ICOT Technical Memorandum: TM-0700

TM-0700

Efficient Implementation of Basic Narrowing

by
A. Ohsuga & K. Sakai

March, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

基底ナローイングの効率化とその実現

Efficient Implementation of Basic Narrowing

大須賀昭彦 坂井公
Akihiko OHSUGA and Kô SAKAI

(財)新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology

ABSTRACT

This paper presents an efficient method of implementing reduction and narrowing in *Metis*. *Metis* is a term rewriting system generator developed at the Institute for New Generation Computer Technology (ICOT). Its main facilities are the Knuth-Bendix completion procedure, refutational theorem proving and inductive theorem proving (inductionless induction). This method is an extension of basic narrowing techniques and we show how the method prunes redundancies of each derivation step.

1 INTRODUCTION

Metis [10] is a term rewriting system generator developed at ICOT. Its main facilities are the Knuth-Bendix completion procedure (*KB*) [7] and two theorem proving methods acquired by extending *KB*, inductive theorem proving (inductionless induction) and refutational theorem proving (S-strategy). Another *KB*-like equational theorem proving technique has been implemented in *CAP – LA*(computer aided proof system) [12], also developed at ICOT. We can see that not only our methods but also many other theorem proving methods are realized as extensions of *KB* [2]. Thus, *KB* is a kernel function in our research.

Roughly speaking, *KB* is an iteration of three processes; the orientation process of equations to obtain rewrite rules from equations, the superposition process to generate critical pairs (CPs) and the reduction process of CPs to check local confluence. In spite of the wide applicability of *KB*, it is, in general, inefficient because it often generates too many CPs iterating the superposition process. That is to say, the efficiency of *KB* is heavily dependent on the efficiency of the superposition process.

Narrowing [13, 3] has been proposed as an algorithm to solve equations in equational theories. It needs a confluent term rewriting system (TRS) that corresponds to the theory, and returns a complete set of unifiers. This algorithm is complete; however, its implementation often wastes time and space because of too many redundancies generated in derivation step. Hullot [5] has proposed "basic narrowing" to prune the unnecessary search space of the algorithm. Recently, Bosco [1] has shown more efficient and complete narrowing than unconstrained basic narrowing by exploiting the refinement that corresponds to the SLD of resolution.

By observing these two processes, superposition and narrowing, in two different contexts, we can find a similarity between them. Based on this observation, we began to study the adaptation of basic narrowing techniques for eliminating redundancies of derivation to the superposition process. Basic narrowing can be more efficient if it runs only on normal forms, i.e., terms derived by narrowing are kept in normal forms by reduction in each derivation step as *KB* usually does. As pointed out by Réty [11], however, such a combination may lose the completeness if they are combined carelessly. Réty has studied such a combination and proposed complete methods.

This paper proposes efficient new methods for reduction and narrowing to make *KB* efficient. The methods are based on basic narrowing in [1] and are complete even when combined. They are controlled with a label status instead of a basic occurrence check, and that modification of the label status is driven by an ordinary replacement of the left hand side by the right hand side of a rewrite rule. Once terms are converted to the internal format, they need neither a basic occurrence check nor maintenance of basic occurrences. Details of the adaptation of these techniques to the superposition process are described in [8].

Section 2 introduces the basic concepts and fix notations. Section 3 presents the term structure in *Metis*. Sections 4 and 5 describe the basic reduction and narrowing algorithm in detail. Section 6 describes their combination and several examples illustrate how our algorithms work.

2 PRELIMINARIES

This section introduces the terminology and notation in this paper and surveys well-known properties of TRSs. It is assumed that the reader is familiar with the basic concepts of TRSs [4].

Definition 2.1 (Terms) Let F be a finite set of *function symbols* and V be a denumerable set of *variables*. A *term* is either a variable from V or $f(t_1,\ldots,t_n)$, where $f \in F$ has arity n and each t_i is a term. We will denote the set of all terms constructed from F and V by $\mathcal{T}(F,V)$. $\mathcal{V}(t)$ denotes the set of variables occurring in term t.

Definition 2.2 (Substitutions) A substitution is a mapping from V to $\mathcal{T}(F,V)$ and is denoted by θ, possibly with subscripts and primes. The domain of θ is $\{x|\theta(x) \neq x\}$ and is denoted by $\mathcal{D}(\theta)$, the set of terms introduced by θ is denoted by $\mathcal{S}(\theta)$, i.e., $\{\theta(x)|x \in \mathcal{D}(\theta)\}$, and the set of variables introduced by θ is denoted by $\mathcal{I}(\theta)$, i.e., $\mathcal{I}(\theta) = \bigcup_{x\in\mathcal{D}(\theta)} \mathcal{V}(\theta(x))$. We write $\theta(t)$ to indicate the result of substitution θ to term t. We define quasi-ordering $\preceq$ on substitutions, i.e., $\theta \preceq \theta'$ if there exists a θ'' such that $\theta\cdot\theta'' = \theta'$, where $\theta\cdot\theta''$ is a composition of substitution, i.e., $(\theta\cdot\theta')(t)$ is $\theta(\theta'(t))$.

Definition 2.3 (Occurrence) Let t be a term. We define the set $\mathcal{O}(t)$ of occurrences as a set of finite sequences of natural numbers as follows:

(i) $\varepsilon \in \mathcal{O}(t)$.
(ii) $o \in \mathcal{O}(t_i) \Rightarrow i\cdot o \in \mathcal{O}(f(t_1,\ldots,t_n)), \forall i \in \{1,\ldots,n\}, f \in F$.

