ICOT Technical Report: TR-0753

TR-0753

Metis: A term Rewriting System Generator

by
A. Ohsuga (Toshiba) & K. Sakai

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

Metis: A Term Rewriting System Generator

— An Inference Engine for Equations and Inequations —

Akihiko Ohsuga
Systems & Software Engineering Laboratory
Toshiba Corporation
70 Yanagi-cho, Saiwai-ku
Kawasaki, Kanagawa 210, Japan

Kô Sakai
ICOT Research Center
1-4-28 Mita, Minato-ku
Tokyo 108, Japan

Abstract

The TRS (term rewriting system) Working Group of ICOT[1] has been studying applications of TRSs to the intelligent programming system. As a result, we have implemented a TRS generator called *Metis*, an experimental tool with the many functions required for such a system. This paper describes the features of *Metis* and several experiments with it.

1 Introduction

A set of rewrite rules is called a term rewriting system or TRS. The theory of TRSs has a wide variety of both theoretical and practical applications. It provides models for abstract data types, operational semantics for functional programming languages, and inference engines for automated theorem proving with equality.

The intelligent programming system is an important research topic of Japan's Fifth Generation Computer System (FGCS) Project. A lot of evidence suggests that the study of TRSs will yield key technologies for the intelligent programming system, in particular for specification, verification, and synthesis of programs. The Institute for New Generation Computer Technology (ICOT) organized the TRS Working Group in 1985 to study the theory of TRSs and their application to the intelligent programming system.

Metis is the first result of the activity of the working group. It generates a complete TRS from a set of equations automatically, semi-automatically, or interactively. It is also an experimental tool with the various functions needed for the study of TRSs.

The kernel function of *Metis* is the Knuth-Bendix completion procedure, significantly improved with better capabilities and operability by the incorporation of many new facilities. For example, *Metis* can provide us with several kinds of ordering methods of terms, but the user can orient an equation with little knowledge of the ordering methods and obtain an appropriate rewrite rule that does not violate termination of

[1] ICOT stands for Institute for New Generation Computer Technology

the TRS. If the equation cannot be oriented to either direction, *Metis* offers the user several kinds of recipe. It manipulates inequations as well as equations and provides special handling of associative-commutative operators in the completion procedure.

Section 2 describes the basic concept of the TRS. Section 3 introduces the features of *Metis* in the general framework, and in Section 4, several concrete examples illustrate how *Metis* actually works.

2 Preliminaries

In this section, we will introduce the terminology and notation in this paper and survey well-known properties of TRSs[10].

We will deal with finite sequences of the following two kinds of symbols (and parentheses and commas for ease of reading):

(1) A finite set F of *function symbols*, and

(2) A denumerable set V of *variables*.

We assume the reader is familiar with the concepts of *terms*, *ground terms*, *occurrences*, *subterms*, *substitutions*, *unifiers*, and *most general unifiers*. In what follows, we will denote the set of all terms constructed from F and V by $\mathcal{T}(F, V)$, and the set of all the ground terms constructed from F by $\mathcal{T}(F)$. The notation $t[s]$ represents a term with 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'. Similarly, we will use the notation $t[s_1, \ldots, s_n]$ to represent a term with $s_1, \ldots, s_n$ subterms, and $t[s'_1, \ldots, s'_n]$ for the term obtained by replacing each s_i in $t[s_1, \ldots, s_n]$ with s'_i. Substitutions are denoted by the Greek letter θ, possibly with subscripts and primes.

Definition 2.1 A *term rewriting system* (TRS) is a finite set of pairs $l \to r$ of terms. An element $l \to r$ of a TRS is called a *rewrite rule*.

Definition 2.2 Let R be a TRS. A term t is said to be *reduced* to another term u with respect to R, if there is a rewrite rule $l \to r$ and a substitution θ such that $c[\theta(l)] = t$ and $c[\theta(r)] = u$, denoted by $t \Rightarrow u$. We denote the reflexive transitive closure of $\Rightarrow$ by $\stackrel{*}{\Rightarrow}$.

Definition 2.3 Let R be a TRS. Two terms u and v are said to be *convergent* (with respect to R) if there is a term t such that $u \stackrel{*}{\Rightarrow} t$ and $v \stackrel{*}{\Rightarrow} t$. A TRS is said to be *confluent* if t_1 and t_2 are convergent for any t and for any two reductions $t \stackrel{*}{\Rightarrow} t_1$ and $t \stackrel{*}{\Rightarrow} t_2$.

Definition 2.4 A TRS is said to *terminate* if there is no infinite reduction $t_1 \Rightarrow t_2 \Rightarrow \cdots \Rightarrow t_n \Rightarrow \cdots$.

