A sufficient condition for the termination of the procedure for solving an order-sorted unification problem

Toshinori Takai, Yuichi Kaji and Hiroyuki Seki

{toshin-t,kaji,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. The procedure requires an instance of the problem to satisfy that the TRS is right-linear and the goal terms are linear and share no variables. If a given instance of the problem satisfies these conditions and the procedure halts, then it answers correctly. In this paper, we propose a sufficient condition to terminate the procedure. The unification procedure constructs tree automata to solve the problem. The proposed condition guarantees the number of the states of the tree automata to be finite and provides a decidable subclass of the order-sorted unification problems.

1 Introduction

Unification problems are significant in theoretical computer science, especially, in automated deduction and term rewriting systems [1]. The unification problem is a problem to decide whether or not given two first-order terms (goal terms) can coincide by some substitution. If the goal terms coincide, then we say they are unifiable. If the semantics of terms is defined by a (confluent) TRS, then the problem is extended to the problem to decide whether goal terms can coincide by a substitution and rewriting. In this case the problem is generally undecidable. For example, consider two terms $s = f(x, a)$ and $t = g(b, y)$ and a TRS $R = \{f(x, a) \to g(x, a)\}$ where a is a constant, f, g are function symbols, and x, y are variables. If we ignore TRS R, then two terms s, t are not unifiable. On the other hand, if we take account of TRS R, then by substitution $\{x \mapsto b, y \mapsto a\}$ the goal terms become $f(b, a)$ and $g(b, a)$, and the former term $f(b, a)$ can be rewritten to the latter term. Hence, s and t are unifiable under TRS R.

One of the most known and useful classifications of terms is order-sorted structure, which can handle partially defined functions and subtypes [7]. An order-sorted term has a sort. For example, if we would like to represent the set of integers by two function symbols 0 (zero) and f (successor function) and explicitly distinguish odd numbers and even numbers, then we can define the sort of 0 is EVEN, the sort of $f(0)$ is EVEN, ... in the order-sorted framework. However, some difficult problems occur when we consider order-sorted term rewriting systems or order-sorted unification problems [9, 10].

In our previous paper [8] we have proposed a procedure for solving an order-sorted unification problem with a confluent TRS by using tree automata. The procedure requires an instance of the problem (goal terms and a TRS) to satisfy the following conditions [8]:

1. The TRS is right-linear.
2. The goal terms share no variables.
3. The goal terms are linear.

Under these conditions, the order-sorted unification procedure proposed in [8] is valid (answers correctly) if it halts. However, the procedure does not always halt when an instance of the problem is not unifiable. For the unsorted case, we proposed a sufficient condition to terminate the unification procedure in [5]. In this report, using a similar technique in [5] we provide a sufficient condition to terminate the order-sorted procedure proposed in [8].

The organization of the rest of this report is as follows. In Section 2, we briefly present the definitions of term rewriting system, order-sorted signature, tree automaton and related concepts. Next, we define the order-sorted unification problem in Section 3. In Section 4, we present a unification procedure to solve the problem. In Section 5, we propose a sufficient condition to terminate the unification procedure.

2 Preliminaries

2.1 TRSs and order-sorted signatures

We assume the readers are familiar with TRSs and the related definitions.

Let $\mathcal{F}$ be a finite set of *function symbols* and $\mathcal{X}$ be a countable set of *variables*. The *arity* of $f \in \mathcal{F}$ (the number of arguments 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* a variable x if $x \in \mathcal{V}ar(s) \cap \mathcal{V}ar(t)$. A *substitution* is a mapping from $\mathcal{X}$ to $\mathcal{T}_{\mathcal{F}}$ and written as $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where t_i with $1 \leq i \leq n$ is a term which substitutes for variable x_i. An application of a substitution is written in the postfix notation; $t\sigma$ is the term obtained from term t by applying a substitution σ to t. 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 $(\mathcal{S}, \mathcal{D}, \mathcal{F}, \mathcal{X}, \mathcal{P})$ where

- $\mathcal{S}$ is a finite set of *sort symbols*.
- $\mathcal{D}$ is a finite 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 finite set of *function symbols*.
- $\mathcal{X}$ is a set of *variables* which can be decomposed into pairwise disjoint countable 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 finite set of *function declarations* $f{:}\, S_1 \times \cdots \times S_n \to S$ 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 as follows:

(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 $\mathcal{S}ort_\Sigma(t) = \{S \in \mathcal{S} \mid t \in \mathcal{T}_{\Sigma,S}\}$. A signature Σ is called *regular* if, for every Σ-term t, $\mathcal{S}ort_\Sigma(t)$ has a unique smallest sort (with respect to $\sqsubseteq$). If S is the smallest among $\mathcal{S}ort_\Sigma(t)$, then we say that the sort of t is S, or t has 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.