Definition 2.4 (Subterms) Let t be a term. If $o \in \mathcal{O}(t)$, we define the *subterm* of t at o as term t/o by:

(i) $t/\varepsilon = t$.
(ii) $f(t_1,\ldots,t_i,\ldots,t_n)/i\cdot o = t_i/o$.

We define the *replacement* in t of s at o as the term $t[o \Leftarrow s]$ as follows:

(i) $t[\varepsilon \Leftarrow s] = s$.
(ii) $f(t_1,\ldots,t_i,\ldots,t_n)[i\cdot o \Leftarrow s] = f(t_1,\ldots,t_i[o \Leftarrow s],\ldots,t_n)$.

We define partial prefix ordering on $\mathcal{O}(t)$: $o \leq p$ iff there exists q such that $o\cdot q = p$. We denote the set of nonvariable occurrences by $\overline{\mathcal{O}}(t)$, i.e., $\overline{\mathcal{O}}(t) = \{o \in \mathcal{O}(t)|t/o \notin V\}$.

Definition 2.5 (Equation) An *equation* is a pair, $l = r$, of terms. Let E be a finite set of equations. We define relation $=_E$ on $\mathcal{T}(T,V)$ as the compatible, stable, symmetric closure of E. An *equality* in equational theory E is the congruence generated by E and denoted by $=_E$.

Definition 2.6 (Term Rewriting System) A *term rewriting system (TRS)* is a finite set of oriented pairs, $l \triangleright r$, of terms such that $\mathcal{V}(l) \subseteq \mathcal{V}(r)$. An element, $l \triangleright r$, of a TRS is called a *rewrite rule*.

Definition 2.7 (Reduction) Let R be a TRS. A term, t, is said to be *reducible* to another term, u, at occurrence o, using rule $l \triangleright r$ in R, with substitution θ iff

(i) $t/o = \theta(l)$[1] with a substitution θ.
(ii) $u \equiv t[o \Leftarrow \theta(r)]$.

We call this relation *reduction* and denote it by $t \rightarrow_{[o,l\triangleright r,\theta]} u$, $t \rightarrow_{[o,l\triangleright r]} u$ or simply $t \rightarrow u$. We denote the reflexive transitive closure of $\rightarrow$ by $\stackrel{*}{\rightarrow}$. A term, t, is said to be *irreducible* if t is not reducible term. An irreducible term, u, such that $t \stackrel{*}{\rightarrow} u$ is called an *irreducible form* of t (with respect to R) and is denoted by $t\downarrow$.

Definition 2.8 A TRS, R, is said to be *confluent* if for any term, t, and any two reductions, $t \stackrel{*}{\rightarrow} t_1$ and $t \stackrel{*}{\rightarrow} t_2$, there exists a term, u, such that $t_1 \stackrel{*}{\rightarrow} u$ and $t_2 \stackrel{*}{\rightarrow} u$. R is said to be *terminating* if there exists no infinite reduction sequence $t_0 \rightarrow t_1 \rightarrow \cdots$. If R is a terminating TRS, then every term t has an irreducible form, $t\downarrow$. Moreover, R is confluent if and only if irreducible form $t\downarrow$ is unique. In this case, TRS R is said to be *complete* and irreducible form $t\downarrow$ is called the *normal form* of t (with respect to R).

Definition 2.9 (Narrowing) Let R be a TRS. A term, t, is said to be narrowable to another term, u, at occurrence o, using rule $l \triangleright r$ in R, with substitution θ iff

(i) $\theta(t/o) \equiv \theta(l)$ with a substitution θ.
(ii) $u \equiv \theta(t[o \Leftarrow r])$.

We call this relation *narrowing* and denote it $t \rightsquigarrow_{[o,l\triangleright r,\theta]} u$, $t \rightsquigarrow_{[o,l\triangleright r]} u$ or simply $t \rightsquigarrow u$.

Since variables in rules are universally quantified, we assume that $\mathcal{V}(t/o) \cap \mathcal{V}(l) = \phi$.

Definition 2.10 (Basic narrowing [5]) A derivation

$$t_0 \rightsquigarrow_{[o_0,l_0\triangleright r_0,\theta_0]} t_1 \rightsquigarrow \cdots \rightsquigarrow_{[o_{n-1},l_{n-1}\triangleright r_{n-1},\theta_{n-1}]} t_n$$

is said to be *basic* iff it is based on $\overline{\mathcal{O}}(t_0)$, i.e.,

(i) $O_0 = \overline{\mathcal{O}}(t_0)$.
(ii) $o_i \in O_i$ and $O_{i+1} = (O_i - \{p \in O_i | o_i \leq p\}) \cup \{o_i\cdot p | p \in \overline{\mathcal{O}}(r_i)\}$.

Definition 2.11 (E-unification) Two terms, t and u, are said to be *E-unifiable* iff there exists a substitution, θ, such that $\theta(t) =_E \theta(u)$. θ is called an *E-unifier* of t and u. $\mathcal{U}(t,u)$ denotes the set of all E-unifiers of t and u.