Definition 2.5 A term t is said to be *irreducible* if there is no term u such that $t \Rightarrow u$. An irreducible term s such that $t \stackrel{*}{\Rightarrow} s$ is called an *irreducible form* of t (with respect to R) and is denoted by $t{\downarrow}$.

If R is a terminating TRS, then every term t has an irreducible form $t{\downarrow}$. Moreover, R is confluent if and only if the irreducible form $t{\downarrow}$ is unique. In this case, the TRS R is said to be *complete* and the irreducible form $t{\downarrow}$ is called the *normal form* of t.

Intuitively, a reduction step represents a computation step. Therefore, termination of a TRS means that every computation process finally stops and a certain result (an irreducible form) is obtained, while confluence of a TRS means that the result is unique. For this reason, completeness plays an important role in the study of TRSs (viewed as computation mechanisms) and the normal form of a term is sometimes called the value of the term.

Historically, however, the concept of TRS appeared as a decision procedure of word problems of universal algebra, where the completeness is very significant as well, because the decidability of the word problems depends on completeness of the TRS obtained by converting equational axioms to rewrite rules.

Definition 2.6 An *equational theory* is a set of pairs $t_1 \simeq t_2$[2] of terms satisfying the following conditions.

(1) $t \simeq t$ for all terms t.

(2) If $t_1 \simeq t_2$, then $t_2 \simeq t_1$.

(3) If $t_1 \simeq t_2$, $t_2 \simeq t_3$, then $t_1 \simeq t_3$.

(4) If $t_1 \simeq t_2$, then $\theta(t_1) \simeq \theta(t_2)$ for any substitution θ.

(5) If $t_1 \simeq t_2$, then $s[t_1] \simeq s[t_2]$ ∎

Any set E of pairs $l \simeq r$ of terms can be extended to an equational theory by considering the closure $T(E)$ of E with respect to the above conditions (1)-(5). In other words, the equational theory $T(E)$ is the least congruence including E. The set E is called an *(equational) axiom system* of the equational theory $T(E)$ and an element of E is called an *axiom*.

The word problem in an equational theory T involves the determination of whether $t_1 \simeq t_2$ for two arbitrary terms t_1 and t_2. Given an equational theory T, suppose that there is a complete TRS such that $t_1 \simeq t_2$ if and only if $t_1\downarrow = t_2\downarrow$ for any two terms t_1 and t_2. Obviously, such a TRS can be viewed as an algorithm to solve the word problem of T. Knuth and Bendix devised a mechanical procedure to convert an axiom system E to a complete TRS that solves the word problems of $T(E)$[11, 14].

Before introducing the procedure, let us define critical pairs.

Definition 2.7 Let $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ be rewriting rules and s be a non-variable subterm of l_2 such that l_1 and s have a most general unifier θ. Let $l_2 = c[s]$. The term $\theta(l_2)$ is called the *superposition* of l_1 on s in l_2. The pair $\theta(c[r_1]) \simeq \theta(r_2)$ is called a *critical pair* between $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ ∎

We are now ready to introduce the Knuth-Bendix completion procedure.

Procedure 2.8 Knuth-Bendix completion

Step 0: Set E to be the initially given axiom system. Set R to be empty. Go to Step 1.

Step 1: If E is empty, the current value of R is the desired TRS. Otherwise, go to Step 2.

[2] We use the symbol $\simeq$ for this purpose, and the symbol $=$ is taken to mean syntactical identity in this paper.

Step 2: Remove a pair $t \simeq u$ from E. If the rule $t \rightarrow u$ or $u \rightarrow t$ can be added to R without violating termination, acquire it as a new rule and go to Step 3. Otherwise, stop; the procedure is unsuccessful.

Step 3: Remove all the rewrite rules $l \rightarrow r$ from R such that either l or r is reducible by the acquired new rule and append $l \simeq r$ to E instead. Go to Step 4.

Step 4: Append the acquired rule to R. Construct all the critical pairs between the acquired rule and all the rules in R (including the acquired rule itself) and append them to E. For each equation $t \simeq u$ in E, find irreducible forms $t\downarrow$ and $u\downarrow$ with respect to R, and set $\{ t\downarrow \simeq u\downarrow \mid t\downarrow \neq u\downarrow, t \simeq u \in E \}$ to be the new E. Go to Step 1 ∎

If the procedure terminates successfully, the resulting R is a complete TRS to solve the word problem of $T(E)$ for the initially given E.

3 Term rewriting system generator Metis

Metis is a TRS generator based on the completion procedure described in the previous section. It has a lot of functions required before, during, and after generation of TRSs for a very user-friendly system. In this section, we will describe several characteristic features of *Metis*.

3.1 Well-founded ordering of terms

As can be seen from the above description, a key point of the completion procedure is ensuring termination of a TRS. The standard way to assure termination of a system is to introduce a well founded order on the objects of the system and show that the operations in the system always reduce the objects with respect to the order.