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 empty sequence is denoted as λ. 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 $o_2 \preceq o_1$. The subterm of t at an occurrence o is denoted by t/o. Especially, $t/\lambda = t$. If $t/o = f(t_1, \ldots, t_n)$, then f is called the function symbol at o (in t). 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 condition is essential in defining a rewrite relation by Σ-rewrite rules and hence we include this condition 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 is 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 $t_2 \to_R^* v$.

2.2 Tree automata

A *tree automaton* [4] is a 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 $(\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{\mathit{final}}, \Delta)$ where $\mathcal{F}$ is a finite set of function symbols, $\mathcal{Q}$ is a finite set of states, $\mathcal{Q}_{\mathit{final}} \subseteq \mathcal{Q}$ 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) an ordinary rule $f(q_1, \ldots, q_n) \rightharpoonup q$ where $f \in \mathcal{F}$, $a(f) = n$, and $q_1, \ldots, q_n, q \in \mathcal{Q}$. Especially, 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 $q_f \in \mathcal{Q}_{\mathit{final}}$, then we say t is *accepted* by M.

The set of terms accepted by M is denoted by $L(M)$. A set T of terms 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 Order-sorted unification problem

In this section, we define an order-sorted unification problem which is considered in this paper.

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 $t_1\sigma \rightarrow_R^* v$ and $t_2\sigma \rightarrow_R^* v$. 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 which is, 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. Σ-terms t_1 and t_2 are called *goal terms.* (Σ, t_1, t_2, R) is called an *instance of the unification problem.* In this paper, we discuss this simple yes-no version of the unification problem.

For a term t, the set of all ground instances of t is denoted by $\mathcal{I}(t)$. For a set of ground terms L and a TRS R, let $R^*(L) = \{t \mid t' \rightarrow_R^* t, t' \in L\}$. A term in $R^*(L)$ is called an *R-descendant* of L. Consider the sets of ground R-instances of goal terms t_1 and t_2, which can be expressed as $R^*(\mathcal{I}(t_1))$ and $R^*(\mathcal{I}(t_2))$, respectively. We can use these sets to solve the unification problem effectively. The following lemma is proved in [5].

Lemma 3.1 *If goal terms t_1 and t_2 satisfy $\mathcal{V}ar(t_1) \cap \mathcal{V}ar(t_2) = \emptyset$ and a TRS R is confluent, then t_1 and t_2 are R-unifiable if and only if $R^*(\mathcal{I}(t_1)) \cap R^*(\mathcal{I}(t_2)) \neq \emptyset$.*□

It is undecidable whether $R^*(\mathcal{I}(t_1)) \cap R^*(\mathcal{I}(t_2)) \neq \emptyset$ or not, even if goal terms do not share any variable and a given TRS is confluent. However, for a class of instances such that $R^*(\mathcal{I}(t_1))$ and $R^*(\mathcal{I}(t_2))$ are both regular tree languages, the problem is decidable since the class of regular tree languages is closed under intersection and the emptiness is decidable for regular tree languages. In the next section, we provide a sufficient condition under which $R^*(\mathcal{I}(t_1))$ and $R^*(\mathcal{I}(t_2))$ are regular, and present a procedure to construct tree automata which accept $R^*(\mathcal{I}(t_1))$ and $R^*(\mathcal{I}(t_2))$ under the sufficient condition.

4 A procedure for solving the unification problem

In this section, we present the order-sorted R-unification procedure appeared in [5, 8]. For detailed explanation of the procedure, see [5, 8].

The procedure takes goal terms t_1, t_2 and a TRS R as inputs and consecutively constructs pairs of tree automata which accept $R^*(\mathcal{I}(t_1))$ and $R^*(\mathcal{I}(t_2))$ in the limit. We define packed Ω-terms and use them as states of tree automata.