Definition 2.12 Let t and u be two terms and W be a finite set of variables containing $X = \mathcal{V}(t) \cup \mathcal{V}(u)$. We say that a set of substitutions, Θ, is a *complete set of E-unifiers* of t and u away from W iff:

(i) $\forall\theta \in \Theta, \mathcal{D}(\theta) \subset X$ & $\mathcal{I}(\theta) \cap W = \phi$.
(ii) $\Theta \subset \mathcal{U}(t,u)$.
(iii) $\forall\theta \in \mathcal{U}(t,u), \exists\theta' \in \Theta$ s.t. $\theta' \preceq \theta$.
Moreover, Θ is said to be *minimal* iff it satisfies the following condition:

[1] $\equiv$ denotes the syntactical equivalence of two terms.

(iv) $\forall \theta, \theta' \in \Theta, \theta \neq \theta' \Rightarrow \theta \not\leq \theta'$ and $\theta \not\geq \theta'$.

Let R be a TRS. We assume, throughout this paper, that R is complete and F is partitioned into a set of *constructors*, denoted by C, and a set of *defined operators*, denoted by D. In addition, every left hand side of rules in R is in the form $d(s_1, \ldots, s_n)$ where $d \in D$, n is an arity of d, and $s_i (i \in \{1, \ldots, n\})$ is a term. If we do not divide F between D and C explicitly, we assume that $D = F$ and $C = \phi$.

3 TERM STRUCTURE

This section introduces the internal term structure in *Metis*. In order to increase the efficiency of reduction and narrowing, a term is represented by a tree structure with labels. The label is $\perp$, $\bullet$ or $\triangledown$, and is attached to each node, which represents an occurrence of the term.

3.1 Goal terms

The label in a goal term shows the status of the subterm at the node. For example, the meaning of label $\bullet$ is that the subterm is a candidate for redex. The subterm with label $\perp$ is in normal form. $\triangledown$ means that the function symbol of the attached node is a constructor, i.e., the subterm itself is not a candidate for any redex but its (proper) subterms may be candidates.

We will denote variables on $\{\perp, \bullet, \triangledown\}$ by $\alpha, \beta, \ldots$, and this tree structure by a pair, $\alpha : t$, for all the subterms occurring in the term, where α is a label, t is a term, and : is an infix operator connecting α and t. We say α is a label of $\alpha : t$, and call a term with label α an *α term*.

The initial status of labels in a goal term is defined as follows: each variable has the label $\perp$, each constructor symbol has the label $\triangledown$, and each defined symbol has the label $\bullet$. Intuitively, the $\bullet$ label indicates the basic occurrence of basic narrowing [5] or the basic occurrence of basic-reduction, which will be defined later. The following example shows the initial status of a goal term.

Example 3.1 Let C be $\{s, 0\}$, D be $\{+\}$ and V be $\{X, \ldots\}$. Then the initial status of a goal term $s(X + 0) + s(0)$ is:

and is denoted in this paper by:

$$\bullet : +(\triangledown : s(\bullet : +(\perp : X, \triangledown : 0)), \triangledown : s(\triangledown : 0)).$$

The purpose of the following definition is to characterize the subterm which corresponds to the subterm at the basic occurrence in basic narrowing. We call such a subterm a candidate.

Definition 3.2 Let t be a term and s be a subterm of t at occurrence o. We say that s is a *candidate* (or candidate term) in t at o iff:

(i) s is a $\bullet$ term.

(ii) $\forall p, p < o \Rightarrow t/p$ is a $\bullet$ term or $\triangledown$ term

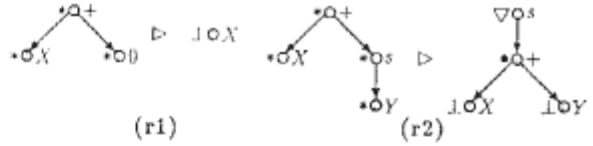
s is said to be a non-candidate if it is not a candidate.

Unlike the basic occurrences, the set of candidates does not contain occurrences of constructor symbols, and it may not be closed by a prefix. For example, the candidates in the above example are $s(X + 0) + s(0)$ and $X + 0$. Terms, $s(X + 0), 0, s(0)$ are non-candidates.

3.2 Rewrite rules

Rewrite rules are also converted into the internal format in a similar way to goal terms. Each left hand side (LHS) of the rules has a label, $\star$, called a *dummy label*. The dummy label matches with any label so that we can use syntactical (classical) unification or matching algorithms when we attempt to unify or match the LHS of the rule with a goal term. Each right hand side (RHS) of the rule is converted in the same way as goal terms.

Example 3.3 Let C and D be the ones used in example 3.1 and R be a complete TRS defining a sum of natural numbers: $R = \{X + 0 \triangleright X \cdots (r1), X + s(Y) \triangleright s(X + Y) \cdots (r2)\}$. Then the structures of these rules become:



3.3 Conversion function

Now we show the conversion function, *conv*, which attaches the label to each node of a goal term or the LHS or RHS of the rewrite rules.

Algorithm 1 (Conversion function)
Input t is a term in an ordinary structure.
The output is a converted term.

```
1. function conv(t)
2.   if t ∈ V then
3.     if LHS then
4.       return(⋆:t)
5.     else
6.       return(⊥:t)
7.   else
8.     let f(s_1, ..., s_n) := t
                 where f ∈ F, n :arity of f, s_i :subterm of t
9.     let (α_i:u_i) := conv(s_i), ∀i ∈ {1, ..., n}
10.    if LHS then
11.      return(⋆:f(α_1:u_1, ..., α_n:u_n))
12.    elseif f ∈ D then
13.      return(•:f(α_1:u_1, ..., α_n:u_n))
```

14. else % $f \in C$
15. return($\bigtriangledown : f(\alpha_1 : u_1, \ldots, \alpha_n : u_n)$)
16. end.

