable subterm of l_2 at position p unifiable with l_1. If $l_1\theta \not\leftarrow r_1\theta$ and $l_2\theta \not\leftarrow r_2\theta$, then a pair $c[r_1]\theta = r_2\theta$ is called a *critical pair* of $l_1 \Leftrightarrow r_1$ on $l_2 \Leftrightarrow r_2$ at position p (with respect to $\succ$), where l_2 is $c[s]$ and θ is the most general unifier of l_1 and s.

A ground completion procedure consists of the following inference rules, where E is a set of equations and R is a UTRS.

E-generation: $$\frac{(E, R)}{(E \cup \{t = u\}, R))} \quad \text{if } t = u \text{ is a critical pair of } R$$

E-simplification: $$\frac{(E \cup \{t \doteq u\}, E)}{(E \cup \{t' = u\}, R)} \quad \text{if } t \rightarrow_R t'$$

E-deletion: $$\frac{(E \cup \{t = t\}, R)}{(E, R)}$$

R-generation: $$\frac{(E \cup \{t = u\}, R)}{(E, R \cup \{t \Leftrightarrow u\})}$$

R-simplification: $$\frac{(E, R \cup \{l \stackrel{\cdot}{\Leftrightarrow} r\})}{(E, R \cup \{l \Leftrightarrow r'\})} \quad \text{if } l \succ r \text{ and } r \rightarrow_R r'$$

R-deletion: $$\frac{(E, R \cup \{l \stackrel{\cdot}{\Leftrightarrow} r\})}{(E \cup \{l' = r\}, R)}$$

if $l \not\succ r$, $l \rightarrow_R l'$ by a rule $g \Leftrightarrow d$ and a substitution θ, $l \rhd g$, and $g\theta \succ d\theta$

Where $\rhd$ is a specialization ordering, i.e. $l \rhd g$ if and only if some subterm of l is an instance of g, but not vice versa.

At the beginning of the procedure, E is a set of equations that corresponds to a set of axioms for a given equational theory and R is an empty set. When one of the inference rules is applied, a tuple (E, R) is transformed to another tuple (E', R'), denoted by $(E, R) \vdash (E', R')$. Let

$$(E_0, R_0) \vdash (E_1, R_1) \vdash (E_2, R_2) \vdash \cdots$$

be a sequence of applications of the inference rules. We denote $\bigcup_{i=0}^{\infty} \bigcap_{j=i}^{\infty} E_j$ by E_∞ and $\bigcup_{i=0}^{\infty} \bigcap_{j=i}^{\infty} R_j$ by R_∞. An inference sequence is called *fair*, if it satisfies the following conditions.

(1) Any critical pair of R_∞ is contained in $\bigcup_{i=0}^{\infty} E_i$.

(2) E_∞ is empty.

We claim that any fair inference sequence can generate a ground convergent UTRS as R_∞.

Theorem 4.1 Let E_0 be a set of equations, R_0 be an empty set, $\succ$ be a strong simplification ordering, and $(E_0, R_0) \vdash (E_1, R_1) \vdash \cdots$ be a fair inference sequence. Then, R_∞ is ground convergent with respect to $\succ$ where the congruence relations $\stackrel{\cdot}{\leftrightarrow}_{E_0}$ and $\stackrel{\cdot}{\leftrightarrow}_{R_\infty}$ are the same.

5 Inference rules for inductive theorem proving

A term t is said to be *inductively reducible* (with respect to R and $\succ$) if and only if for every ground instance $t\theta$, $t\theta$ is reducible (with respect to R and $\succ$). A rewrite $l \Leftrightarrow r$ is said to be inductively reducible if and only if $l \not\succ r$ and l is inductively reducible. A set of rewrite rules S is said to be *provably inconsistent* (with respect to R and $\succ$)if it contains a rewrite rule that is not inductively reducible (with respect to R and $\succ$). A set S is inconsistent if S is provably inconsistent.

A set of equations T is called a *covering set* for a set of rewrite rules S with respect to $\succ$ if and only if, for all ground instances $l\theta = r\theta$ of all inductively reducible rewrite rules $l = r$ in S, there exists a ground instance $g\theta' = d\theta'$ of an equation $g = d$ in T such that $l\theta = r\theta \sqsupset g\theta' = d\theta'$.

