A Procedure for Solving an Order-Sorted Unification Problem — extension for left nonlinear system

Toshinori Takai, Yuichi Kaji, Takehiko Tanaka and Hiroyuki Seki
{toshin-t,kaji,takehiko,seki}@is.aist-nara.ac.jp

Graduate School of Information Science,
Nara Institute of Science and Technology,
8916-5, Takayama, Ikoma, Nara, 630-0101, Japan

Abstract

The authors have proposed a procedure for solving an order-sorted unification problem in an equational theory which is defined by a confluent TRS. Though the procedure works correctly for some kind of left-nonlinear TRS, it cannot handle left-nonlinear TRS in general. For example, the procedure assumes that a variable in the right-hand side of a rule in a given TRS does not appear more than once in its left-hand side. This paper extends the previous procedure so that it works for a general left-nonlinear order-sorted TRS. It is also shown that the procedure is sound and complete under some linearity conditions.

1 Introduction

In this paper, we propose a procedure to solve a semantical order-sorted unification problem whose semantics is defined by a confluent term rewriting system with some restrictions. For two given terms t_1 and t_2 and a confluent TRS R, the order-sorted unification problem is to decide whether there exist a term v and a well-sorted substitution σ such that $t_1\sigma \to_R^* v$ and $t_2\sigma \to_R^* v$. For unsorted case, unification problems are extensively studied by many researchers (for example, see [1, 4, 6]). On the other hand, the problem, especially a semantic unification problem, is not sufficiently studied for order-sorted case.

Order-sorted unification problems are based on order-sorted signatures. Comon[3] have pointed out that an order-sorted signature can be represented by a *tree automaton*. A set of terms is called a *tree language*. If a tree language T is accepted by some tree automaton M, that is, $T = L(M)$ where $L(M)$ is a set of terms which are accepted by M, then T is called a *regular tree language*. The class of regular tree languages is closed under intersection, and the emptiness problem is decidable for the class. For an order-sorted signature Σ and a term t, a tree automaton which corresponds to Σ can be easily extended to accept all instances of t. Therefore, the order-sorted unification problem without TRS is decidable by using the properties of regular tree languages. Procedures for solving semantic problems using tree automata have been proposed for the unsorted case in [7] and for the order-sorted case in [10]. However, [7] and [10] assume that a variable in the right-hand side of a rule does not appear more than once in its left-hand side. This paper extends the procedure in [10] to the one which is valid for a left-nonlinear order-sorted TRS.

For a term t, let $W_*(t) = \{t' \mid t\sigma \rightarrow^* t' \text{ for a substitution } \sigma\}$. $W_*(t)$ is the set of terms which are obtained by rewriting $t\sigma$ where σ is an arbitrary well-sorted substitution. If given terms t_1 and t_2 do not share a variable, then the unification problem is equivalent to decide whether

$$W_*(t_1) \cap W_*(t_2) \neq \emptyset. \tag{1.1}$$

However, the condition (1.1) is undecidable in general. Therefore, we consider the class of unification problems such that (1.1) is effectively decidable. If $W_*(t_1)$ and $W_*(t_2)$ are regular, then (1.1) is decidable. To guarantee $W_*(t_1)$ and $W_*(t_2)$ to be regular, we must assume some conditions on t_1, t_2 and R, which include the following conditions:

1. The goal terms t_1 and t_2 are linear and do not share a variable.
2. Every rule in R is right-linear.

Under these conditions, we propose a procedure for constructing two tree automata $M_*(t_1)$ and $M_*(t_2)$ that accept $W_*(t_1)$ and $W_*(t_2)$, respectively. The unification problem can be decided by checking whether $L(M_*(t_1)) \cap L(M_*(t_2)) \neq \emptyset$ or not.

The rest of this paper is organized as follows. We review an order-sorted framework and the definition of a tree automaton. In Section 3, a procedure to solve the order-sorted unification problem is presented. Because of the inherent difficulty of the problem and the limitation of the computational power of tree automata, we need some conditions on linearity of goal terms and rewriting rules. A sufficient condition for our procedure to be sound and complete is also proposed.

2 Preliminaries

2.1 Order-Sorted Unification Problem

We use the usual notions for terms, substitutions and order-sorted framework (see [4] and [8] for detail). Let $\mathcal{F}$ be a set of *function symbols* and $\mathcal{X}$ be a set of *variables*. The *arity* of $f \in \mathcal{F}$ (the number of parameters which f takes) is denoted by $a(f)$ and the set of *first-order terms*, defined in the usual way, is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{X})$ or $\mathcal{T}_{\mathcal{F}}$. A first-order term is also called an *$\mathcal{F}$-term*. The set of variables occurring in t is denoted by $\mathcal{V}ar(t)$. A term t is *ground* if $\mathcal{V}ar(t) = \emptyset$. A term is *linear* if no variable occurs more than once in the term. The set of ground $\mathcal{F}$-terms is denoted by $\mathcal{G}(\mathcal{F})$ or $\mathcal{G}_{\mathcal{F}}$. Two terms s and t *share* variables if $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) \neq \emptyset$. A *substitution* is a mapping from $\mathcal{X}$ to $\mathcal{T}_{\mathcal{F}}$, and written as a set of replacements $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where t_i with $1 \leq i \leq n$ is a term which substitutes for the variable x_i. An application of a substitution is written in the postfix notation; $t\sigma$ is the term obtained from a term t by applying a substitution σ. For terms t and t', if there exists a substitution σ such that $t = t'\sigma$, then t is an *instance* of t'. An *order-sorted signature* Σ is a 5-tuple $(\mathcal{S}, \sqsubseteq, \mathcal{F}, \mathcal{X}, \mathcal{D})$ where

- $\mathcal{S}$ is a finite set of *sort symbols*.
- $\sqsubseteq$ is a set of *subsort declarations* $S \sqsubseteq S'$ with $S, S' \in \mathcal{S}$. It is required that the reflexive and transitive closure of $\sqsubseteq$ (denoted by $\sqsubseteq$ as well) is a partial order on $\mathcal{S}$. If $S \sqsubseteq S'$, then S is *smaller* than S'. If neither $S \sqsubseteq S'$ nor $S' \sqsubseteq S$ holds, then S and S' are *incomparable*.
- $\mathcal{F}$ is a set of *function symbols*.
- $\mathcal{X}$ is a set of *variables* which can be decomposed into pairwise disjoint enumerable infinite sets $\mathcal{X}_S$ with $S \in \mathcal{S}$. A variable $x \in \mathcal{X}$ is sometimes denoted by x_S or $x{:}\, S$ if $x \in \mathcal{X}_S$.

- $\mathcal{P}$ is a set of *function declarations* $f\colon S_1 \times \cdots \times S_n \to S$ with $S_1, \ldots, S_n, S \in \mathcal{S}$, and $n = a(f)$.

The set $\mathcal{T}_{\Sigma,S}$ of Σ-*terms* (or *well-sorted terms*) of sort $S \in \mathcal{S}$ is recursively defined by the following:

(i) $x_S \in \mathcal{T}_{\Sigma,S}$.

(ii) $c \in \mathcal{T}_{\Sigma,S}$ with $a(c) = 0$, if $c\colon S \in \mathcal{P}$.

(iii) $f(t_1, \ldots, t_n) \in \mathcal{T}_{\Sigma,S}$, if $a(f) = n$, $f\colon S_1 \times \cdots \times S_n \to S \in \mathcal{P}$ and $t_i \in \mathcal{T}_{\Sigma,S_i}$ for $1 \leq i \leq n$.

(iv) $t \in \mathcal{T}_{\Sigma,S}$, if $t \in \mathcal{T}_{\Sigma,S'}$ and $S' \sqsubseteq S$.