In our system, we need to distinguish between function symbols F used in defined symbols D and constructors C only in the context of inductive theorem proving. In other situations, such as KB and S-strategy, we assume that every function symbol is divided into the category of defined symbols, and lines 14 and 15 in algorithm 1 are no longer needed.

4 INNERMOST BASIC REDUCTION

The first step in efficient reduction is to eliminate unnecessary rule searches and attempts to match them with a goal term in each reduction step, such as rule searches for already normalized terms and rule matching to constructor terms. This can be realized by borrowing the notion of basic narrowing, i.e., a subterm is not evaluated if it is not in a basic occurrence. We call this reduction *innermost basic reduction*. In order to map the notion of basic occurrences to the internal tree structure, we define well-definedness. We say a term, t, is well-defined if the set of occurrences of candidates in t is sufficiently large on t in the sense of [11].

Definition 4.1 Let t be a term, s be a subterm of t and R be a complete TRS. t is said to be *well-defined* with respect to R iff for any subterms s of t, s has no candidates $\Rightarrow$ s is in normal form with respect to R.

Let t be a non-labeled term. Then the term, $conv(t)$ obviously satisfies a well-defined property.

We show the algorithm that returns a normal form of input terms. We call it "innermost basic reduction". In the algorithm, *DCN-select* stands for the "don't care nondeterministic selection" function used in [1].

Algorithm 2 (Innermost basic reduction)
Let R be a complete TRS.
Input $\alpha : t$ is a term to be reduced.
The output is a reduced term.

1. **function** *reduce*($\alpha : t$)
2. **while** there exists a candidate in t
3. DCN-select an innermost candidate $\bullet : s$ in $\alpha : t$ at o
4. **if** $\forall g \triangleright d \in R$, there is no θ s.t. $\theta(g) \equiv \bullet : s$ **then**
5. replace the *label* of s at o **with** $\bot$
6. **else**
7. DCN-select $g \triangleright d \in R$ s.t. $\theta(g) \equiv \bullet : s$ with substitution θ
8. **let** $\alpha : t := \alpha : t[o \Leftarrow \theta(d)]$
9. **end_while**
10. **return**($\alpha : t$)
11. **end.**

As the dummy labels on the LHS of each rule allow the use of syntactical (classical) unification or matching algorithms, we need not extend the notion of unification or matching in spite of the use of the labeled tree structure. For example, when we attempt to match the LHS of a rule, $* : +(* : X, * : 0) \triangleright \bot : X$, with a goal term, $\bullet : +(\bigtriangledown : s(\bigtriangledown : 0), \bigtriangledown : 0)$, the first matching algorithm tries to match the top operators of both terms, : with :. Secondly, it tries to match the labels of both terms, $*$ with $\bullet$. Both the first and second matching always succeed, then an attempt is made to match $+(* : X, * : 0)$ with $+(\bot : s(\bot : 0), \bot : 0)$ recursively.

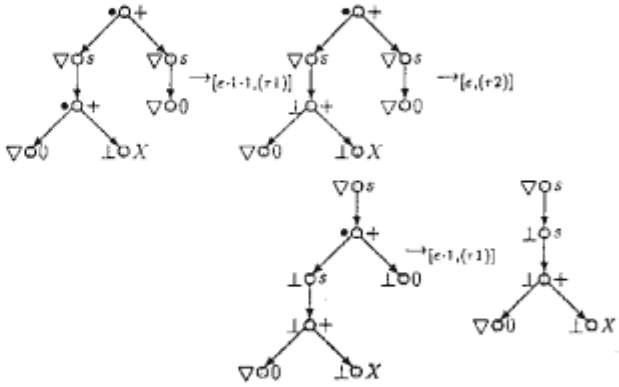
We show the completeness of this algorithm.

Lemma 4.2 (Completeness of innermost basic reduction) Let t be a term and R be a complete TRS. Then algorithm 2 always terminates and returns the normal form of an input term, t, with respect to R if t is well-defined.

Proof: Since R is complete, termination of this algorithm is guaranteed by lines 2 and 5. If the output of the algorithm is well-defined, then the lemma obviously holds. Therefore, we show that the algorithm holds a well-defined property of t_i in each execution step i. Let t_i be well-defined. Suppose that $\bullet : s_i$ be a selected candidate term in line 3. If no rule in R matches $\bullet : s_i$, then $\bullet : s_i$ is in normal form w.r.t. R since t_i is well-defined and s_i is innermost candidate. Then the term, t_{i+1}, obtained by replacing the label of $\bullet : s_i$ with $\bot$, satisfies the well-defined property. Otherwise, let $g_i \triangleright d_i$ be a selected rule and θ_i be a substitution s.t. $\theta_i(g_i) \equiv \bullet : s_i$ in line 7. Since $\bullet : s_i$ is an innermost candidate, it does not contain a reducible (and proper) subterms. Thus, for all $u \in \mathcal{S}(\theta_i)$, u is in normal form, i.e., θ_i never instantiates a reducible term to the variables in $\mathcal{V}(g_i)$ and $\mathcal{V}(d_i)$. Thus, the term t_{i+1}, obtained by replacing $\bullet : s_i$ with $\theta_i(d_i)$, still satisfies the well-defined property. ∎

Hence, the algorithm always finds the redex of an input term and reduces it. It is guaranteed to return a normal form with respect to R.