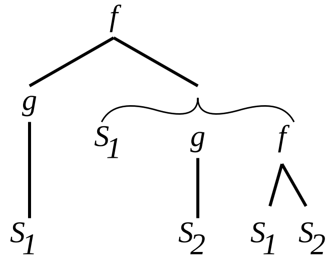

Figure 1: A packed Ω-term $\langle f(g(S_1), \langle S_1, g(S_2), f(S_1, S_2)\rangle)\rangle$.

Definition 4.1 Let $\Sigma = (\mathcal{S}, \mathcal{D}, \mathcal{F}, \mathcal{X}, \mathcal{P})$ be a signature. For a term $t \in \mathcal{T}_\Sigma$, let t_Ω be obtained from t by replacing each variable $x_S (S \in \mathcal{S})$ with $\mathcal{S}$. t_Ω is called an *Ω-term*. We define two sets $\mathcal{B}_{\mathcal{F},\Omega}$ and $\mathcal{P}_{\mathcal{F},\Omega}$ as the smallest sets which satisfies the following conditions.

- For a Ω-term t, $t \in \mathcal{B}_{\mathcal{F},\Omega}$.
- For $m \geq 1$ and $t_i \in \mathcal{B}_{\mathcal{F},\Omega}$ $(1 \leq i \leq m)$, $\langle t_1, \ldots, t_m \rangle \in \mathcal{P}_{\mathcal{F},\Omega}$.
- For $f \in F$ and $t_i \in \mathcal{P}_{\mathcal{F},\Omega}$ $(1 \leq i \leq a(f))$, $f(t_1, \ldots, t_{a(f)}) \in \mathcal{B}_{\mathcal{F},\Omega}$.

An element of $\mathcal{P}_{\mathcal{F},\Omega}$ is called a packed Ω-term (or packed term). Similarly, an element of $\mathcal{B}_{\mathcal{F},\Omega}$ is called a basic packed (Ω-)term. As defined below, a packed Ω-term is a finite representation of an infinite set of Ω-terms. For the readability, a packed term $\langle t \rangle$ is abbreviated as t. □

For example, let $\mathcal{F}$ be $\{f, g\}$ and Ω be $\{S_1, S_2\}$. We can easily verify that $f(g(S_1), S_2)$, $\langle f(g(S_1), S_2), g(\langle S_1, S_2, g(S_1)\rangle)\rangle$, and $g(\langle f(S_1, S_2), g(\langle g(S_1), g(S_2)\rangle)\rangle)$ belong to $\mathcal{P}_{\mathcal{F},\Omega}$.

Definition 4.2 The *extension* of a (basic) packed term q, denoted $E(q)$, is the set of Ω-terms defined as follows.

- If $q = t$ is 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 $\{f(g(S_1), S_2)\}$, $\{f(g(S_1), S_2), g(S_1), g(S_2), g(g(S_1))\}$, and $\{g(f(S_1, S_2)), g(g(g(S_1))), g(g(g(S_2)))\}$.

The unification procedure requires an instance (Σ, t_1, t_2, R) of the problem to satisfy the following Condition 4.1. The reason why the procedure presupposes Condition 4.1 has been mentioned in [5, 8].

Condition 4.1 Σ-TRS R is confluent and right-linear. Goal terms t_1 and t_2 are linear and $\mathcal{V}ar(t_1) \cap \mathcal{V}ar(t_2) = \emptyset$. □

If a given instance of the unification problem satisfies Condition 4.1, then the procedure correctly answers whether the goal terms can be unified or not if the procedure halts. In the next section, we investigate a sufficient condition to terminate the procedure. For a tree automaton $M = (\mathcal{F}, \mathcal{Q}, \mathcal{Q}_{final}, \Delta)$ and $q_f \in \mathcal{Q}_{final}$, we let $M(q_f) = (\mathcal{F}, \mathcal{Q}, \{q_f\}, \Delta)$.

Procedure 4.1 [UNIFICATION]

Input: an order-sorted signature Σ, goal terms t_1 and t_2 and a TRS R which satisfy Condition 4.1.
Output: *"unifiable"* or *"not unifiable"*.

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

Step 2. Construct a basic tree automaton $M_0 = (\mathcal{F}, \mathcal{Q}_0, \mathcal{Q}_{\mathit{final}}, \Delta_0)$ by executing **BASIS**(t) for the goal terms $t \in \{t_1, t_2\}$ (see Procedure 4.2).