The set of all Σ-terms are denoted by $\mathcal{T}_\Sigma(\mathcal{F}, \mathcal{X})$ or $\mathcal{T}_\Sigma$ and the set of all the *ground* Σ-*terms* are denoted by $\mathcal{G}_\Sigma(\mathcal{F})$ or $\mathcal{G}_\Sigma$. $\mathcal{F}$-terms that are not well-sorted are called *ill-sorted*. By the definition of Σ-terms, if $t \in \mathcal{T}_{\Sigma,S}$, then $t \in \mathcal{T}_{\Sigma,S'}$ for any sort S' such that $S \sqsubseteq S'$. Define $\mathcal{S}ort_\Sigma(t) = \{S \in \mathcal{S} \mid t \in \mathcal{T}_{\Sigma,S}\}$. The signature Σ is called *regular* if, for every Σ-term t, $\mathcal{S}ort_\Sigma(t)$ has a unique smallest sort. If S is the smallest among $\mathcal{S}ort_\Sigma(t)$, then we say that the sort of t is S, or t has the sort S. In the following, it is assumed that the signature Σ is regular. It can be easily shown that any order-sorted signature can be converted into a regular signature by introducing subsidiary sorts [3]. A substitution σ is called a Σ-*substitution* (or *well-sorted substitution*) if the sort of $x\sigma$ is equal to or smaller than the sort of x. For a term t, a term $t\sigma$ with Σ-substitution σ is called a Σ-*instance* of t. If t is a Σ-term and σ is a Σ-substitution, then $t\sigma$ is also a Σ-term. However, the converse is not always true; a Σ-instance of an ill-sorted term can be a well-sorted term.

Example 2.1 Let $\mathcal{S} = \{S_1, S_2\}$, $S_1 \sqsubseteq S_2$, $\mathcal{F} = \{f, c\}$, and $\mathcal{P} = \{f\colon S_1 \to S_1, c\colon S_1\}$. Consider an ill-sorted term $t = f(y\colon S_2)$ and a Σ-substitution $\sigma = \{y \mapsto c\}$, then $t\sigma = f(c)$ is a well-sorted term. □

An *occurrence* in a term t is defined as a sequence of positive integers as usual, and the set of all the occurrences in the term t is denoted by $\mathcal{O}cc(t)$. The *size* of a term t is defined to be the cardinality of $\mathcal{O}cc(t)$. If an occurrence o_1 is a prefix of o_2, that is, if o_2 is written as $o_2 = o_1 \cdot o_3$ for some o_3, then we write $o_1 \preceq o_2$. If $o_1 \preceq o_2$ and $o_1 \neq o_2$, then we write $o_1 \prec o_2$. Two occurrences o_1 and o_2 are *disjoint* if neither $o_1 \preceq o_2$ nor $o_2 \preceq o_1$. A subterm of t at an occurrence o is denoted by t/o. If a term t is obtained from a term t' by replacing subterms of t' at occurrences $o_1, \ldots, o_m$ ($o_i \in \mathcal{O}cc(t')$, o_i and o_j are disjoint if $i \neq j$) with terms $t_1, \ldots, t_m$, respectively, then we write $t = t'[o_i \leftarrow t_i \mid 1 \leq i \leq m]$.

A Σ-*rewrite rule* is an ordered pair of Σ-terms, written as $l \to r$ where l and r are Σ-terms such that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$, l is not a variable and the sort of r is equal to or less than the sort of l. In some literatures, the last condition on the sorts of l and r is called *sort-decreasing* property, and is not included in the definition of Σ-rewrite rules. However, the authors consider that this property is essential in defining the rewrite relation by Σ-rewrite rules hence included in the definition. A Σ-*term rewriting system* (Σ-*TRS*) is a finite set of Σ-rewrite rules. For Σ-terms t, t' and a Σ-TRS R, we write $t \to_R t'$ if t' can be obtained from t by one application of a Σ-rewrite rule in R, that is, there exist an occurrence $o \in \mathcal{O}cc(t)$, a Σ-substitution σ and a Σ-rewrite rule $l \to r \in R$ such that $t/o = l\sigma$ and $t' = t[o \leftarrow r\sigma]$. Define $\to_R^*$ to be the reflexive and transitive closure of $\to_R$. If $t \to_R^* t'$, then we say that t' is an R-*instance* of t. If a Σ-TRS R can be understood from the context, then the subscript R of $\to_R$ is omitted.

A Σ-TRS R is said to be *confluent* if, for Σ-terms t_1, t_2 and u such that $u \to_R^* t_1$ and $u \to_R^* t_2$, there is a Σ-term v such that $t_1 \to_R^* v$ and $t_2 \to_R^* v$. For a confluent Σ-TRS R, two Σ-terms t_1 and t_2 are R-*unifiable* if there exist a Σ-substitution σ and a Σ-term v such that $t_1\sigma \to_R^* v$ and $t_2\sigma \to_R^* v$. Remark that if $\mathcal{V}ar(t_1) \cap \mathcal{V}ar(t_2) = \emptyset$, then t_1 and t_2 are R-unifiable

if and only if there exists a Σ-term which is an R-instance of t_1 and t_2. An *order-sorted R-unification problem* (*unification problem* for short) is the decision problem such that, given an order-sorted signature Σ, a Σ-TRS R and two Σ-terms t_1 and t_2, to decide whether t_1 and t_2 are R-unifiable or not. The given Σ-terms t_1 and t_2 are called *goal terms*. In this paper, we discuss this simple yes-no version of a unification problem.

2.2 Tree Automata

A *tree automaton* is the natural extension of a usual string automaton such that the input is extended to a tree. A tree automaton is defined by a 4-tuple $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{final}, \Delta)$ where $\mathcal{F}$ is a set of function symbols, $\mathcal{Q}$ is a finite set of states, $\mathcal{Q}_{final} \subseteq \mathcal{Q}$ is a set of final states, and Δ is a finite set of transition rules. Different from string automata, no initial state is defined explicitly. There are two types of transition rules: (1) an ε-rule $q' \rightharpoonup q$ where $q, q' \in \mathcal{Q}$ and (2) the ordinary rule $f(q_1, \ldots, q_n) \rightharpoonup q$ where $f \in \mathcal{F}$, $a(f) = n$, and $q_1, \ldots, q_n, q \in \mathcal{Q}$. As a special case of the type (2), a tree automaton may have rules of the form $c \rightharpoonup q$ with $c \in \mathcal{F}$, $a(c) = 0$ and $q \in \mathcal{Q}$. Consider the set of terms $\mathcal{G}(\mathcal{F} \cup \mathcal{Q})$ where $a(q) = 0$ for $q \in \mathcal{Q}$. Terms in $\mathcal{G}(\mathcal{F} \cup \mathcal{Q})$ are called *$\mathcal{Q}$-terms*. A *$\mathcal{Q}$-substitution* is a mapping from $\mathcal{X}$ to $\mathcal{Q}$. By applying a $\mathcal{Q}$-substitution to a Σ-term, we have a $\mathcal{Q}$-term. A *move* of a tree automaton can be regarded as a 'rewriting' by considering transition rules in Δ as rewrite rules on $\mathcal{G}(\mathcal{F} \cup \mathcal{Q})$. For $\mathcal{Q}$-terms t and t', we write $t \vdash t'$ if and only if $t \rightarrow_\Delta t'$. The reflective and transitive closure of $\vdash$ is denoted by $\vdash^*$. For a tree automaton M and $t \in \mathcal{G}(\mathcal{F})$, if $t \vdash^* q_f$ for some final state $q_f \in \mathcal{Q}_{final}$, then we say t is *accepted* by M.