Example 4.3 The term in example 3.1 is reduced by algorithm 2 with respect to R given in example 3.3 as follows:



The goal term in example 4.3 initially has two candidates, ε and $\varepsilon \cdot 1 \cdot 1$. The algorithm selects the innermost one $(\varepsilon \cdot 1 \cdot 1)$ and tries to reduce it. Since none of the rules in R matches to the subterm, the label of the subterm is changed to $\bot$. The obtained term has only one candidate,

ε, so the algorithm immediately selects and reduces it. The next term also has only one candidate, $\varepsilon \cdot 1$, and the last term obtained by reducing the candidate has no candidate term, which means that the last term is in normal form. Thus, we can select redexes efficiently in each reduction step and find out the normality of the obtained term without unnecessary attempts to match rules with terms.

This algorithm is not complete in the sense of lemma 4.2 if a reduction strategy other than the innermost one is adopted. However, it can be extended to the complete algorithm in any strategy. See section 7.

5 SLD BASIC NARROWING

The following algorithm is the one in [1], called *SLD refinement of basic narrowing* (SLD-narrowing), to compute the E-unifiers of two input terms. We have modified it in order to exploit the internal structure of terms, and then have removed the basic occurrence check. In the algorithm, *DKN-execute* and *DKN select* stand for the "don't know nondeterministic execution" function and "don't know nondeterministic selection" function, like [1]. We introduce a meta function symbol, E, i.e., $E \notin F$, in order to handle two terms as one term. This symbol is treated in the algorithm as a constructor.

Algorithm 3 (SLD basic narrowing)
The inputs are two terms to be unified.
The output is the E-unifier of the two input terms.

1. **function** $unify(t, u)$
2. **let** $(\bigtriangledown : E(t', u'), \theta) := narrow(\bigtriangledown : E(t, u))$
3. **if** there exists θ' s.t. $\theta'(t') \equiv \theta'(u')$ **then**
4. **return**$(\theta \cdot \theta')$
5. **else**
6. failure (retry 2.)
7. **end.**

Let R be a complete TRS.
Input $\alpha : t$ is a term to be narrowed.
The output is a pair of a narrowed term and its substitution.

1. **function** $narrow(\alpha : t)$
2. **let** $\theta := \phi$
3. **while** there exists a candidate in $\lambda : t$
4. DCN-select an innermost candidate $\bullet : s$ in $\alpha : t$ at o
5. DKN-execute
6. replace the *label* of s at o with $\bot$
7. or
8. DKN-select $g \triangleright d \in R$ s.t. $\theta'(g) \equiv \theta'(\bullet : s)$ with m.g.u. θ'
9. let $\alpha : t := \theta'(\alpha : t[o \leftarrow d])$
10. let $\theta := \theta \cdot \theta'$
11. **end_**DKN-execute
12. **end_while**
13. **return**$((\alpha : t, \theta))$
14. **end.**

Lemma 5.1 (Completeness of SLD basic narrowing) Let t and u be two well-defined terms, and let R be a complete TRS and Θ be a set of all substitution θs such that $\theta := unify(t, u)$. Then Θ is a complete set of E-unifiers.

Proof: See [1]. ∎

6 COMBINATION OF INNERMOST BASIC REDUCTION AND SLD BASIC NARROWING

This section describes the combination of innermost basic reduction and SLD basic narrowing described in sections 4 and 5. Although it may prune many redundancies of derivation, a careless combination may lose the completeness in the sense of lemma 5.1 pointed out in [11]. This comes from the fact that candidates for reduction and for narrowing generally do not coincide.

6.1 Paired labels

In order to guarantee the completeness of a combined algorithm, we extend the label to an ordered pair of labels, enclosed by ⟨ and ⟩. The pair of labels in a goal term represents the status of the subterm as a single label does, the LHS of the pair is used for narrowing, and the RHS for reduction. For example, the paired label $\langle \bullet\bullet \rangle$ indicates that the subterm is a candidate for the redex of both narrowing and reduction. The subterm with the label $\langle \bullet\bot \rangle$ (resp. $\langle \bot\bullet \rangle$) is in the normal form of reduction (resp. narrowing), i.e., it is a candidate only for redex of narrowing (resp. reduction). $\langle \bigtriangledown\bigtriangledown \rangle$ means that the function symbol of the attached node is a constructor, i.e., the subterm itself is not a candidate for any redex but its (proper) subterms may be candidates.

From now on, we call this pair "label" instead of "paired label" and call the label used in the previous section a "single label". We will denote variables on $\{\bot, \bullet, \bigtriangledown\}$ by $\alpha, \beta, \ldots$, and variables which represent any label by $\lambda, \mu, \ldots$. We can define a tree structure with labels in the same way as with single labels. The initial status of labels in a goal term is defined as follows: each variable has the label $\langle \bot\bot \rangle$ instead of $\bot$, each constructor symbol has the label $\langle \bigtriangledown\bigtriangledown \rangle$ instead of $\bigtriangledown$, and each defined symbol has the label $\langle \bullet\bullet \rangle$ instead of $\bullet$.