A subterm position p in term t is said to be *complete* (with respect to R and $\succ$) if t/p is not a variable and, for all ground instances $(t/p)\theta$, $(t/p)\theta$ is reducible (by R and $\succ$) either at the variable position of t/p or at the position p. A critical pair

of $l_1 \Leftrightarrow r_1$ on $l_2 \Leftrightarrow r_2$ at position p (with respect to R and $\succ$) is said to be complete if p is complete (with respect to R and $\succ$).

Let R be a ground convergent rewrite system with respect to $\succ$. An inductive theorem proving procedure consists of the following inference rules, where T is a set of equations (called *conjectures*) and S is a set of rewrite rules.

T-generation: $$\frac{(T, S)}{(T \cup \{g = d\}, S)}$$ if $g = d$ is a complete critical pair of R on S

T-simplification: $$\frac{(T \cup \{g = d\}, S)}{(T \cup \{g' = d\}, S)}$$ if $g \rightarrow_{R \cup S} g'$

T-deletion: $$\frac{(T \cup \{g = g\}, S)}{(T, S)}$$

S-generation: $$\frac{(T \cup \{g = d\}, S)}{(T, S \cup \{g \Leftrightarrow d\})}$$

At the beginning of the procedure, T is a set of equations that corresponds to a set of theorems to be proved, and S is an empty set. When one of the inference rules is applied, a tuple (T, S) is transformed to another tuple (T', S'), denoted by $(T, S) \vdash (T', S')$. Let

$$(T_0, S_0) \vdash (T_1, S_1) \vdash (T_2, S_2) \vdash \cdots$$

be a sequence of applications of the inference rules. We denote $\bigcup_{i=0}^{\infty} \bigcap_{j=i}^{\infty} T_j$ by T_∞ and $\bigcup_{i=0}^{\infty} S_i$ by S_∞. An inference sequence is called *fair*, if it satisfies the following conditions.

(1) Any complete critical pair of R on S_∞ is contained in $\bigcup_{i=0}^{\infty} T_i$. In other words, $\bigcup_{i=0}^{\infty} T_i$ is a cover set for S_∞.

(2) T_∞ is empty.

We claim that any fair inference sequence transforms inconsistent sets of rewrite rules to provably inconsistent sets.

Theorem 5.1 Let T_0 be a set of equations, S_0 be an empty set, $\succ$ be a strong simplification ordering, and $(T_0, S_0) \vdash (T_1, S_1) \vdash \cdots$ be a fair inference sequence. Then, S_∞ is provably inconsistent with respect to $\succ$ if and only if T_0 is inconsistent with respect to R and $\succ$.

6 Detecting infinite execution

Our procedure, which is the same as the other inductive completions, sometimes loops to generate an infinite number of critical pairs. To solve this problem, we introduce a criterion, called *recursive pair*, to detect some of the cases where procedures fail to terminate. The criterion is based on Hermann's crossed pair which

examines the structure of critical pairs to find infinite loops in the Knuth-Bendix completion[6].

Let $l_1 \Leftrightarrow r_1$ and $l_2 \Leftrightarrow r_2$ be rewrite rules and s be a non-variable subterm of l_2 unifiable with l_1. If there exists a non-variable subterm t of r_1 such that $t\theta$ is unifiable with l_1', $c[d[t]]\theta \not\prec r_2\theta$, $l_1\theta \not\prec r_1\theta$, and $l_2\theta \not\prec r_2\theta$, then a pair $c[d[t]]\theta = r_2\theta$ is called a *recursive pair* of $l_1 \Leftrightarrow r_1$ on $l_2 \Leftrightarrow r_2$ (with respect to $\succ$). Where r_1 is $d[t]$, l_2 is $c[s]$, θ is the most general unifier of l_1 and s, and l' is a variant of l, i.e. all variables are consistently renamed.