Example 2.2 Consider a tree automaton $M = (\{f, g, c\}, \{q_0, q_1\}, \{q_0\}, \Delta)$ where $\Delta = \{f(q_1, q_1) \rightharpoonup q_0,\ g(q_0, q_0) \rightharpoonup q_1, c \rightharpoonup q_0\}$. If a term $t = f(g(c, c), g(c, c))$ is given as an input, then the following sequence of moves

$$f(g(c, c), g(c, c)) \vdash^* f(g(q_0, q_0), g(q_0, q_0)) \vdash^* f(q_1, q_1) \vdash q_0$$

is possible. Remark that q_0 is a final state and thus t is accepted by M. $\Box$

The set of terms accepted by M is denoted by $L(M)$. A set of terms T is called a *regular tree language* if there is a tree automaton M such that $T = L(M)$. Regular tree languages inherit some useful properties of regular (string) languages. For example, the emptiness problem of the languages is decidable, and the class of regular tree languages is closed under intersection. These properties play an important role in the proposed procedure.

3 A Unification Procedure

In this section, a procedure to solve the order-sorted R-unification problem is investigated. Throughout this section, we assume that an order-sorted signature Σ, a Σ-TRS R and goal terms $G = \{t_1, t_2\}$ are given as an instance of the problem.

For a term t, define $W_0(t)$ and $W_*(t)$ as

$$W_0(t) = \{t' \mid t\sigma = t',\ \sigma \text{ is a ground } \Sigma\text{-substitution}\},\ \text{and}$$
$$W_*(t) = \{t' \mid t\sigma \rightarrow_R^* t',\ \sigma \text{ is a ground } \Sigma\text{-substitution}\}.$$

If the instance of the unification problem satisfies certain conditions which will be presented below, then we can use these sets to solve the problem effectively. Consider the following Conditions 3.1 and 3.2, then the following Lemma 3.1 holds.

Condition 3.1 The Σ-TRS R is confluent.

Condition 3.2 Goal terms t_1 and t_2 satisfy $\mathcal{V}ar(t_1) \cap \mathcal{V}ar(t_2) = \emptyset$.

Lemma 3.1 *If a given instance of the problem satisfies Condition 3.1 and 3.2, then t_1 and t_2 are R-unifiable if and only if $W_*(t_1) \cap W_*(t_2) \neq \emptyset$.*$\Box$

It is undecidable whether $W_*(t_1) \cap W_*(t_2) \neq \emptyset$ or not, even if Conditions 3.1 and 3.2 are satisfied. However, for a class of instances such that $W_*(t_1)$ and $W_*(t_2)$ are regular tree languages, the problem is decidable thanks to the favorable properties of regular tree languages. In the following, we investigate sufficient conditions under which $W_*(t_1)$ and $W_*(t_2)$ are regular, and devise procedure to construct tree automata that accept $W_*(t_1)$ and $W_*(t_2)$.

In Section 3.1, a tree automaton $M_0(t)$ is constructed for each subterm t of goal terms so that $L(M_0(t)) = W_0(t)$. In Section 3.2, we propose a procedure to construct a tree automaton $M_*(t)$ such that $L(M_*(t)) = W_*(t)$ by modifying $M_0(t)$ under some additional conditions.

3.1 Basic Tree Automata

In this section, for a subterm t of goal terms, a tree automaton $M_0(t)$ such that $L(M_0(t)) = W_0(t)$ is constructed under the following Condition 3.3.

Condition 3.3 Goal terms t_1 and t_2 are linear.

The condition guarantees that the sets $W_0(t_1)$ and $W_0(t_2)$ are regular tree languages. Without this condition, it can happen that the sets become non-regular and cannot be dealt with tree automata. It is also notable that Conditions 3.2 and 3.3 imply that no variable occurs more than once in goal terms. Therefore, every variable in goal terms can be replaced independently. In such a situation, only the occurrences and sorts of variables in a term are significant, but names of variables are of no use. Therefore we introduce a concept of Ω-terms. For a Σ-term t, the *Ω-term* of t, denoted t_Ω, is defined as a term which is obtained from t by replacing each variable in t with the sort symbol of the variable. For example, if $t = f(g(x{:}\, S_1), y{:}\, S_2)$, then $t_\Omega = f(g(S_1), S_2)$. A ground Σ-term s is said to be an instance of an Ω-term t if there is a Σ-term t' such that $t'_\Omega = t$ and s is a Σ-instance of t'.

In the following, we extend a first-order term to have a set of terms at its subterm occurrences. Formally, the set of *packed terms*, denoted $\mathcal{P}_{\mathcal{F},\mathcal{S}}$, is defined as follows.

- $\{t\} \in \mathcal{P}_{\mathcal{F},\mathcal{S}}$ for any Ω-term t.
- if $q_1, q_2 \in \mathcal{P}_{\mathcal{F},\mathcal{S}}$, then $q_1 \cup q_2 \in \mathcal{P}_{\mathcal{F},\mathcal{S}}$.
- if $f \in \mathcal{F}$ and $q_1, \ldots, q_{a(f)} \in \mathcal{P}_{\mathcal{F},\mathcal{S}}$, then $\{f(q_1, \ldots, q_{a(f)})\} \in \mathcal{P}_{\mathcal{F},\mathcal{S}}$.

For the readability, a packed term $\{t_1, \ldots, t_n\}$ is written as $\langle t_1, \ldots, t_n \rangle$ and a packed term with only one element $\langle t \rangle$ is written as t. We can easily verify that $f(g(S_1), S_2)$, $\langle f(g(S_1), S_2), g(g(S_1)), g(S_2) \rangle$, $f(S_1, \langle g(S_1), S_2 \rangle)$ and $g(\langle f(S_1, S_2),\ g(\langle g(S_1), g(S_2) \rangle) \rangle)$ belong to $\mathcal{P}_{\mathcal{F},\mathcal{S}}$. The *extension* of a packed term q, denoted $E(q)$, is a set of Ω-terms defined as follows.

- if $q = \langle t \rangle$ with t an Ω-term, then $E(q) = \{t\}$.
- if $q = \langle q_1, \ldots, q_n \rangle$, then $E(q) = \bigcup_{i=1}^{n} E(q_i)$.
- if $q = f(q_1, \ldots, q_n)$ with $f \in \mathcal{F}$ and $a(f) = n$, then

$$E(q) = \{f(t_1, \ldots, t_n) \mid t_i \in E(q_i) \text{ for } 1 \leq i \leq n\}.$$

Thus the extensions of the above packed terms are respectively $\{f(g(S_1), S_2)\}$, $\{f(g(S_1), S_2), g(g(S_1)), g(S_2)\}$, $\{f(S_1, g(S_1)), f(S_1, S_2)\}$ and $\{g(f(S_1, S_2)), g(g(g(S_1))), g(g(g(S_2)))\}$.

In the following, we consider a class of tree automata. These automata have the same sets of function symbols, states and transition rules, but different final states. They have packed terms as states and have only one final state. For each state (packed term) τ in the state set, $M(\tau)$ is defined to be a tree automaton which has τ as the only final state. $M(\tau)$ is constructed so that $s \in L(M(\tau))$ if and only if s is an instance of any $t \in E(\tau)$, and therefore $s \in \bigcap_{t \in E(\tau)} W_0(t)$.

For a goal term t, the sets of states and transition rules of $M_0(\langle t_\Omega \rangle)$ which accepts $W_0(t)$ are defined as follows.

(i) for each subsort declaration $S \sqsubseteq S'$, define $\langle S \rangle$ and $\langle S' \rangle$ as states and also define $\langle S \rangle \rightharpoonup \langle S' \rangle$ as a transition rule.

(ii) for each function declaration $f(x_1{:}\, S_1, \ldots, x_1{:}\, S_n){:}\, S$, let $f(\langle S_1 \rangle, \ldots, \langle S_n \rangle) \rightharpoonup \langle S \rangle$,

(iii) for a goal term t, execute **ADDTRANS**$(\langle t_\Omega \rangle)$ shown below.