Well-founded orders $\prec$ on $T(F, V)$ with the following properties are usually used on TRSs.

(1) If $t_1 \prec t_2$, then $\theta(t_1) \prec \theta(t_2)$ for any substitution θ.

(2) If $t_1 \prec t_2$, then $s[t_1] \prec s[t_2]$.

Property (1) is called *stability* and (2) *monotonicity*. If there is a monotonic and stable well-founded order on $T(F, V)$ such that $l \succ r$ for every rule $l \rightarrow r$, it is obvious that the TRS terminates. There is a lot of research for such ordering methods, such as the well-known Dershowitz's recursive path ordering[4]. The original version of the recursive path ordering is defined on the set $T(F)$ of ground terms. Here, however, we extend the definition on the set $T(F, V)$ of all the terms.

Definition 3.1 (Recursive path ordering) Let $<$ be a partial order on the set of function symbols F. The recursive path ordering $\prec$ of $T(F, V)$ is then defined recursively as follows:

(1) For a variable v, there are no terms t such that $t \prec v$.

(2) For a non-variable term $t = g(t_1, \cdots, t_n)$ and a term s, $s \prec t$ if and only if

(2-1) there is j such that $s \preceq t_j$ or

(2-2) $s = f(s_1, \cdots, s_m)$ and $s_i \prec t$ for all i and

(2-2-1) $f < g$ or

(2-2-2) $f = g$ and $(s_1, \cdots, s_m) \stackrel{\cdot}{\prec} (t_1, \cdots, t_n)$, where $\stackrel{\cdot}{\prec}$ is the multi-set ordering[3] induced by $\prec$ ∎

In (2-2-2) of the above definition, employment of the multi-set ordering is not always necessary. If the function symbol f is varyadic (takes an arbitrary number of arguments) and the order of the arguments does not affect the value of the function (for example, $\sum$ and $\prod$ representing finite sum and product), the multi-set ordering is probably the most reasonable. However, if the function symbol f has a fixed arity, the lexicographic ordering is more suitable in many cases. There may be cases where the kachinuki ordering[17] is the most appropriate.

Metis can handle any of these three versions of the recursive path ordering, namely multi-set, lexicographic, and kachinuki. The user can employ arbitrary combinations of them, function by function. As long as the lexicographic order is applied only to function symbols of fixed arity, any combination defines a monotone and stable well-founded order on $\mathcal{T}(F, V)$. Moreover, if the underlying order $<$ on F is total and the lexicographic or the kachinuki ordering are employed for any function symbol, then it is a total ordering on the limited domain $\mathcal{T}(F)$ of the ground terms, a very important property as we shall see later.

Metis converts axioms to rewrite rules $l \rightarrow r$ such that $l \succ r$. *Metis* allows the user to define the underlying partial order $<$ on F incrementally during the completion procedure. If the user knows little about the above ordering method, *Metis* can suggest what ordering is needed on F in order to orient an equation to a certain direction. Thus, when both are possible, the user just has to decide in which direction an equation should be oriented.

3.2 Associative and commutative operators

The weakest point of the Knuth-Bendix completion procedure is revealed by equations that cannot be converted to rules without violating the termination of the TRS. The most typical example of such axioms is the commutative laws, such as $A + B \simeq B + A$. Encounter with such an equation causes unsuccessful stop in Step 2 of the procedure. *Metis* has several countermeasures to deal with this situation. The general measures will be described later.

It is clearly the commutativity of operators that is the main source of the above failure. In many cases, commutative operators are also associative. *Metis* has a specific countermeasure effective only against the commutative laws combined with the associative laws of the same operators. A function symbol is called an *AC-operator* if it satisfies the associative and the commutative law. *Metis* is equipped with an algorithm of special unification for AC-operators (called AC-unification) devised by Fages[5] and can execute the AC-completion procedure based on Peterson and Stickel's principle[15].

For example, if *Metis* is told that $+$ is an AC-operator, then the axioms $A + B \simeq$

$B+A$ and $(A+B)+C \simeq A+(B+C)$ are acquired implicitly and AC-unification and AC-reduction are activated for $+$. Thus, *Metis* can generate $0+Y+(-(X+Y)) \simeq (-X)+0$ as a critical pair between the same two rules $(-X)+X \rightarrow 0$ by AC-unification, since

$$(-X)+X+Y+(-(X+Y)) \Rightarrow (-X)+0$$

and

$$(-X)+X+Y+(-(X+Y)) \Rightarrow 0+Y+(-(X+Y)).$$

If it has the rule $0+A \rightarrow A$, the above critical pair is reduced to $Y+(-(X+Y)) \simeq -X$ by AC-reduction.