Obviously, if a pair is recursive then it is critical. When a recursive pair exists and there is no rewrite rule which reduces the recursive pair, an infinite number of critical pairs may be generated in the following form.

$$\begin{aligned} c[d[t]]\theta_0 &= r_2\theta_0 \\ c[d[d[t]]]\theta_0\theta_1 &= r_2\theta_0\theta_1 \\ &\vdots \\ c[d\cdots d[t]\cdots]\theta_0\cdots\theta_n &= r_2\theta_0\cdots\theta_n \end{aligned}$$

Where θ_n is the most general unifier of $t\theta_{n-1}$ and l_n.

7 Implementation

In our procedure, which is similar to the other procedures, the notion of inductive reducibility plays an important role. It is known that the inductive reducibility is decidable for finite rewriting systems[11], but often takes exponential time. In *Metis*, we adopt a notion of (free) constructors since the inductive reducibility can be checked quite easily using theories with constructors. From now on, we assume that function symbols are decomposed into two sets: a set of constructors, denoted by $\mathcal{C}$, and a set of defined symbols (or non-constructors), denoted by $\mathcal{D}$. We also assume that $\mathcal{C}$ contains at least two symbols. Then, a term t is inductively reducible if and only if t contains a defined symbol[7].

In the following, we show an implementation of inference rules for inductive theorem proving with constructors. Some of the cases where the procedure does not terminate are detected by the detector of recursive pairs. A way to avoid infinite loops caused by such recursive pairs is to introduce new lemmas which can reduce the general form of these recursive pairs. So far, on *Metis*, this is done by users.

procedure *inductive_theorem_proving*(T, R, S)
  while $T \neq \phi$ **do**
    select $g = d$ in T
    if $g \equiv d$ **or** $g = d$ is subsumed by $R \cup S$ **then**
      $T := T - \{g = d\}$
    elseif $g \equiv c(g_1, \ldots, g_n), d \equiv c(d_1, \ldots, d_n), c \in C$ **then**
      $T := (T - \{g = d\}) \cup \{g_i = d_i \mid 1 < i < n\}$
    else
      if $g \equiv c_1(g_1, \ldots, g_m), d \equiv c_2(d_1, \ldots, d_n), c_1, c_2 \in C, c_1 \neq c2$ **or**
        $g \equiv c(g_1, \ldots, g_n), c \in C, d \in V$ **or**
        $g \in V, d \equiv c(d_1, \ldots, d_n), c \in C$ **or**
        $g, d \in V$ **or**
        $g \succ d$ and g is in $\mathcal{T}(\mathcal{C}, \mathcal{V})$ **or**

$g \prec d$ and d is in $\mathcal{T}(\mathcal{C},\mathcal{V})$ **or**
$g \not\succ d, g \not\prec d$ and $g = d$ is in $\mathcal{T}(\mathcal{C},\mathcal{V})$ **then**
stop with answer "disproved"
end_if
$T := (T - \{g = d\}) \cup \{\text{complete critical pairs of } R \text{ on } g \Leftrightarrow d\}$
$S := S \cup \{g \Leftrightarrow d\}$
if there are recursive pairs between R and S **then**
$T := T \cup \{\text{lemma by user}\}$
end_if
reduce T by $R \cup S$
end_if
end_while
stop with answer "proved"
end_procedure

8 Experiments

8.1 Addition and multiplication of natural numbers

When we express natural numbers with successor function s and 0, addition $+$ is defined by equations as $\{X + 0 = X,\ X + s(Y) = s(X + Y)\}$. We show that this operation satisfies two conjectures: associativity and commutativity, that is, $\forall X, Y, Z.\ (X + Y) + Z = X + (Y + Z)$ and $\forall X, Y.\ X + Y = Y + X$. Usually, such properties are proved by rewriting modulo congruence obtained by extending unifications. Our method, however, can prove these without any extension. Let $\{+\}$ be a set of defined symbols, $\{s, 0\}$ be a set of constructors.

```
[METIS]-> pr i
<< inductive theorem proving >>
Theorem> A+B=B+A.
Theorem> (A+B)+C=A+(B+C).
```

We start the proving procedure with these conjectures. Since the equation is unorientable, an orientation-free rule r3 is obtained.

```
New r3: A+B <-> B+A (e3)
  divergent critical pairs...
    e5: A = 0+A  (r1/r3)
    e6: s(A+B) = s(B)+A  (r2/r3)
       %%% "s(..s(A+B)..)" = s("..s(B)..")+A is a recursive pair %%%
New r4: 0+A -> A (e5)
New r5: A+B+C -> A+(B+C) (e4)
```