Procedure 3.1 [**ADDTRANS**] This procedure takes a packed term τ as an input. If the input τ has been already introduced as a state, then the procedure defines no transitions. If τ is not a state, then the procedure first defines τ as a new state and also defines transition rules as follows. It is required that if $\tau = \langle t_1, \ldots, t_n \rangle$, then each $\langle t_i \rangle$ has been introduced as a state.

Case 1. If $\tau = \langle c \rangle$ with c a constant in $\mathcal{F}$, then, define $c \rightharpoonup \langle c \rangle$ as a transition rule.

Case 2. If $\tau = \langle f(\tau_1, \ldots, \tau_n) \rangle$ with $f \in \mathcal{F}$ and $a(f) = n$, define $f(\tau_1, \ldots, \tau_n) \rightharpoonup \tau$ as a transition rule and execute **ADDTRANS**(τ_i) for $1 \leq i \leq n$.

Case 3. If $\tau = \langle t_1, \ldots, t_n \rangle$, then do the following (i) and (ii).

(i) For each transition rule of the form $\tau' \rightharpoonup t_i$ ($\tau' \in \mathcal{Q}, 1 \leq i \leq n$), define a new ε-rule $\tau'' \rightharpoonup \tau$ and execute **ADDTRANS**(τ'') where τ'' is the packed term defined as $\tau'' = \tau' \cup \tau \setminus \langle t_i \rangle$.

(ii) If there are a function symbol f and states τ_{ij} with $1 \leq i \leq n, 1 \leq j \leq a(f)$ such that $f(\tau_{i1}, \ldots, \tau_{ia(f)}) \rightharpoonup \langle t_i \rangle$ for $1 \leq i \leq n$, then define a new rule $f(\tau_1, \ldots, \tau_{a(f)}) \rightharpoonup \tau$ and execute **ADDTRANS**(τ_j) for $1 \leq j \leq a(f)$ where $\tau_j = \tau_{1j} \cup \cdots \cup \tau_{nj}$.

□

For the tree automata defined above, the following theorem and lemmas hold.

Theorem 3.2 *For every subterm t of goal terms, $L(M_0(t)) = W_0(t)$ holds. That is, $s \in W_0(t)$ if and only if $M_0(t)$ accepts s.*□

Proof. Observe that while constructing the tree automata M_0, Case 3 of **ADDTRANS** is never used. Hence, the construction is essentially the same as that of [3, 7], and the the theorem holds. □

The following lemmas are easily understood by the construction of rewrite rules (Case 3(i) of ADDTRANS).

Lemma 3.3 *For a positive integer n and n pairs of states $\langle t_i \rangle$ and $\langle t'_i \rangle$ with $1 \leq i \leq n$, if $\langle t_i \rangle \vdash_M^* \langle t'_i \rangle$ for $1 \leq i \leq n$, then $\langle t_1, \ldots, t_n \rangle \vdash_M^* \langle t'_1, \ldots, t'_n \rangle$.* □

Lemma 3.4 *If* $\langle t_1, \ldots, t_m \rangle \vdash_M^* \langle t'_1, \ldots, t'_n \rangle$ *where* m *and* n *are positive integers, then, for each* t'_i *with* $1 \leq i \leq n$, *there is* t_j *with* $1 \leq j \leq m$ *such that* $\langle t_j \rangle \vdash_M^* \langle t'_i \rangle$. $\Box$

Lemma 3.5 *Let* $M_0 = (\mathcal{F}, \mathcal{Q}_0, \mathcal{Q}_{final}, \Delta_0)$ *be the tree automaton defined above, and let* $M = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{final}, \Delta)$ *be a tree automaton satisfying the followings:*

1. $\mathcal{Q}_0 \subseteq \mathcal{Q}$ *and* $\Delta_0 \subseteq \Delta$.
2. $\mathcal{Q}$ *is a set of packed terms.*
3. *If* $q \in \mathcal{Q}$ *is of the form* $q = \langle t_1, \ldots, t_m \rangle$ *with* $m > 1$, *then* **ADDTRANS**(q) *has been executed.*

Then, for each state $\langle t_1, \ldots, t_n \rangle$ *of the tree automata* M, $L(M(\langle t_1, \ldots, t_n \rangle)) = L(M(\langle t_1 \rangle)) \cap \cdots \cap L(M(\langle t_n \rangle))$.

Proof. We prove that for every ground Σ-term s, $s \vdash_M^* \langle t_1, \ldots, t_n \rangle$ if and only if $s \vdash_M^* \langle t_i \rangle$ for $1 \leq i \leq n$. First the 'if' part will be shown by induction on the depth of the terms. If $s = c$ with $a(c) = 0$, then, the sequence $s \vdash_M^* \langle t_i \rangle$ can be written as

$$c \vdash_M \langle c \rangle \vdash_M^* \langle t_i \rangle \quad (1 \leq i \leq n) \tag{3.1}$$

Remark that, for a constant c, there is only one rule $c \rightharpoonup \langle c \rangle$. Since the moves in the sequence $\langle c \rangle \vdash_M^* \langle t_i \rangle$ in (3.1) are all ε-moves, transition rules are defined so that $\langle c \rangle \vdash_M^* \langle t_1, \ldots, t_n \rangle$ by Lemma 3.3. Therefore $c \vdash_M \langle c \rangle \vdash_M^* \langle t_1, \ldots, t_n \rangle$.

Assume that the if part holds for every term with depth $k-1$ or less, and consider a term $s = f(s_1, \ldots, s_{a(f)})$ with depth k. The sequences $s \vdash_M^* \langle t_i \rangle$ can be written as

$$s \vdash_M^* f(q_{i1}, \ldots, q_{ia(f)}) \vdash_M \langle t'_i \rangle \vdash_M^* \langle t_i \rangle \quad (1 \leq i \leq n) \tag{3.2}$$

where q_{ij} $(1 \leq j \leq a(f))$ and $\langle t'_i \rangle$ are states. This implies $s_j \vdash_M^* q_{ij}$ for $1 \leq i \leq n$ and $1 \leq j \leq a(f)$, and therefore $s_j \vdash_M^* \langle q_{ij}, \ldots, q_{nj} \rangle$ for $1 \leq j \leq a(f)$ by the induction hypothesis. Hence the sequence

$$s \vdash_M^* f(\langle q_{11}, \ldots, q_{n1} \rangle, \ldots, \langle q_{1a(f)}, \ldots, q_{na(f)} \rangle) \tag{3.3}$$

is possible. On the other hand, by the sequences $f(q_{i1}, \ldots, q_{ia(f)}) \vdash_M \langle t'_i \rangle$ with $1 \leq i \leq n$ of (3.2) and (ii) of **ADDTRANS**, the transition rule

$$f(\langle q_{11}, \ldots, q_{n1} \rangle, \ldots, \langle q_{1a(f)}, \ldots, q_{na(f)} \rangle) \rightharpoonup \langle t'_1, \ldots, t'_n \rangle \tag{3.4}$$

is defined. Furthermore, since all the moves in sequence $\langle t'_i \rangle \vdash_M^* \langle t_i \rangle$ for $1 \leq i \leq n$ of (3.2) are ε-moves, transition rules are defined so that the sequence

$$\langle t'_1, \ldots, t'_n \rangle \vdash_M^* \langle t_1, \ldots, t_n \rangle \tag{3.5}$$

is possible (Lemma 3.3). Summarizing (3.3),(3.4) and (3.5), we have $s \vdash_M^* \langle t_1, \ldots, t_n \rangle$.