As shown in the above example, an AC operator is supposed to be a binary function symbol and *Metis* allows us to use infix notation for binary function symbols. Inside *Metis* an AC-operator is treated as if it were varyadic. For example, the term $t_1+\cdots+t_n$ is converted to $+(t_1,\ldots,t_2)$ with a varyadic function symbol $+$, in whatever order the operator $+$ is applied to the arguments. The multi-set ordering is assumed to be the ordering method for AC-operators unless otherwise specified, since the above treatment makes it the most reasonable ordering as mentioned in the previous section.

3.3 Orientation-free rules and S-strategy

There are many equations, other than commutative laws, that cannot be converted to terminating rules. The approach of incorporating special unification algorithms for such equations has been studied systematically by Jouannaud and Kirchner[13].

A simple trick to handle non-orientable equations is introducing a new function symbol. For example, if the equation $A^2 \simeq A \times A$ cannot be oriented in either direction, a new function symbol *square* is introduced and the problematic equation is divided into the two equations $A^2 \simeq square(A)$ and $A \times A \simeq square(A)$. Thus, *Metis* can continue the completion procedure, since both equations can be oriented left to right. This technique seems to be too simple, but the effect is worth implementation[14, 16].

A more radical remedy for such equations is adoption of orientation-free rules. This remedy is called the unfailing completion procedure[1, 9]. *Metis* is equipped with an extended version of the unfailing completion procedure called S-strategy devised by Hsiang and Rusinowitch[8]. The S-strategy has enabled *Metis* to manipulate not only non-orientable equations, but also inequational axioms as well as equational axioms.

The S-strategy can be viewed as a kind of refutational theorem proving technique for systems of equations and inequations. Before introducing the S-strategy, we will extend the concepts of reduction and critical pairs and introduce the concept of extended narrowing and subsumption. Let us fix a monotonic and stable well-founded order $\prec$ on $\mathcal{T}(F,V)$.

Definition 3.2 A term t is said to be *reduced* to another term u by an equation $l \simeq r$ (or $r \simeq l$), if $t \succ u$ and there is a substitution θ such that $c[\theta(l)] = t$ and $c[\theta(r)] = u$. This reduction is called *extended reduction* (by an equation) and is denoted also by $t \Rightarrow u$ ∎

Definition 3.3 Let $l_1 \simeq r_1$ (or $r_1 \simeq l_1$) and $l_2 \simeq r_2$ (or $r_2 \simeq l_2$) be equations. Let s be a non-variable subterm of l_2 such that l_1 and s have a most general unifier θ. Let $l_2 = c[s]$. If $\theta(l_1) \not\succ \theta(r_1)$ and $\theta(l_2) \not\succeq \theta(r_2)$, then the pair $\theta(c[r_1]) \simeq \theta(r_2)$ is called an *extended critical pair* between $l_1 \simeq r_1$ (or $r_1 \simeq l_1$) and $l_2 \simeq r_2$ (or $r_2 \simeq l_2$) ∎

If every rule $l \to r$ has the property that $l \succ r$, the above definitions are natural extensions of the ordinary reduction by a rule and the ordinary critical pairs between rules. For example, if $l \succ r$, the condition that $t \succ u$ in reducing t to u weakens the rewrite power of the equation $l \simeq r$ exactly to the same level as that of the rule $l \to r$, since $\succ$ is stable and monotonic. Similarly, if $l_1 \succ r_1$ and $l_2 \succ r_2$, the set of all extended critical pairs between equations $l_1 \simeq r_1$ and $l_2 \simeq r_2$ is equal to the set of all critical pairs between rules $l_1 \to r_1$ and $l_2 \to r_2$.

Definition 3.4 Let $l_1 \simeq r_1$ (or $r_1 \simeq l_1$) be an equation and $l_2 \not\simeq r_2$ (or $r_2 \not\simeq l_2$) be an inequation. Let s be a non-variable subterm of l_2 such that l_1 and s have a most general unifier θ. Let $l_2 = c[s]$. If $\theta(l_1) \not\succeq \theta(r_1)$, then the inequation $\theta(c[r_1]) \not\simeq \theta(r_2)$ is said to be *narrowed* from $l_2 \not\simeq r_2$ (or $r_2 \not\simeq l_2$) using $l_1 \simeq r_1$ (or $r_1 \simeq l_1$) ∎

Definition 3.5 An equation $t \simeq u$ is said to be *subsumed* by other equations $l_1 \simeq r_1$ (or $r_1 \simeq l_1$), ..., $l_n \simeq r_n$ (or $r_n \simeq l_n$), if there is a substitution θ such that

$$c[\theta(l_1), \ldots, \theta(l_n)] = t \text{ and } c[\theta(r_1), \ldots, \theta(r_n)] = u.$$

An inequation $t \not\simeq u$ is said to be *subsumed* by another inequation $l \not\simeq r$ (or $r \not\simeq l$), if there is a substitution θ such that $\theta(l) = t$ and $\theta(r) = u$ ∎

Unfailing completion is a modified version of ordinary completion employing extended critical pairs and extended reduction instead of the ordinary ones; and the S-strategy can be viewed as the unfailing completion with refutation by extended narrowing.