If we have an order which orients e6 right to left, then the procedure successfully terminates. Unfortunately, we do not have such an order. (Actually, recursive path ordering with multiset status can do this, but this is not strong simplification ordering.) Then, an orientation-free rewrite rule r6 is obtained.

```
New r6: s(A+B) <-> s(B)+A (e6)
  divergent critical pairs...
    e7: s(A) = 1+A  (r1/r6)
    e8: s(s(A+B)) = s(s(B))+A  (r2/r6)
       %%% s("s(..s(A+B)..)") = s(s("..s(B).."))+A is a recursive pair %%%
```

Metis finds equations e6 and e8 as a series of recursive pairs r2 on r3 (for e6), r2 on r6 (for e8). Thus, it is suggested that new lemmas which reduce the general form of these recursive pairs are necessary. We introduce a new equation $s(X)+Y=s(X+Y)$ as such a lemma.

```
<< introduce a new lemma >>
Lemma> s(A)+B=s(A+B).
New r7: s(A)+B -> s(A+B) (e9)
  reduced equations...
    e7: s(A) = 1+A  (r1/r6[r7,r4])
       => s(A) = s(A) [trivial]
    e8: s(s(A+B)) = s(s(B))+A  (r2/r6[r7,r7])
       => s(s(A+B)) = s(s(B+A)) [subsumed]
    ####### PROVED #######
CP(s)         : 10 found, 4 asserted.
Rule(s)       : 7 generated, 7 remain.
Reduction     : 12 steps.
Runtime       : 672 msec
              ( 13% for selection, 8% for ordering, 18% for reduction of
                equations, 45% for superposition, 16% for others )
```

Similarly, the associativity and the commutativity of multiplication can be proved. We add axioms for multiplication: $\{A*0=0,\ A*s(B)=A+A*B\}$ to the previous axioms and prove the conjectures $\forall X,Y.X*Y=Y*X$, $\forall X,Y,Z.(X*Y)*Z=X*(Y*Z)$ with several auxiliary equations.

```
[METIS]-> pr i
<< inductive theorem proving >>
Theorem> A+B=B+A.
Theorem> (A+B)+C=A+(B+C).
Theorem> A*B=B*A.
Theorem> (A*B)*C=A*(B*C).
Theorem> s(A)+B=s(A+B).
Theorem> A*(B+C)=A*B+A*C.
Theorem> A+B*A=s(B)*A.
Theorem> A+(B+C)=B+(A+C).
New r5: A+B <-> B+A
( "s(..s(A+B)..)" = s("..s(B)..")+A : recursive pair )
New r6: A*B <-> B*A
( "A+..A*B.." = s("..s(B)..")*A : recursive pair )
New r7: 0+A -> A
New r8: 0*A -> 0
New r9: A+B+C -> A+(B+C)
New r10: A*B*C -> A*(B*C)
New r11: A+(B+C) <-> B+(A+C)
New r12: s(A)+B -> s(A+B)
New r13: s(A)*B -> B+A*B
New r14: A*(B+C) -> A*B+A*C
    ####### PROVED #######
CP(s)         : 20 found, 4 asserted.
Rule(s)       : 10 generated, 10 remain.
Reduction     : 32 steps.
Runtime       : 1.652 sec
              ( 6% for selection, 8% for ordering, 20% for reduction of
                equations, 44% for superposition, 22% for others )
```