Next, we show the 'only if' part by induction on the depth of an input term. If $s = c$ with $a(c) = 0$, then the sequence $s \vdash_M^* \langle t_1, \ldots, t_n \rangle$ can be written as $c \vdash_M \langle c \rangle \vdash_M^* \langle t_1, \ldots, t_n \rangle$. By Lemma 3.4, we have $\langle c \rangle \vdash_M^* \langle t_i \rangle$ for $1 \leq i \leq n$, and therefore $c \vdash_M^* \langle t_i \rangle$ for $1 \leq i \leq n$.

Assume that the only if part holds for any term with depth $k-1$ or less, and consider a term $s = f(s_1, \ldots, s_{a(f)})$ with depth k. The sequence $s \vdash_M^* \langle t_1, \ldots, t_n \rangle$ can be written as

$$s \vdash_M^* f(q_1, \ldots, q_{a(f)}) \vdash_M \langle t'_1, \ldots, t'_m \rangle \vdash_M^* \langle t_1, \ldots, t_n \rangle \tag{3.6}$$

where q_i $(1 \leq i \leq a(f))$ and $\langle t'_1, \ldots, t'_m \rangle$ are states. By Lemma 3.4, for each t_i $(1 \leq i \leq n)$, there is an integer j_i with $1 \leq j_i \leq m$ such that

$$\langle t'_{j_i} \rangle \vdash^*_M \langle t_i \rangle. \tag{3.7}$$

On the other hand, the move $f(q_1, \ldots, q_{a(f)}) \vdash_M \langle t'_1, \ldots, t'_m \rangle$ is enabled by a transition rule defined by (ii) of **ADDTRANS**. Therefore, there must be states q'_{ik} with $1 \leq i \leq n$ and $1 \leq k \leq a(f)$ and transition rules such that

$$f(q'_{i1}, \ldots, q'_{ia(f)}) \rightharpoonup \langle t'_i \rangle \quad (1 \leq i \leq n) \tag{3.8}$$

and $q'_{1k} \cup \cdots \cup q'_{nk} = q_k$ for $1 \leq k \leq a(f)$. Since $s_k \vdash^*_M q_k$, we have $s_k \vdash^*_M q'_{ik}$ for $1 \leq i \leq n$ by the induction hypothesis. Hence

$$s \vdash^*_M f(q'_{i1}, \ldots, q'_{ia(f)}) \quad (1 \leq i \leq n). \tag{3.9}$$

Summarizing (3.7),(3.8) and (3.9), we have $s \vdash^*_M \langle t_i \rangle$ for $1 \leq i \leq n$ and the lemma holds. □

Example 3.1 Let $\Sigma = \{S_1 \sqsubseteq S_0,\ a\colon S_0,\ b\colon S_1,\ f(x_{S_1})\colon S_1,\ f(x_{S_0})\colon S_0\ \}$, and the goal terms $t_1 = x_{S_1}$ and $t_2 = f(y_{S_0})$. In this case, $\mathcal{Q}_0 = \{\ \langle S_0 \rangle, \langle S_1 \rangle, \langle f(S_0) \rangle\ \}$ and Δ_0 is defined as follows.

$$\begin{aligned} \langle S_1 \rangle &\rightharpoonup \langle S_0 \rangle, \\ a &\rightharpoonup \langle S_0 \rangle, \\ b &\rightharpoonup \langle S_1 \rangle, \\ f(\langle S_1 \rangle) &\rightharpoonup \langle S_1 \rangle, \\ f(\langle S_0 \rangle) &\rightharpoonup \langle S_0 \rangle, \\ f(\langle S_0 \rangle) &\rightharpoonup \langle f(S_0) \rangle. \end{aligned}$$

The sets of final states of $M_0(t_1)$ and $M_0(t_2)$ are $\mathcal{Q}^1_{final} = \{\langle S_1 \rangle\}$ and $\mathcal{Q}^2_{final} = \{\langle f(S_0) \rangle\}$, respectively. If a term $f(b)$ is given to the tree automaton $M_0(t_2)$, then the following accepting sequence of moves is possible:

$$f(b) \vdash_{M_0} f(\langle S_1 \rangle) \vdash_{M_0} f(\langle S_0 \rangle) \vdash_{M_0} \langle f(S_0) \rangle.$$

Note that $\langle f(S_0) \rangle \in \mathcal{Q}^2_{final}$ and thus $f(b) \in W_0(t_2)$. □

3.2 Modification of Tree Automata

In this section, the tree automaton $M_0(t)$ defined in the previous section is modified to accept $W_*(t)$. For this sake, the set $W_*(t)$ must be a regular tree language. In addition to Condition 3.1 to Condition 3.3, we assume the following condition about linearity of rules, and restrict the scope of the procedure to rewrite rules that satisfy these conditions. As shown in Theorem 3.9 and Theorem 3.10, the procedure constructs a tree automaton which accepts $W_*(t)$ (if it halts).

Condition 3.4 Every rule in R is right-linear.

Procedure 3.2

Input: a signature Σ, a Σ-TRS R, and terms $G = \{t_1, t_2\}$ that satisfy Conditions 3.1 through 3.4. Let γ be the smallest integer such that, for every rule $l \to r \in R$, no variable has more than γ occurrences in l.

Output: "*unifiable*" or "*not unifiable*".

This procedure has a loop-structure to expand the set of states and the set of transition rules. For a nonnegative integer k and every subterm t of goal terms, let $M_k(t)$ be the tree automaton which is obtained from $M_0(t)$ by executing the loop k times. We write $\vdash_k$ for $\vdash_{M_k}$.

Step 1. Let $k := 0$. This k is used as a loop counter of the procedure.

Step 2. Construct $M_0 = (\mathcal{F}, \mathcal{Q}_0, \mathcal{Q}_{final}, \Delta_0)$ as in the previous section.

Step 3. Let $\mathcal{Q}_{k+1} := \mathcal{Q}_k$ and $\Delta_{k+1} := \Delta_k$.

Step 4. The set of transition rules is modified at this step. Without loss of generality, let $l \to r$ be a rewrite rule in R such that l has $m(\geq 0)$ variables $x_1, \ldots, x_m$ of sorts $S_1, \ldots, S_m$, respectively, and the variable x_i has γ_i occurrences $p_i^j \in \mathcal{O}cc(l)$ $(1 \leq j \leq \gamma_i)$ in l. If there are states q and q_i^j $(1 \leq i \leq m, 1 \leq j \leq \gamma_i)$ in $\mathcal{Q}_k$ such that

$$l[p_i^j \leftarrow q_i^j \mid 1 \leq i \leq m, 1 \leq j \leq \gamma_i] \vdash_k^* q \tag{3.10}$$

and

$$L(M_k(q_i^1)) \cap \cdots \cap L(M_k(q_i^{\gamma_i})) \cap L(M_k(\langle S_i \rangle)) \neq \emptyset \tag{3.11}$$

for $1 \leq i \leq m$, then define $q_i = \bigcup_{1 \leq j \leq \gamma_i} q_i^j \cup \{S_i\}$ $(1 \leq i \leq m)$ as new states and a $\mathcal{Q}_k$-substitution $\rho = \{x_i \mapsto q_i\}$, and do the following (a) and (b).

(a) Add the transition rule $\langle r\rho \rangle \rightharpoonup q$ to Δ_{k+1}. If a move of the tree automaton is caused by this rule, then the move is called a *rewriting move* of *degree* $k+1$.

(b) Execute **ADDTRANS**$(\langle r\rho \rangle)$ (see Procedure 3.1).

Simultaneously execute this Step 4 for every rewrite rule and every tuple of states that satisfy conditions (3.10) and (3.11).

Step 5. Continue the loop until $\Delta_{k+1} = \Delta_k$. If $\Delta_{k+1} \neq \Delta_k$, then $k := k+1$ and go to Step 3.