Procedure 3.6 (S-strategy) Suppose that a system of equational and inequational axioms is given together with an equation or inequation to be solved (called the target formula).

Step 0: Set E to be the given axiom system plus the negation of the target formula (Skolemized if necessary). Set R to be empty. Go to Step 1.

Step 1: If E is empty, the current value of R is a complete set of equations and inequations deduced from the axioms and the negation of the target formula, in the sense that neither new equations nor new inequations can be derived. Since R is also consistent, the target formula cannot be deduced from the axioms. If E is not empty, go to Step 2.

Step 2: Remove an equation $t \simeq u$ or inequation $t \not\simeq u$ (called the ruling formula) from E. Go to Step 3.

Step 3: If the ruling formula is an equation, move all the equations $l \simeq r$ and all the inequations $l \not\simeq r$ from R to E such that either l or r is reducible by the ruling formula and remove all the equations subsumed by the ruling formula from R. If the ruling formula is an inequation, remove all the inequations subsumed by the ruling formula from R. Go to Step 4.

Step 1: Append the ruling formula to R. Construct all the extended critical pairs and all the narrowed inequations between the ruling formula and all the equations and inequations in R. Append them to E. For each equation $t \simeq u$ or inequation $t \not\simeq u$ in E, find irreducible forms $t\downarrow$ and $u\downarrow$ with respect to equations in R. If there is an inequation $t \not\simeq u$ such that $t\downarrow$ and $u\downarrow$ are unifiable, then stop. A contradiction is detected and, therefore, the target formula is deduced from the originally given axiom system. Otherwise, let the new E be the set of equations $t\downarrow \simeq u\downarrow$ such that $t\downarrow \neq u\downarrow$ not subsumed by any equation in R and inequations $t\downarrow \not\simeq u\downarrow$ not subsumed by any inequation in R. Go to Step 1 ∎

The unfailing completion differs from the S-strategy only in that it does not treat non-ground inequations. If the ordering $\prec$ is total on the set $\mathcal{T}(F)$ of all the ground terms, the S-strategy is logically complete and, therefore, so is the unfailing completion.

4 Experiments

4.1 AC-Completion

Let us begin with purely algebraic examples. The first example is the word problem of ring theory.

Example 4.1 *Metis* is given an AC-operator $+$ and a binary operator $*$, (not AC in general) with the following axioms:

(1) $0 + A = A$
(2) $(-A) + A = 0$
(3) $(A * B) * C = A * (B * C)$
(4) $(A + B) * C = A * C + B * C$
(5) $A * (B + C) = A * B + A * C$

We have *Metis* run the completion procedure in automatic mode. *Metis* obtains $(A * B) * C = A * (B * C)$ and $0 + A = A$ as the first and the second ruling formulas and converts them to the rules $(A*B)*C \rightarrow A*(B*C)$ and $0+A \rightarrow A$, respectively. The third ruling formula $(-A) + A = 0$ can be oriented left to right by the recursive path ordering, if $0 < +$ or $0 < -$. So *Metis* asks the user which should be introduced.

```
[METIS] -> k s
<< Knuth-Bendix (semiautomatic execution) >>
New r1: (A*B)*C -> A*(B*C)
New r2: 0+A -> A
You can orient -A+A -> 0 by the following...
  [1]   0 << +
  [2]   0 << -
  else  exit
```

After selecting $0 < +$, we have *Metis* continue the procedure.

```
Which ? 1
```

```
[ 0 << + is asserted. ]
New r3: -A+A -> 0
New r4: -(-A) -> A
New r5: -(0) -> 0
You want to orient
  [1]   A*(B+C) -> A*B+A*C
  [2]   A*(B+C) <- A*B+A*C
  else  exit
```

The sixth ruling formula is the left distributive law and it can be oriented in either direction depending on the orderings on function symbols. Since we instruct *Metis* to convert it to the rule $A*(B+C) \rightarrow A*B+A*C$, the system automatically introduces $+ < *$ as the ordering on function symbols.

```
Which ? 1
[ + << * is asserted. ]
New r6: A*(B+C) -> A*B+A*C
New r7: (A+B)*C -> A*C+B*C
New r8: A+ -(B+A) -> -B
[ + << - is asserted. ]
New r9: -(A+(-B)) -> B+(-A)
```