Step 3. Let $\mathcal{Q}_{k+1} := \mathcal{Q}_k$, $\Delta_{k+1} := \Delta_k$ and let $M_k = (\mathcal{F}, \mathcal{Q}_k, \mathcal{Q}_{\mathit{final}}, \Delta)$. If $t \vdash t'$ (resp. $t \vdash^* t'$) in M_k, then we write $t \vdash_k t'$ (resp. $t \vdash_k^* t'$).

Step 4. Let $l \to r$ be a rewrite rule in R and assume that l has $m(\geq 0)$ variables $x_1, \ldots, x_m$ of sorts $S_1, \ldots, S_m$, respectively, and 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

$$l[p_i^j \leftarrow q_i^j \mid 1 \leq i \leq m, 1 \leq j \leq \gamma_i] \vdash_k^* q \tag{4.1}$$

and

$$L(M_k(q_i^1)) \cap \cdots \cap L(M_k(q_i^{\gamma_i})) \cap L(M_k(\langle S_i \rangle)) \neq \emptyset \tag{4.2}$$

for $1 \leq i \leq m$, then add

$$q_i = \bigcup_{1 \leq j \leq \gamma_i} q_i^j \cup \{S_i\} \quad (1 \leq i \leq m) \tag{4.3}$$

to $\mathcal{Q}_{k+1}$ as new states, let $\rho = \{x_i \mapsto q_i\}$ be a $\mathcal{Q}_{k+1}$-substitution, and do the following (a) and (b).

(a) Add the transition rule $\langle r\rho \rangle \rightharpoonup q$ to Δ_{k+1} where $\langle r\rho \rangle$ is a new state. 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 4.3). In **ADDTRANS**, new states $\langle r\rho/o \rangle \notin \mathcal{Q}_k (o \in \mathcal{O}cc(r))$ are introduced.

Simultaneously execute this Step 4 for every rewrite rule and every tuple of states that satisfy conditions (4.1) and (4.2).

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 $L(M_*(\langle t_{1\Omega} \rangle)) \cap L(M_*(\langle t_{2\Omega} \rangle)) \neq \emptyset$, and "*not unifiable*" otherwise.

□

Procedures **BASIS** and **ADDTRANS** used in Procedure 4.1 are defined as follows.

Procedure 4.2 [BASIS]

Input: an order-sorted signature $\Sigma = (\mathcal{S}, \mathcal{D}, \mathcal{F}, \mathcal{X}, \mathcal{P})$ and goal terms t_1, t_2 which satisfy Condition 4.1.
Output: a tree automaton M_0 such that $\mathcal{I}(t) = M_0(\langle t_\Omega \rangle)$ for each goal term t.

Step 1. For each sort $S \in \mathcal{S}$, define $\langle S \rangle$ as a state.

Step 2. 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.

Step 3. For each function declaration $f\colon S_1 \times \cdots \times S_n \to S$, let $\langle f(S_1, \ldots, S_n)\rangle$ be a new state and $f(\langle S_1\rangle, \ldots, \langle S_n\rangle) \rightharpoonup \langle f(S_1, \ldots, S_n)\rangle$ and $\langle f(S_1, \ldots, S_n)\rangle \rightharpoonup \langle S\rangle$ be new transition rules.

Step 4. For each goal term t, execute **ADDTRANS**($\langle t_\Omega\rangle$) shown below.

□

Procedure 4.3 [**ADDTRANS**] This procedure takes a packed Ω-term τ as an input. If the input τ has been already introduced as a state, then the procedure defines no transitions. If τ has not yet been a state, then the procedure first makes τ 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 $1 \le i \le n$.

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

□

Example 4.1 [8] Let Σ, R and G be an instance of the unification problem where

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

The tree automata M_0 is constructed as follows. $\mathcal{Q}_0 = \{\ \langle S_0\rangle, \langle S_1\rangle, \langle a\rangle, \langle c\rangle, \langle f(S_0, S_0)\rangle, \langle g(S_0)\rangle, \langle f(c, g(S_0))\rangle\ \}$ and the transition rules are:

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

Remark that

$$\begin{array}{rcl} L(M_0(\langle f(c, g(S_0))\rangle)) & = & \{f(c, g(t)) \mid t \text{ has sort } S_0\} \quad \text{and} \\ L(M_0(\langle c\rangle)) & = & \{c\} \end{array}$$