8.2 Append and two reverse operations

Properties of programs defined by rewrite rules can be verified using inductive theorem proving. In the next example, the procedure detects inconsistencies caused by incorrect axioms. We give the system a set of axioms $\{app([\,], A) = A$, $app([A|B], C) = [A|app(B, C)]$, $rev([\,]) = [\,]$. $rev([A|B]) = app(rev(B), A)\}$. However, this is not correct since the last axiom should be $rev([A|B]) = app(rev(B), [A])$. Through theorem proving, the existence of such errors is known. As a conjecture, we try to prove an equation: $\forall X.\ rev(rev(X)) = X$.

```
[METIS]-> pr i
<< inductive theorem proving >>
Theorem> rev(rev(A))=A.
New r5: rev(rev(A)) -> A
( rev("app(..app(rev(A),B)..,C)") = [C|"..[B|A].."] : recursive pair )
New r6: rev(app(rev(A),B)) -> [B|A]
( rev(app("app(..app(rev(A),B)..,C)",D)) = [D,C|"..[B|A].."] : recursive pair )
New r7: rev(A) -> [A]
    ####### DISPROVED #######
    By e5: [] = [[]] (r3/r7)
    ####### CHECK THE FOLLOWING #######
Axiom(s):
  r1: app([],A) -> A (axiom)
  r3: rev([]) -> [] (axiom)
  r4: rev([A|B]) -> app(rev(B),A) (axiom)
CP(s)         : 6 found, 5 asserted.
Rule(s)       : 3 generated, 3 remain.
Reduction     : 13 steps.
Runtime       : 502 msec
              ( 3% for selection, 0% for ordering, 17% for reduction of
                equations, 37% for superposition, 43% for others )
```

Metis tells us that inconsistency $[\,] = [[\,]]$ comes from axioms: r1, r3, and r4. Then, the user can identify the error of r4 efficiently. Next, we prove the equivalence of two different reverse operations. Append and two reverse operations are defined by a $\{app([\,], A) \Leftrightarrow A$, $app([A|B], C) \Leftrightarrow [A|app(B, C)]$, $rev([\,]) \Leftrightarrow [\,]$, $rev([A|B]) \Leftrightarrow app(rev(B), [A])$, $new([\,], A) \Leftrightarrow A$, $new([A|B], C) \Leftrightarrow new(B, [A|C])\}$. We start the procedure with a conjecture $\forall X.\ new(X, [\,]) = rev(X)$, which shows that the two reverses are equivalent. Let $\{rev, app, new\}$ be a set of defined symbols and $\{[\,], [\_|\_]\}$ be a set of constructors.

```
[METIS]-> pr i
<< inductive theorem proving >>
Theorem> rev(A)=new(A,[]).
You want to orient
  [1] rev(A) -> new(A,[])
  [2] rev(A) <- new(A,[])
  else exit
Which ? 1
[ new << rev is asserted. ]
New r7: rev(A) -> new(A,[]) (e1)
  divergent critical pairs...
    e2: app(rev(B),[A]) = new([A|B],[])  (r4/r7[r7,r6])
       => app(new(B,[]),[A]) = new(B,[A])
You want to orient
  [1] app(new(A,[]),[B]) -> new(A,[B])
  [2] app(new(A,[]),[B]) <- new(A,[B])
```

```
  else exit
Which ? 1
[ new << app is asserted. ]
New r8: app(new(A,[]),[B]) -> new(A,[B]) (e3)
  divergent critical pairs...
    e3: app(new(B,[A]),[C]) = new([A|B],[C])  (r6/r8[r6])
       => app(new(B,[A]),[C]) = new(B,[A,C])
       %%% app("new(A,[B,C])",[D]) = new([C|"..[B|A].."],[D]) is a recursive pair %%%
New r9: app(new(A,[B]),[C]) -> new(A,[B,C]) (e3)
  divergent critical pairs...
    e4: app(new(B,[A,C]),[D]) = new([A|B],[C,D])  (r6/r9[r6])
       => app(new(B,[A,C]),[D]) = new(B,[A,C,D])
       %%% app("new(A,[B,C,D])",[E]) = new([C|"..[B|A].."],[D,E]) is a recursive pair %%%
```