The ninth ruling formula can be converted to the rule $-(A+(-B)) \rightarrow B+(-A)$ if and only if $+ < -$. So *Metis* introduces the ordering without interaction.

```
New r10: -(A+B) -> -A+(-B)
Del r8
Del r8*
Del r9
New r11: A*0+A*B -> A*B
New r12: A*0 -> 0
Del r11
Del r11*
New r13: 0*A+B*A -> B*A
New r14: 0*A -> 0
Del r13
Del r13*
New r15: (-A)*B+A*B -> 0
Which do you want to orient ?
  [1]   (-A)*B -> -A*B
  [2]   (-A)*B <- -A*B
  else  exit
select no ? 1
[ - << * is asserted. ]
New r16: (-A)*B -> -A*B
Del r15
Del r15*
New r17: A*(-B)+A*B -> 0
```

```
New r18: A*(-B) -> -A*B
Del r17
Del r17*
```

The procedure terminates successfully. Here is the resulting complete TRS for the word problem of rings.

```
[METIS] -> lis all
<< state listing >>
  "--- Ring ---"
operators:
  + / AC  ( multiset ordering )
  0 / 0
  - / 1
  * / 2   ( left to right lexicographic ordering )
orderings:
  {0}  <  "+"  <  {*,-}
  "0"  <  {+,*,-}
  {+,0}  <  "-"  <  {*}
  {+,-,0}  <  "*"
equations:
  No equations.
rules:
  r1: (A*B)*C -> A*(B*C)
  r2: 0+A -> A
  r2*:A+0+B -> A+B
  r3: -A+A -> 0
  r3*:A+(-B)+B -> A+0
  r4: -(-A) -> A
  r5: -(0) -> 0
  r6: A*(B+C) -> A*B+A*C
  r7: (A+B)*C -> A*C+B*C
  r10: -(A+B) -> -A+(-B)
  r12: A*0 -> 0
  r14: 0*A -> 0
  r16: (-A)*B -> -A*B
  r18: A*(-B) -> -A*B
```

4.2 Refutational Theorem Proving

Several examples are taken from the theory of λ-calculus and combinators[2, 7]. In the theory of combinators, the combinator $\mathbf{K} = \lambda XY.\ X$ and $\mathbf{S} = \lambda XYZ.\ X * Z * (Y * Z)$ (as usual we assume that symbols $*$ standing for application of functions are left associative) are called basic combinators because all the λ-terms without free variables can be constructed from $\mathbf{S}$ and $\mathbf{K}$ only.

Example 4.2 It is well-known that the identity $\mathbf{I} = \lambda X.\ X$ is represented by $\mathbf{S}*\mathbf{K}*\mathbf{K}$. *Metis* is given the two axioms $\mathbf{K} * X * Y = X$ and $\mathbf{S} * X * Y * Z = X * Z * (Y * Z)$ for

K and **S** to derive the identity. The problem can be expressed as $\exists I.\forall X.\ I * X = X$. *Metis* converts its negation to Skolemized form $A * \$1(A) \neq \$1(A)$ ($1 is the Skolem function).

```
[METIS] -> pro ss
<< prove formulas by S-strategy >>
Formula> some(i, all(x, i*x=x )).
New r1: A*$1(A) <-/-> $1(A)
New r2: k*A*B -> A
New r3: A <-/-> $1(k*A)
New r4: s*A*B*C <-> A*C*(B*C)
New r5: s*k*A*B -> B
    ####### PROVED #######
    By e22: $1(s*k*A) =/= $1(s*k*A) (r1%r5)
    [ i = s*k*A ]
CP(s)         : 17 found, 13 asserted.
Inequation(s): 6 deduced, 6 asserted.
Rule(s)       : 3 generated, 3 remain.
Reduction     : 1 steps.
Runtime       : 2.082 sec
```

The first ruling formula is the target formula $A * \$1(A) \neq \$1(A)$ and the second is the axiom for **K**, which is oriented left to right. The third formula is an extended narrowing from the first using the second, since $A = \mathbf{K} * A * \$1(\mathbf{K} * A) \neq \$1(\mathbf{K} * A)$. The fourth is the axiom for **S**, which could not be oriented. The fifth is an extended critical pair between the fourth and the second, since $\mathbf{S}*\mathbf{K}*A*B = \mathbf{K}*B*(A*B) = B$. Using this, a contradictory narrowing is obtained from the first ruling formula. By examining this process, we easily find all terms of the form $\mathbf{S} * \mathbf{K} * A$ are equal to the identity function, and $\mathbf{S} * \mathbf{K} * \mathbf{K}$ is merely an instance of such terms.