Step 6. For each $q \in \mathcal{Q}_k$, let $M_*(q) := M_k(q)$, and output "*unifiable*" if $L(M_*(t_1)) \cap L(M_*(t_2)) \neq \emptyset$, and "*not unifiable*" otherwise.□

Example 3.2 Let

$$\begin{aligned} \Sigma &= \{S_1 \sqsubseteq S_0, a\colon S_0, c\colon S_1, f(x_{S_0}, y_{S_0})\colon S_0, g(x_{S_0})\colon S_1\}, \\ R &= \{f(x_{S_1}, x_{S_1}) \to g(x_{S_1}),\ g(x_{S_1}) \to x_{S_1}\} \quad \text{and} \\ G &= \{f(c, g(y_{S_0})), c\} \end{aligned}$$

be an input of the unification problem. The basic tree automata M_0 is constructed as follows. $\mathcal{Q}_0 = \{\ \langle f(c, g(S_0))\rangle, \langle g(S_0)\rangle, \langle c\rangle, \langle S_1\rangle, \langle S_0\rangle\ \}$,

$$\begin{aligned} \langle S_1 \rangle &\rightharpoonup \langle S_0 \rangle, \\ a &\rightharpoonup \langle S_0 \rangle, \\ c &\rightharpoonup \langle S_1 \rangle, \\ f(\langle S_0 \rangle, \langle S_0 \rangle) &\rightharpoonup \langle S_0 \rangle, \\ g(\langle S_0 \rangle) &\rightharpoonup \langle S_1 \rangle, \\ f(\langle c \rangle, \langle g(S_0) \rangle) &\rightharpoonup \langle f(c, g(S_0)) \rangle, \\ g(\langle S_0 \rangle) &\rightharpoonup \langle g(S_0) \rangle, \\ c &\rightharpoonup \langle c \rangle. \end{aligned}$$

Remark that

$$\begin{aligned} L(M_0(\langle f(c, g(S_1)\rangle))) &= \{f(c, g(t)) \mid \ t \text{ have the sort } S_0\} \quad \text{and} \\ L(M_0(\langle c\rangle)) &= \{c\} \end{aligned}$$

are the set of ground Σ-instances of goal terms $f(c, g(y_{S_0}))$ and c, respectively.

Next, consider the rewrite rule $g(x_{S_1}) \to x_{S_1}$ at Step 4. Then the following accepting sequence of moves

$$g(\langle c\rangle) \vdash_0 g(\langle S_1\rangle) \vdash_0 g(\langle S_0\rangle) \vdash_0 \langle g(S_0)\rangle$$

is possible so that a new transition rule $\langle c\rangle \rightharpoonup \langle g(S_0)\rangle$ is added.

The case of the rule $f(x_{S_1}, x_{S_1}) \to g(x_{S_1})$ is more complicated. The following accepting sequence of moves

$$f(\langle c\rangle, \langle g(S_0)\rangle) \vdash_1 \langle f(c, g(S_0))\rangle$$

is possible and

$$L(M_1(\langle c\rangle)) \cap L(M_1(\langle g(S_0)\rangle)) \cap L(M_1(\langle S_0\rangle)) \neq \emptyset$$

holds. By Step 4, new transition rules are added to Δ_1 in order to satisfy that the following sequence is possible:

$$g(\langle c, g(S_0), S_0\rangle) \quad \vdash_2^* \quad \langle f(c, g(S_0))\rangle$$

where $\langle c, g(S_0), S_0\rangle$ is a new state such that

$$L(M_2(\langle c, g(S_0), S_0\rangle)) = L(M_2(\langle c\rangle)) \cap L(M_*(\langle g(S_0)\rangle)) \cap L(M_2(\langle S_0\rangle)).$$

Remark that $c \in L(M_2(\langle c, g(S_0), S_0\rangle))$. Again consider the rule $g(x_{S_1}) \to x_{S_1}$, then the following sequence is possible:

$$g(\langle c, g(S_0), S_0\rangle) \vdash_2 \langle f(c, g(S_0))\rangle$$

so that the new transition rule

$$\langle c, g(S_0), S_0\rangle \rightharpoonup \langle f(c, g(S_0))\rangle$$

is added.

Consider $L(M_3(f(c, g(x_{S_0}))))$. The sequence

$$c \vdash_3^* \langle c, g(S_0), S_0\rangle \vdash_3 \langle f(c, g(S_0))\rangle$$

is possible and hence $c \in L(M_3(f(c, g(x_{S_0})))) \cap L(M_3(c))$. That is, the goal terms in G are unifiable. $\Box$

We have constructed new tree automata M_k ($k \geq 1$) by adding new states and new transition rules to M_0. The essential difference between these modified tree automata M_k and the basic tree automaton M_0 is that the former allows rewriting moves. If modified tree automata M_k do not use rewriting moves, then M_0 and M_k have essentially the same property. Thereby, the following lemma holds as a generalization of Theorem 3.2.

Lemma 3.6 *Consider moves of the tree automaton $M_k(q)$ for a ground Σ-term s where q is an arbitrary state. The sequence $s \vdash_0^* q$ is possible if and only if there is an accepting sequence $s \vdash_k^* q$ that does not contain rewriting moves (of any degree).*

Proof. A sequence of moves does not contain rewriting moves if and only if every transition rule used in the sequence is defined by **ADDTRANS**. The lemma holds obviously by the construction of transition rules in **ADDTRANS**. □

If a sequence of moves of the tree automaton M_k for a term s contains no rewriting move, then the sequence has certain simple properties. One of such properties is that, for some occurrence o of s, there is strong relation between the term s/o and the state when a head of M_k visits o. This property is formally stated in the following lemma.

Lemma 3.7 *Let r be a linear Σ-term that has m variables $x_1, \ldots, x_m$ at $o_1, \ldots, o_m$ respectively, s be a ground Σ-term and ρ be the $\mathcal{Q}_k$-substitution $\{x_i \mapsto q_i \mid 1 \leq i \leq m\}$ such that $q_i \in \mathcal{Q}_k$ $(1 \leq i \leq m)$ and $\langle r\rho \rangle \in \mathcal{Q}_k$. If s is accepted by a tree automaton $M_k(\langle r\rho \rangle)$, and if the accepting sequence $s \vdash_k^* \langle r\rho \rangle$ does not contain rewriting moves, then the sequence can be written as $s \vdash_k^* s[o_i \leftarrow q_i \mid 1 \leq i \leq m] \vdash_k^* \langle r\rho \rangle$. That is, when a head visits o_i $(1 \leq i \leq m)$, the state q_i is associated with the head.*

Proof. By construction of tree automata there is a sequence of moves written as $s \vdash_k^* s[o_i \leftarrow q'_i \mid 1 \leq i \leq m] \vdash_k^* \langle r\rho \rangle$ where $q'_1, \ldots, q'_m \in \mathcal{Q}_k$. Let o be an arbitrary occurrence in r. We can show by induction on the length of o that when a head of the tree automaton visits the occurrence o in s, the state q_i is attached to that head, and there is no other possibility. Hence, at the occurrence o_i $(1 \leq i \leq m)$, the state $q'_i = q_i$ is attached to that head. □

From the construction of the tree automata in Procedure 3.2, it is easily understood that inclusion hierarchy $L(M_0(q)) \subseteq L(M_1(q)) \subseteq \cdots$ holds for an arbitrary state $q \in \mathcal{Q}_0$. First we show that $L(M_k(t)) \subseteq W_*(t)$ for any subterm t of goal terms. The following technical Lemma 3.8 relates rewriting moves of tree automata to actual applications of term rewriting rules. This lemma means that if the modified tree automaton makes a rewriting move and accepts an input term, then the tree automaton also accepts the term which is not rewritten by the corresponding rewrite rule.