Example 6.1 Let C, D and V be the C, D and V used in example 3.1. Then the initial status of labels in a goal term $s(X + 0) + s(0)$ becomes:

$$\langle \bullet\bullet \rangle : +(\langle \bigtriangledown\bigtriangledown \rangle : s(\langle \bullet\bullet \rangle : +(\langle \bot\bot \rangle : X, \langle \bigtriangledown\bigtriangledown \rangle : 0)), \langle \bigtriangledown\bigtriangledown \rangle : s(\langle \bigtriangledown\bigtriangledown \rangle : 0))$$

We need to extend the definition of candidate to handle the new labels as follows:

Definition 6.2 Let t be a term and s be a subterm of t at occurrence o. We say that s is an *R-candidate* (resp. *N-candidate*) in t at o iff:

(i) s is $\langle \bullet\_ \rangle$ term (resp. $\langle \_\bullet \rangle$ term)
(ii) $\forall p, p < o \Rightarrow t/p$ is $\langle \bullet\_ \rangle$ term (resp. $\langle \_\bullet \rangle$ term) or $\langle \bigtriangledown\bigtriangledown \rangle$ term.

where _ is used to denote any label value, for instance, $\langle \bullet\_ \rangle$ represents either $\langle \bullet\bullet \rangle$, $\langle \bullet\bot \rangle$ or $\langle \bullet\bigtriangledown \rangle$.

The status of labels in a rewrite rules is slightly complicated and technical in order to increase the efficiency of derivation by eliminating unnecessary candidates, keeping the well definedness in a goal term.

Let $l \triangleright r$ be a rewrite rule. Labels are attached to each node of two terms, l and r, as follows:

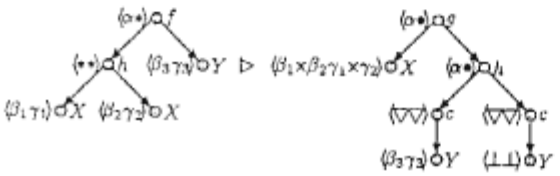
(i) Each node of l has a label $\langle **\rangle$ except that the top node of l has a label $\langle \alpha * \rangle$, where α is a label variable, and $v \in V(l)$ has a label $\langle \beta_i \gamma_i \rangle$ where i is a unique identifier for every occurrence of variables in l.

(ii) $d(s_1, \ldots, s_n)$ in r where $d \in D$ has a label $\langle \alpha \bullet \rangle$ where α is the same as the α at the top of l.

(iii) $c(s_1, \ldots, s_n)$ in r where $c \subset C$ has a label $\langle \bigtriangledown\bigtriangledown \rangle$.

(iv) For variables $v \in V(r)$, one variable v in r has a label $\langle \beta_i \gamma_i \rangle$ if v is linear in l and the unique occurrence of v in l has the label $\langle \beta_i . \rangle$. Otherwise ($v$ is non-linear in l), it has a label $\langle \beta_i \times \ldots \times \beta_j \gamma_i \times \ldots \times \gamma_j \rangle$ where $\beta_i, \ldots, \beta_j$ and $\gamma_i, \ldots, \gamma_j$ come from all labels of v occurring in l. If v is non-linear in r, then other vs have labels $\langle \perp \gamma_i \rangle$ (if v is linear in l) or $\langle \perp \gamma_i \times \ldots \times \gamma_j \rangle$ (if v if non-linear in l).

Example 6.3 Let C be $\{c\}$ and D be $\{f, g, h\}$. Then the internal structure of a rewrite rule $f(h(X, X), Y) \triangleright g(X, h(c(Y), c(Y)))$ becomes:



6.2 Unification and matching

The narrowing process consists of two processes, normalized unification and reduction. The reduction process also consists of two processes, pattern matching and replacement. In this subsection, we will extend unification and matching in order to handle our term structure efficiently.

The unification process between the goal term and LHS of a rule is made up of two procedures. The first is an ordinary unification, described in section 4, which obtains substitution θ but does not instantiate variables. The second changes the status of labels in the introduced term by $\mathcal{S}(\theta)$ and the status of labels in the goal term. The second process occurs only if the first process succeeds, and it never fails.

Let t be a goal term, s be a subterm of t, $l \triangleright r$ be a rewrite rule, and θ be a substitution obtained by the first part of unifying t and l. Then the second process replaces labels in t as follows:

(U-i) $\forall v \in \mathcal{V}(t), v \in \mathcal{D}(\theta) \Rightarrow$ replace all the label of subterms $\langle \beta . \rangle : s$ with $\langle \beta \bullet \rangle$ if s is a prefix of v.

(U-ii) $\forall v \in \mathcal{V}(t), v \in \mathcal{D}(\theta) \Rightarrow$ replace all the labels of subterms in $\theta(v)$ with

$\langle \perp \bullet \rangle$ if the subterm is in the form $d(u_1, \ldots, u_n)$ where $d \in D$.
$\langle \bigtriangledown\bigtriangledown \rangle$ if the subterm is in the form $c(u_1, \ldots, u_n)$ where $c \in C$.
$\langle \perp \perp \rangle$ if the subterm is in V.

(U-i) makes a subterm of t R-candidate only if its suffix is instantiated by θ. It keeps the set of R-candidates minimal. (U-ii) is only for making the introduced term an R-candidate.

The matching process of the LHS of a rewrite rule with a goal term is also made up of two procedures. The first is an ordinary matching procedure, described in section 4. The second is a process to instantiate label variables on the LHS. The second process occurs only if the first process succeeds, and it never fails. If the matching process succeeds, then the replacement process, which replaces the instantiated LHS in a goal term by the corresponding RIIS, makes a status of each label on the RHS dynamically as follows:

(M-i) Label variables in r take the same value as the corresponding variables in l when they are matched with the goal term label.