Example 4.3 Next, we have *Metis* try to prove the fixed-point theorem: that there is a fixed-point for any combinator if the combinators $\mathbf{B} = \lambda XYZ.\ X * (Y * Z)$ of composition of functions and $\mathbf{M} = \lambda X.\ X * X$ of self-application exist. *Metis* is given the axioms $\mathbf{B} * X * Y * Z = X * (Y * Z)$ and $\mathbf{M} * X = X * X$. The theorem can be expressed as $\forall F.\exists P.\ F * P = P$.

```
[METIS] -> list all
<< state listing >>
    "--- Fixed Point ---"
operators:
  * / 2    (*)
  b / 0
  m / 0
orderings:
  No orderings.
equations:
  e1: m*A = A*A (axiom)
  e2: b*A*B*C = A*(B*C) (axiom)
```

```
rules:
  No rules.
[METIS] -> pr s
<< prove equations by S-strategy >>
Formula> all(f, some(p, f*p=p )).
New r1: $1*A <-/-> A
New r2: m*A <-> A*A
New r3: m*$1 <-/-> $1
[ * / 2 (left to right lexicographic ordering) is asserted. ]
New r4: b*A*B*C -> A*(B*C)
New r5: m*b*A*B -> b*(A*B)
New r6: m*(b*A)*B -> A*(b*A*B)
New r7: m*(b*A*B) <-> A*(B*(b*A*B))
    ####### PROVED #######
    By e31: m*(b*$1*A) =/= A*(b*$1*A) (r1%r7)
    [ p = m*(b*$1*m) ]
CP(s)          : 33 found, 26 asserted.
Inequation(s): 3 deduced, 2 asserted.
Rule(s)        : 5 generated, 5 remain.
Reduction      : 4 steps.
Runtime        : 4.341 sec
```

Metis finally finds a contradictory inequation. The inequation e31 is from r1 and r7, since

$$\mathbf{M} * (\mathbf{B} * \$1 * A) = \$1 * (A * (\mathbf{B} * \$1 * A)) \neq A * (\mathbf{B} * \$1 * A).$$

and the rule r7 is from r2 and r4, since

$$\mathbf{M} * (\mathbf{B} * A * B) = \mathbf{B} * A * B * (\mathbf{B} * A * B) = A * (B * (\mathbf{B} * A * B)).$$

Examining this process of refutation shows us that $\mathbf{M} * (\mathbf{B} * \$1 * \mathbf{M})$ is the value substituted to the original variable A in the inequality obtained by the negation of the target formula. In fact, it is a fixed point of \$1, since

$$\$1 * (\mathbf{M} * (\mathbf{B} * \$1 * \mathbf{M})) = \mathbf{M} * (\mathbf{B} * \$1 * \mathbf{M})$$

4.3 Inductive Theorem Proving

Huet and Hullot have developed a method to prove inductive theorems without explicit induction[12] using a modified version of the Knuth-Bendix completion procedure. Their method is called *inductionless induction* and is effective for many theorems which usually require explicit induction.

In order to use the method, ground terms have to be classified into two categories, namely, *constructor terms* which are always irreducible and constructed only of special function symbols called constructors, and *non-constructor terms* which are always reducible and include a function symbol other than constructors. To prove

an inductive theorem, we add the statement as an axiom and execute the completion procedure. The statement is an inductive theorem if the process succeeds to completion without yielding any rules to rewrite constructor terms.

Metis is given an ordinary definition of the append operation for two lists and two different definitions of the reverse operation of a list.

```
[METIS] -> list rule
<< rules listing >>
    "--- Append, Reverse, and Newrev ---"
rules:
  r1: append([],A) -> A (axiom)
  r2: append([A|B],C) -> [A|append(B,C)] (axiom)
  r3: reverse([]) -> [] (axiom)
  r4: reverse([A|B]) -> append(reverse(B),[A]) (axiom)
  r5: newrev([],A) -> A (axiom)
  r6: newrev([A|B],C) -> newrev(B,[A|C]) (axiom)
```

If we define $[\_|\_]$ (*cons*) and $[]$ (*nil*) as the constructors, then the above conditions are satisfied. We add an equation $newrev(A,[]) = reverse(A)$ and have *Metis* execute the completion procedure.

```
[METIS]-> pr i
<< prove formulas by inductionless induction >>
Formula> newrev(A,[])=reverse(A).
You want to orient
  [1] newrev(A,[]) -> reverse(A)
  [2] newrev(A,[]) <- reverse(A)
  else exit
Which ? 2
[ newrev << reverse is asserted. ]
New r7: reverse(A) -> newrev(A,[])
You want to orient
  [1] append(newrev(A,[]),[B]) -> newrev(A,[B])
  [2] append(newrev(A,[]),[B]) <- newrev(A,[B])
  else exit
Which ? 1
[ newrev << append is asserted. ]
New r8: append(newrev(A,[]),[B]) -> newrev(A,[B])
(RP append("newrev(A,[B,C])",[D]) = newrev([C,B|A],[D]) )
New r9: append(newrev(A,[B]),[C]) -> newrev(A,[B,C])
(RP append("newrev(A,[B,C,D])",[E]) = newrev([C,B|A],[D,E]) )
New r10: append(newrev(A,[B,C]),[D]) -> newrev(A,[B,C,D])
(RP append("newrev(A,[B,C,D,E])",[F]) = newrev([C,B|A],[D,E,F]) )
```