Lemma 3.8 *For a ground Σ-term s, and states $q, q' \in \mathcal{Q}_k$, if there is a sequence of moves $s \vdash_k^* q' \vdash_k q$ such that*

(i) *the last move $q' \vdash_k q$ is a rewriting move at the root occurrence, and*

(ii) *there is no rewriting move in $s \vdash_k^* q'$,*

then there is a Σ-term s' such that $s' \rightarrow_R s$ and $s' \vdash_{k-1}^* q$.

Proof. We first show that s is an instance of the right-hand side of a rewrite rule. Then the corresponding instance of the left-hand side of the rule is specified and is shown to be accepted by $M_{k-1}(q)$. Without loss of generality, assume that

(i) the last rewriting move is defined for $l \rightarrow r \in R$,

(ii) l has m variables $x_1, \ldots, x_m$ of sorts $S_1, \ldots, S_m$, respectively,

(iii) x_i has γ_i occurrences in l at $p_i^j \in \mathcal{O}cc(l)$ $(1 \leq j \leq \gamma_i)$, and

(iv) if the variable x_i occurs in r, then it is at $o_i \in \mathcal{O}cc(r)$ (r is linear by Condition 3.4).

Since the last move $q' \vdash_k q$ is a rewriting move of degree k, and since it is defined for the rule $l \rightarrow r$ at Step 4 of Procedure 3.2, there are states q_i^j $(1 \leq i \leq m, 1 \leq j \leq \gamma_i)$ in $\mathcal{Q}_{k-1}$ such that

$$l[p_i^j \leftarrow q_i^j \mid 1 \leq i \leq m, 1 \leq j \leq \gamma_i] \vdash_{k-1}^* q, \tag{3.12}$$

and

$$L(M_{k-1}(q_i^1)) \cap \cdots \cap L(M_{k-1}(q_i^{\gamma_i})) \cap L(M_{k-1}(\langle S_i \rangle)) \neq \emptyset. \tag{3.13}$$

Furthermore, $q' = \langle r\rho \rangle$ holds for the $\mathcal{Q}_k$-substitution $\rho = \{x_i \mapsto q_i \mid \text{for } i \text{ such that } x_i \text{ occurs in } r\}$ where q_i is defined as $q_i = \bigcup_{1 \leq j \leq \gamma_i} q_i^j \cup \{S_i\}$ by Step 4.

On the other hand, since there is no rewriting move in $s \vdash_k^* q'$ and $q' = \langle r\rho \rangle$, by Lemma 3.7 the sequence $s \vdash_k^* q'$ can be written as

$$\begin{aligned} s \ &\vdash_{k-1}^* \ s[o_i \leftarrow \ q_i \mid \text{for } i \text{ such that } x_i \text{ occurs in } r] \\ &\vdash_{k-1}^* \ q'. \end{aligned} \tag{3.14}$$

Let $\sigma = \{x_1 \mapsto u_1, \ldots, x_m \mapsto u_m\}$ be a ground Σ-substitution where $u_1, \ldots, u_m$ are defined as follows:

(i) If x_i occurs in r, then let $u_i = s/o_i$. By construction of tree automata and (3.14), we have $u_i \vdash_{k-1}^* q_i$, which does not contain any rewriting move. By Lemma 3.5 and the definition of q_i, $u_i \vdash_{k-1}^* q_i$ implies $u_i \vdash_{k-1}^* q_i^j$.

(ii) If x_i does not occur in r, then u_i is chosen to satisfy $u_i \in L(M_{k-1}(t_i^1)) \cap \cdots \cap L(M_{k-1}(t_i^{\gamma_i})) \cap L(M_{k-1}(\langle S_i \rangle))$. Such u_i exists by (3.13). Remark that $u_i \vdash_{k-1}^* q_i^j$ for $1 \leq j \leq \gamma_i$.

In either case (i) or (ii), we have

$$u_i \vdash_{k-1}^* q_i^j \qquad (1 \leq i \leq m, 1 \leq j \leq \gamma_i). \tag{3.15}$$

Let $s' = l\sigma$, then, by (3.12), (3.15) we have

$$\begin{aligned} s' &= l[p_i^j \leftarrow u_i \mid 1 \leq i \leq m, 1 \leq j \leq \gamma_i] \\ &\vdash_{k-1}^* l[p_i^j \leftarrow q_i^j \mid 1 \leq i \leq m, 1 \leq j \leq \gamma_i] \vdash_{k-1}^* q. \end{aligned}$$

That is, $s' \vdash_{k-1}^* q$. Since $s = r\sigma$, the relation $s' \rightarrow_R s$ holds. □

The previous Lemma 3.8 concerns only one rewriting at the root occurrence. By using the lemma inductively, we have the following soundness theorem.

Theorem 3.9 *For every subterm t of goal terms, $L(M_*(\langle t_\Omega \rangle)) \subseteq W_*(t)$ holds.*

Proof. It is sufficient to show that for a ground Σ-term s and a state q,

$$s \vdash_k^* q \Rightarrow \exists \text{ground } \Sigma\text{-term } u\colon u \rightarrow_R^* s, u \vdash_0^* q. \tag{3.16}$$

If (3.16) is valid, then taking $q = \langle t_\Omega \rangle$ and hence $s \in L(M_k(\langle t_\Omega \rangle))$, and by using Lemma 3.6, the theorem holds. In the following, we show (3.16) holds by induction on k. The claim holds obviously when $k = 0$. Assume that (3.16) holds for $k-1$ or less, and consider an accepting sequence $\alpha\colon s \vdash_k^* q$. The inductive part is shown by another induction on the number of rewriting moves of degree k in the sequence α.

If α contains no rewriting move of degree k, then every move in α is a valid move of $M_{k-1}(q)$. Thus $M_{k-1}(q)$ also accepts s and (3.16) holds by the inductive hypothesis for k.

Assume that (3.16) holds for every sequence $s \vdash_k^* q$ with $n-1$ or less rewriting moves of degree k, and consider the case that $\alpha\colon s \vdash_k^* q$ has n rewriting moves of degree k. Without loss of generality, the sequence α can be written as

$$s \vdash_k^* s[o \leftarrow q''] \ \vdash_k \tag{3.17}$$

$$s[o \leftarrow q'] \ \vdash_k^* \ q \tag{3.18}$$

where o is an occurrence in s, q', q'' are states in $\mathcal{Q}_k$, the move $s[o \leftarrow q''] \vdash_k s[o \leftarrow q']$ is a rewriting move of degree k, and there is no rewriting move of degree k in $s \vdash_k^* s[o \leftarrow q'']$. That is, o is the occurrence where the first rewriting move of degree k takes place. By construction of tree automata and (3.17), we have $s/o \vdash_k^* q''$. Since the sequence $s/o \vdash_k^* q''$ does not contain rewriting moves of degree k, by induction hypothesis, there is a term v such that $v \rightarrow_R^* s/o$ and $v \vdash_0^* q''$. Therefore

$$v \vdash_k^* q'' \vdash_k q' \tag{3.19}$$

where the last move is a rewriting move of degree k and there is no rewriting move in $v \vdash_k^* q''$. By applying Lemma 3.8 to (3.19), it follows that there is a term v' such that $v' \rightarrow_R v$ and $v' \vdash_{k-1}^* q'$. By (3.18) and $v' \vdash_{k-1}^* q'$, we have $s[o \leftarrow v'] \vdash_k^* s[o \leftarrow q'] \vdash_k^* q$. Remark that this accepting sequence $s[o \leftarrow v'] \vdash_k^* q$ has only $n-1$ rewriting moves of degree k, and hence there is a term u such that $u \rightarrow_R^* s[o \leftarrow v']$ and $u \vdash_0^* q$ by the induction hypothesis. Since $v' \rightarrow_R v \rightarrow_R^* s/o$, we have $u \rightarrow_R^* s[o \leftarrow v'] \rightarrow_R s[o \leftarrow v] \rightarrow_R^* s[o \leftarrow g/o] = s$, and (3.16) holds. □

The following theorem shows the completeness of the procedure.