Metis notifies that e3 and e4 are series of recursive pairs r6 on r8 (for e3), r6 on r9 (for e4), and there is no rewrite rule to reduce them. Thus, there is a possibility of infinite execution. The generalized form of critical pairs derived from these recursive pairs is

$$app(new(X,[Y_1,Y_2,\ldots,Y_n]),[Z]) = new(X,[Y_n,\ldots,Y_2,Y_1|X],[Z])$$

and is reduced to

$$app(new(X,[Y_1,Y_2,\ldots,Y_n]),[Z]) = new(X,[Y_1,Y_2,\ldots,Y_n,Z]).$$

We can easily find new lemma which reduces the above formula

$$app(new(X,Y),[Z]) = new(X,app(Y,[Z]))$$

then we add it.

```
<< introduce a new lemma >>
Lemma> app(new(A,B),[C])=new(A,app(B,[C])).
New r10: app(new(A,B),[C]) -> new(A,app(B,[C])) (e5)
  reduced equations...
    e4: app(new(A,[B,C]),[D]) = new(A,[B,C,D])  (r6/r9[r10,r2,r2,r1,r6])
       => new(A,[B,C,D]) = new(A,[B,C,D]) [trivial]
    ####### PROVED #######
CP(s)         : 8 found, 3 asserted.
Rule(s)       : 4 generated, 4 remain.
Reduction     : 18 steps.
Runtime       : 1.338 sec
              ( 9% for selection, 52% for ordering, 6% for reduction of
                equations, 22% for superposition, 11% for others )
```

Acknowledgments

This research is partially supported by MITI through the research and development of the Fifth Generation Computer System (FGCS) project in Japan. The authors would like to thank to Dr. K.Fuchi (ICOT Director), Dr. R.Hasegawa (Chief of ICOT 4th and 5th Laboratories), Mr. S.Nishijima (Chief of Toshiba Systems and Software Engineering Laboratory), Mr. Y.Ofude (Senior Manager of Toshiba Systems and Software Engineering Laboratory), and Dr. S.Honiden (Senior research scientist of Toshiba Systems and Software Engineering Laboratory) for the opportunity of conducting this research.

References

[1] Bachmair, L., Dershowitz, N., and Hsiang, J.: Orderings for equational proofs, in *Proc. 1st IEEE Symposium on Logic in Computer Science*, (1986), pp. 346–357.

[2] Bachmair, L.: Proof by Consistency, in *Proc. 3rd IEEE Symposium on Logic in Computer Science* (1988), pp. 228–233.

[3] Dershowitz, N.: Applications of the Knuth-Bendix completion procedure, in *Seminaire d'Informatique Theorique* (1982), pp. 95–111.

[4] Fribourg, L.: A strong restriction of the inductive completion procedure, in *Proc. 13th Int. Colloquium Automata, Languages and Programming*, Lecture Notes in Computer Science 226, Springer-Verlag (1986), pp. 105–115.

[5] Goguen, J. A.: How to prove algebraic induction hypothesis without induction, with application to the correctness of data type implementation, in *Proc. 5th Int. Conf. on Automated Deduction*, Lecture Notes in Computer Science 87, Springer-Verlag (1980), pp. 356–373.

[6] Hermann, M. and Privara, I.: On nontermination of Knuth-Bendix algorithm, Research Report CS-812-21, Institute of Socio Economic Information and Automation (1985), VUSEI-AR-OPS-3/85.

[7] Huet, G. and Oppen, D. C.: Equations and Rewrite Rules: A Survey, in *Formal Languages: Perspective and Open Problems* (Book, R. ed.), Academic Press (1980), pp. 349–405.

[8] Huet, G. and Hullot, J.-M.: Proofs by induction in equational theories with constructors, *J. Comput. Syst. Sci.*, Vol. 25, No. 2 (1982), pp. 239–266.

[9] Knuth, D. E. and Bendix, P. B.: Simple word problems in universal algebras, in *Proc. Computational problems in abstract algebra* (Leech, J. ed.), Pergamon Press, Oxford (1970), pp. 263–297; also in *Automation of Reasoning 2* (Siekmann, J. H. and Wrightson eds.), Springer Verlag (1983), pp. 342–376.

[10] Musser, D. R.: On proving inductive properties of abstract data types, in *Proc. 7th ACM Symposium on Principles of Programming Languages* (1980), pp. 154–162.

[11] Plaisted, D. A.: Semantic confluence tests and completion methods, *Inf. Control*, Vol. 65 (1985), pp. 182–215.

[12] Reddy, U. S.: Term Rewriting Induction, in *Proc. 10th Int. Conf. on Automated Deduction*, Lecture Notes in Computer Science 449, Springer Verlag (1990), pp. 162–177.