By showing the formulas:

$$\begin{aligned}
(RP\ append(newrev(A,[B,C]),[D]) &= newrev([C,B|A],[D])\ ) \\
(RP\ append(newrev(A,[B,C,D]),[E]) &= newrev([C,B|A],[D,E])\ ) \\
(RP\ append(newrev(A,[B,C,D,E]),[F]) &= newrev([C,B|A],[D,E,F]),\ )
\end{aligned}$$

Metis suggests that the infinite critical pairs may be generated and, therefore, the procedure may not terminate. The general form of these formulas is

$$append(newrev(A,[B_1,\ldots,B_n]),[C]) = newrev([B_n,\ldots,B_1|A],[C])$$

and it can be reduced to

$$append(newrev(A,[B_1,\ldots,B_n]),[C]) = newrev(A,[B_1,\ldots,B_n,C]).$$

This formula suggests that the lemma:

$$append(newrev(A,B),[C]) = newrev(A,append(B,[C]))$$

would be useful, we add it.

```
[METIS/INDUCTION]-> new l
<< introduce a new lemma >>
Lemma > append(newrev(A,B),[C])=newrev(A,append(B,[C])).
New r11: append(newrev(A,B),[C]) -> newrev(A,append(B,[C]))
    ####### PROVED #######
CP(s)        . : 10 found, 4 asserted.
Rule(s)       : 5 generated, 5 remain.
Reduction     : 24 steps.
Runtime       : 2.765 sec
```

The completion terminates and, therefore, both the target statement and the lemma inserted on the way are proved to be inductive theorems.

Acknowledgments

This work was done while the first author was a member of ICOT. It is based on research into the intelligent programming systems in the FGCS project. We would like to thank Dr. Fuchi (ICOT Director) and Dr. Hasegawa (Chief of ICOT 5th Laboratory) for the opportunity to carry out this research.

References

1. L. Bachmair, N. Dershowitz, and D. A. Plaisted, Completion without failure, in *Proc. Colloquium on Resolution of Equations in Algebraic Structures* (1987).

2. H. P. Barendregt, *The lambda calculus:Its syntax and semantics, Studies in Logic and the Foundations of Mathematics* **103**, revised edition (North-Holland, 1984).

3. N. Dershowitz and Z. Manna, Proving termination with multiset orderings, *Comm. ACM* **22** (1979) pp. 465–467.

4. N. Dershowitz, Orderings for term-rewriting systems, *Theor. Comput. Sci.* **17** (1982) pp. 279–301.

5. F. Fages, Associative-commutative unification, in *Proc. 7th Int. Conf. on Automated Deduction*, Lecture Notes in Computer Science 170 (Springer-Verlag, 1984) pp. 194–208.

6. L. Fribourg, 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.

7. J. R. Hindley and J. P. Seldin, *Introduction to combinators and λ-calculus*, London Mathematical Society Student Texts 1 (Cambridge University Press, 1986).

8. J. Hsiang, Refutational Theorem Proving using Term rewriting Systems, *Artif. Intell.* **25** (1985) pp. 255–300.

9. J. Hsiang and M. Rusinowitch, On Word Problems in Equational Theories, in *Proc. 14th Int. Colloquium Automata, Languages and Programming*, Lecture Notes in Computer Science 267 (Springer-Verlag, 1987) pp. 54–71.

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

11. G. Huet, A complete proof of correctness of the Knuth-Bendix completion algorithm, *J. Comput. Syst. Sci.* **23** (1981) pp. 11–21.

12. G. Huet and J.-M. Hullot, Proofs by induction in equational theories with constructors, *J. Comput. Syst. Sci.* **25** (1982) pp. 239–266.

13. J.-P. Jouannaud and H. Kirchner, Completion of a set of rules modulo a set of equations, in *Proc. 11th ACM Symposium on Principles of Programming Languages* (1984) pp. 83–92.

14. D. E. Knuth and P. B. Bendix, Simple word problems in universal algebras, in *Proc. Computational problems in abstract algebra*, ed. J. Leech (Pergamon Press, Oxford, 1970) pp. 263–297.

15. G. E. Peterson and M. Stickel, Complete sets of reductions for equational theories with complete unification algorithms, *J. ACM* **28** (1981) pp. 233–264.

16. K. Sakai, An ordering method for term rewriting systems, ICOT Tech. Report TR-062 (1984).

17. K. Sakai, Knuth-Bendix algorithm for Thue system based on kachinuki ordering, ICOT Tech. Memorandum TM-0087 (1985).