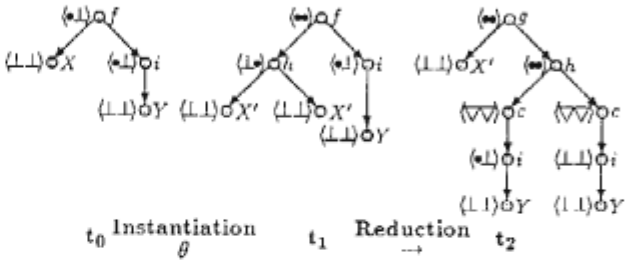
(M-ii) $\beta_i \times \ldots \times \beta_j$ becomes $\perp$ if $\exists i$ s.t. $\beta_k \equiv \perp\ k \subset \{i, \ldots, j\}$, otherwise, it takes the value of β_i.

The main role of label making is to propagate the N-candidates of a goal term to the reduced term. The following example illustrates unification and matching.

Example 6.4 Let C be $\{c\}$ and D be $\{f, g, h, i\}$, and let R consist of one rule, the same as example 6.3, i.e., $R = \{f(h(X, X), Y) \triangleright g(X, h(c(Y), c(Y)))\}$. Then a goal term:

$$\langle \bullet \perp \rangle : f(\langle \perp \perp \rangle : X, \langle \bullet \perp \rangle : i(\langle \perp \perp \rangle : Y))$$

is instantiated and reduced as follows:



In this example, $\theta = \{X \Leftarrow h(X', X')\}$ instantiates X in t_0 at $\epsilon \cdot 1$, then the label of the subterm at occurrence ϵ is changed by (U-i) to the one in t_1 at ϵ. Introduced term $h(X', X')$ in t_1 by θ is given the label status by (U-ii). In t_2, the label status of subterms at ϵ, $\epsilon \cdot 2$ and $\epsilon \cdot 2 \cdot 1 \cdot 1$ is determined dynamically by (M-i). For the variable at $\epsilon \cdot 1$ in t_2, (M-ii) makes the label status $\perp$ as a result of computing $\perp \times \perp$. Thus, the obtained term includes $\{g(X', h(c(i(Y)), c(i(Y)))), h(c(i(Y)), c(i(Y))),\ i(Y)\}$ as N-candidate terms and $\{g(X', h(c(i(Y)), c(i(Y)))),\ h(c(i(Y)), c(i(Y)))\}$ as R candidate terms.

Remark: This example only shows how instantiation and reduction work. It does not occur in our algorithm, because in the algorithm $f(X, i(Y))$ may not be selected as a candidate for narrowing before the evaluation of $i(Y)$.

6.3 Normalized SLD basic narrowing

Now we are ready to show the final algorithm, which computes E-unifiers effectively.

Algorithm 4 (Normalized SLD basic narrowing)
The inputs are two terms to be unified.
The output is the E-unifier of two input terms.

1. **function** $Nunify(t, u)$
2. let $\langle\backslash\!/\rangle : E(t', u') := reduce(\langle\backslash\!/\rangle : E(t, u))$
3. let $(\langle\!\backslash\!/\rangle : E(t'', u''), \theta) := Nnarrow(\langle\!\backslash\!/\rangle : E(t', u'))$
4. if there exists θ' s.t. $\theta'(t'') \equiv \theta'(u'')$ then
5. return$(\theta \cdot \theta')$
6. else
7. failure (retry 3.)
8. **end.**

Let R be a complete TRS.
Input $\lambda : t$ is a term to be narrowed.
The output is a pair of a narrowed term and its substitution.

1. **function** $Nnarrow(\lambda : t)$
2. let $\theta := \phi$
3. **while** there exists an N-candidate in $\lambda : t$
4. **DCN-select** an innermost N candidate $\langle \bullet\beta \rangle : s$ in $\lambda : t$ at o
5. **DKN execute**
6. replace the *label* of s at o **with** $\langle \perp\beta \rangle$
7. **or**
8. **DKN-select** $g \triangleright d \in R$ s.t. $\theta'(g) \equiv \theta'(\langle \bullet\beta \rangle : s)$ with m.g.u. θ'
9. let $\lambda : t := Nreduce(\theta'(\lambda : t[o \Leftarrow d]))$
10. let $\theta := \theta \cdot \theta'$
11. **end_DKN-execute**
12. **end_while**
13. return$((\lambda : t, \theta))$
14. **end.**

Input is a term to be reduced.
The output is a reduced term.

1. **function** $Nreduce(\lambda : t)$
2. **while** there exists a R-candidate in t
3. **DCN-select** an innermost R-candidate $\langle \alpha\bullet \rangle : s$ in $\lambda : t$ at o
4. if $\forall g \triangleright d \in R$, there is no θ s.t. $\theta(g) \equiv \langle \alpha\bullet \rangle : s$ **then**
5. replace the *label* of s at o **with** $\langle \alpha\, ! \rangle$
6. **else**
7. **DCN-select** $g \triangleright d \in R$ s.t. $\theta(g) \equiv \langle \alpha\bullet \rangle : s$ with substitution θ
8. let $\lambda : t := \lambda : t[o \Leftarrow \theta(d)]$
9. **end_while**
10. return$(\lambda : t)$
11. **end.**

This algorithm can be seen in the SLD refinement of SL-basic N-narrowing in [11]. Unlike it, our algorithm uses two kinds of occurrence set, sets of basic occurrences for reduction (corresponding to R-candidates) and for narrowing (N-candidates). A normalized goal term has no R-candidate terms, but R-candidate terms are added when substitution to goal variables occurs. N-candidates are treated in similarly to candidates in *narrow*, except that any N-candidate term in a goal term occurs only once in a reduced term as N-candidate if it is matched to a variable on the LHS which appears on the RHS at least once. Conversely, *Nnarrow* does not increase the number of N candidate terms when *Nreduce* reduces the number of non N-candidate terms.