Theorem 3.10 *For a Σ-term $s \in W_*(t)$, there is an integer k such that $s \in M_k(\langle t_\Omega \rangle)$.*

Proof. For a state $q \in \mathcal{Q}_0$, we prove that if $s' \rightarrow_R^* s$ and $s' \in L(M_0(q))$, then there is an integer k such that $s \vdash_k^* q$, which implies that $s \in L(M_k(q))$. The proof is shown by induction on the length of the derivation $s' \rightarrow_R^* s$. For the basis $s' = s$, the lemma holds obviously. If $s' \rightarrow_R^+ s$, then there is a term u such that $s' \rightarrow_R^* u \rightarrow_R s$. By induction hypothesis applied to $s' \rightarrow_R^* u$, we have an integer k' such that $u \vdash_{k'}^* q$. Moreover, since $u \rightarrow_R s$ there is a rewrite rule $l \rightarrow r \in R$, a Σ-substitution σ, and an occurrence $o \in \mathcal{O}cc(u)$ such that $u/o = l\sigma$ and $s/o = r\sigma$. By construction of tree automata, there is a state $q' \in \mathcal{Q}_{k'}$ such that $u = u[o \leftarrow l\sigma] \vdash_{k'}^* u[o \leftarrow q'] \vdash_{k'}^* q$ and we have

$$l\sigma \vdash_{k'}^* q'. \tag{3.20}$$

Now, let us show that $r\sigma \vdash_{k'+1}^* q'$. Without loss of generality, assume that l has m variables $x_1, \ldots, x_m$ of sorts $S_1, \ldots, S_m$, respectively, and the variable x_i has γ_i occurrences in l at $p_i^j \in \mathcal{O}cc(l)$. By construction of tree automata, there are states q_i^j for $1 \leq i \leq m$ and $1 \leq j \leq \gamma_i$ such that

$$x_i\sigma \vdash_1^* q_i^j \tag{3.21}$$

and

$$l[p_i^j \leftarrow q_i^j \mid 1 \leq i \leq m, 1 \leq j \leq \gamma_i] \vdash_1^* q'. \tag{3.22}$$

The sequence (3.21) means that $x_i\sigma \in L(M_*(q_i^1)) \cap \cdots \cap L(M_*(q_i^{\gamma_i}))$. In addition, since the substitution σ is well-sorted, the term $x_i\sigma$ has the sort S_i and thus

$$L(M_*(q_i^1)) \cap \cdots \cap L(M_*(q_i^{\gamma_i})) \cap L(M_*(\langle S \rangle)) \neq \emptyset \tag{3.23}$$

for $1 \leq i \leq m$. By (3.22), (3.23) and Step 4 of Procedure 3.2, a $\mathcal{Q}_{k'+1}$-substitution $\rho = \{x_1 \mapsto q_i \mid \text{for } i \text{ such that } x_i \text{ occurs in } r\}$ is defined. By Lemma 3.5, the co-domain of the $\mathcal{Q}_{k'+1}$-substitution ρ satisfy

$$L(M_{k'+1}(q_i^1)) \cap \cdots \cap L(M_{k'+1}(q_i^{\gamma_i})) \cap L(M_{k'+1}(\langle S \rangle)) = L(M_{k'+1}(q_i)) \tag{3.24}$$

for $1 \leq i \leq m$ and the transition rules are defined by the procedure **ADDTRANS** to satisfy that

$$r\rho \vdash_{k'+1}^* q'. \tag{3.25}$$

By (3.21) and (3.24), we have

$$x_i\sigma \vdash^*_{k'+1} q_i \qquad (1 \leq i \leq m). \tag{3.26}$$

Summarizing (3.25) and (3.26), we have $r\sigma \vdash^*_{k'+1} q'$, and the lemma holds. □

The following Theorem 3.11 summarizes the whole discussion in this section. The proof follows from Theorem 3.9 and Theorem 3.10.

Theorem 3.11 *Σ-term t_1 and t_2 are R-unifiable if and only if $L(M_*(t_1)) \cap L(M_*(t_2)) \neq \emptyset$.* □

Theorem 3.11 states that if goal terms are R-unifiable, then we can obtain a correct yes/no answer within finite steps. On the other hand, if goal terms are not unifiable, then the procedure does not always terminate. In [7], a sufficient condition under which the procedure in [7] terminates is provided. We conjecture that the condition is also sufficient for the termination of the procedure proposed in this paper.

Conjecture 3.12 *If a given unification problem satisfies Condition 3.1 to 3.4 and Condition 4.6 of [7], then the unification problem is decidable.*

4 Conclusion and Future Work

We have proposed a procedure for solving an order-sorted unification problem. The procedure solves the problem with the help of tree automata. We also have proved that the procedure is sound and complete.

The presented Procedure 3.2 returns yes/no answer only. However, the authors conjecture that if we analyze the tree automata constructed by Procedure 3.2 carefully, we can obtain unifiers of the goal terms also. Another direction of the future work is to extend tree automata to more powerful computational models such as in [2]. This will relax conditions on the instances (Conditions 3.1 through Condition 3.4) and make the procedure applicable to a larger class of TRS.

References

[1] Baader, F., Siekmann, J.H.: "Unification Theory," *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 2, Deduction Methodologies*, pp. 41–125, Oxford University Press, 1994.

[2] Bogaert, B., Tison, S.: "Equality and Disequality Constraints on Direct Subterms in Tree Automata," *Proc. of the Ninth Annual Symposium on Theoretical Aspect of Computer Science, LNCS* **577**, pp. 161–171, 1992.

[3] Comon, H.: "Equational Formulas in Order-Sorted Algebras," *Proc. of the Fourth Intl. Conf. on RTA, LNCS* **443**, pp. 674–688, 1990.

[4] Dershowitz, N., Jouannaud, J.-P.: "Rewrite Systems," *Handbook of Theoretical Computer Science, Vol. B, Formal Models and Semantics*, pp. 243–320, Elsevier Science Publishers, 1990.

[5] Gécseq, F., Steinby, M.: *Tree Automata*, Akadémiai Kiadó, 1984.

[6] Jouannaud, J.-P., Kirchner, C.: "Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification." *Computational Logic*, pp. 257–321, MIT-Press, 1991.

[7] Kaji, Y.,Fujiwara, T., Kasami, T.: "Solving a Unification Problem under Constrained Substitutions Using Tree Automata," *Journal of Symbolic Computation,* **23***, 1*, pp. 79–117, 1997.

[8] Schmidt-Schauß, M.: *Computational Aspects of an Order-Sorted Logic with Term Declarations, LNAI* **395**, 1989.

[9] Takada, K., Kaji, Y. and Kasami, T.: "Syntactic Unification Problems under Constrained Substitutions," *IEICE Trans. on Information and Systems*, E80-D, pp. 553–561, 1997.

[10] Takai, T., Kaji, Y. and Kasami, T.: "Solving an Order-Sorted Unification problem with Term Declarations Using Tree Automata," *Techical Report of IEICE,* COMP97-5, pp. 33–40, 1997.

[11] Waldmann, U.: "Semantics of Order-sorted Specifications," *Theoretical Computer Science* **94**, pp. 1–35, 1992.

[12] With, L.: "Completeness and Confluence of Order-Sorted Term Rewriting," *Proc. of 3rd CTRS, LNCS* **656**, pp. 393–407, 1992.