Theorem 6.5 (Completeness of normalized basic narrowing) Let t and u be two well-defined terms, R be a complete TRS, and Θ be a set of all substitution θs such that $\theta := Nunify(t, u)$, then Θ is a complete set of E-unifiers.

Proof. See [9]. ∎

7 CONCLUSION

This paper presented several methods of implementing reduction and narrowing in *Metis*. Each derivation is an extension of basic narrowing. Their combination was also presented to realize normalized narrowing.

Innermost basic reduction in section 4 realizes an efficient reduction by eliminating unnecessary attempts to match rewrite rules with goal terms. It is not complete if reduction strategies other than the innermost one are adopted. However, it can be easily extended to complete reduction in any strategy using the techniques in section 6, which keep the well-definedness of a goal term when substitution to the variable on the LHS of a rewrite rule occurs in each reduction step. However, adoption of such a strategy may lose the efficiency which our algorithm has. For example, the outermost strategy tries to match a rule with the top of the goal term every time its subterm is reduced. In such a case, the technique used in [6] will be effective. That is, each R-candidate has a demon as its constraint if an attempt to match a rule with the node is partially successful.

SLD basic narrowing is intrinsically the same as SLD-narrowing in [1]. Our algorithm is controlled with a label status instead of a basic occurrence check, and that modification of the label status is driven by an ordinary replacement of the LHS by the RHS. Once terms are converted to the internal format, they need neither a basic occurrence check nor maintenance of basic occurrences.

Their combination is not straightforward because a careless combination of innermost basic reduction and SLD basic narrowing may lose the completeness of derivation. In order to guarantee the completeness, our algorithm needs an extension of unification and matching processes, which mainly maintain the label status of a goal term. In spite of such overheads, this algorithm is in fact efficient because it prunes many redundancies of derivations.

These algorithms have already been implemented on an experimental version of *Metis* in order to increase the efficiency of the Knuth-Bendix completion procedure by eliminating unnecessary critical pairs. In such a context, the

algorithms described in this paper can be viewed as a restriction of the procedure. Results of such criteria to restrict the superposition process are known. Compared with these results, our algorithm is less powerful but more efficient because once terms (equations) are converted to the internal format, our restriction of the reduction and narrowing (superposition) process is naturally embedded in the original processes.

ACKNOWLEDGMENTS

This work is based on the intelligent programming systems (IPS) research in the FGCS project. We would like to thank Dr. Fuchi (ICOT Director), Dr. Furukawa (ICOT Deputy Director) and Dr. Hasegawa (Chief of ICOT 1st Laboratory) for the opportunity to carry out this research.

REFERENCES

[1] Bosco, P. G., Giovannetti, E., and Moiso, C. Refined strategies for semantic unification. in *TAPSOFT*, LNCS 250, pp. 276–290, Springer-Verlag, 1987.

[2] Buchberger, B. History and Basic Features of the Critical-Pair/Completion Procedure. *J. Symbolic Computation*, 3(1&2):3–38, 1987.

[3] Fay, M. J. First Order Unification in an Equational Theory. in *4th Workshop on Automated Deduction*, pp. 161–167, 1979.

[4] Huet, G. and Oppen, D. C. Equations and Rewrite Rules: A Survey. in R. Book, editor, *Formal Languages: Perspective and Open Problems*, pp. 349–406, Academic Press, 1980.

[5] Hullot, J. M. Canonical forms and unification. in *International Conference on Automated Deduction (CADE)*, LNCS 87, pp. 318–334, Springer-Verlag, 1980.

[6] Josephson, A. and Dershowitz, N. An Implementation of Narrowing: The RITE Way. in *Symposium on Logic Programming (SLP)*, pp. 187–197, 1986.

[7] Knuth, D. E. and Bendix, P. B. Simple word problems in universal algebras. in J. Leech, editor, *Computational problems in abstract algebra*, pp. 263–297, Pergamon Press,Oxford, 1970. also in Siekmann and Wrightson, editors, *Automation of Reasoning 2*, pp. 342–376, Springer-Verlag, 1983.

[8] Ohsuga, A. *A restriction method for the superposition process.* Tech. Report (to appear), ICOT, 1988.

[9] Ohsuga, A. and Sakai, K. An Efficient Implementation Method of Reduction and Narrowing in Metis. in *Proc. 2nd Int. Workshop on Unification (UNIF)*, 1988. also Tech. Report (to appear).

[10] Ohsuga, A. and Sakai, K. Metis: A Term Rewriting System Generator. in *Symposium on Software Science and Engineering (SSE)*, RIMS, 1986. also Tech. Memoramdum TM-0226,ICOT.

[11] Réty, P. Improving basic narrowing techniques. in *Rewriting Techniques and Applications (RTA)*, LNCS 256, pp. 228–241, Springer-Verlag, 1987.

[12] Sakai, K. Toward Mechanization of Mathematics – Proof Checker and Term Rewriting System –. in K. Fuchi and M. Nivat, editors, *Programming of Future Generation Computers*, pp. 335–390, North Holland, 1988.

[13] Slagle, J. R. Automated theorem-proving for theories with simplifiers, commutativity and associativity. *J. ACM*, 21(4):622–642, 1974.