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

Next, assume Procedure 4.1 deals with rewrite rule $g(x_{S_1}) \rightarrow x_{S_1}$ in Step 4. 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 rule $f(x_{S_1}, x_{S_1}) \rightarrow g(x_{S_1})$ 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

$$c \in L(M_1(\langle c\rangle)) \cap L(M_1(\langle g(S_0)\rangle)) \cap L(M_1(\langle S_1\rangle))$$

holds. Note that $c \in L(M_1(\langle g(S_0)\rangle))$ since we add $\langle c\rangle \rightharpoonup \langle g(S_0)\rangle$ to Δ_1. By Step 4, new transition rules are added to Δ_1 in order to make the following sequence possible:

$$g(\langle c, g(S_0), S_1\rangle) \ \vdash_2^* \ \langle f(c, g(S_0))\rangle$$

where $\langle c, g(S_0), S_1\rangle$ is a new state such that

$$L(M_1(\langle c, g(S_0), S_1\rangle)) = L(M_1(\langle c\rangle)) \cap L(M_1(\langle g(S_0)\rangle)) \cap L(M_1(\langle S_1\rangle)).$$

Remark that $c \in L(M_1(\langle c, g(S_0), S_1\rangle))$. Again consider rule $g(x_{S_1}) \rightarrow x_{S_1}$, then the following sequence

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

is possible 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 $L(M_3(\langle f(c, g(S_0))\rangle))$. 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 $c \in L(M_3(\langle f(c, g(S_0))\rangle)) \cap L(M_3(\langle c\rangle))$. That is, the goal terms in G are unifiable. □

The correctness of Procedure 4.1 has been proved in [5, 8]. The next theorem plays a key role in the correctness proof.

Theorem 4.1 [5, 8] *For goal terms t_1, t_2 and a TRS R which satisfy Condition 4.1, there exists an integer k such that $L(M_k(\langle t_{1\Omega}\rangle)) = R^*(\mathcal{I}(t_1))$ and $L(M_k(\langle t_{2\Omega}\rangle)) = R^*(\mathcal{I}(t_2))$ if Procedure 4.1 halts.*□

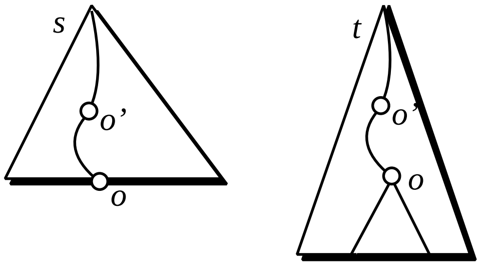

Figure 2: The path from the root to occurrence o

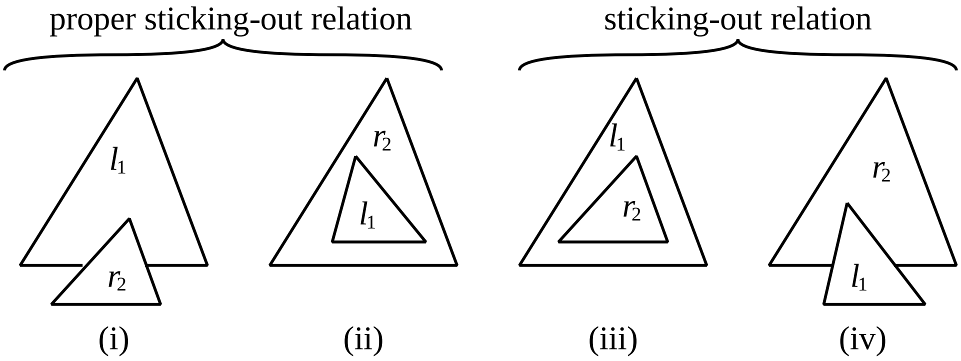

Figure 3: Rewrite rules and sticking-out relations.

5 A sufficient condition to terminate the procedure

In this section, we provide a sufficient condition which guarantees the termination of Procedure 4.1. The idea behind the sufficient condition is essentially the same as in [5]. However, the procedure in this paper allows a TRS to have an arbitrary right-linear rewrite rule while the procedure in [5] allows only simple right-linear rewrite rules [6].

We need some auxiliary definitions to describe the sufficient condition (Condition 5.1).

Definition 5.1 A term t *sticks out of a term* s if there is a variable occurrence o of s such that, for every occurrence o' with $\lambda \preceq o' \prec o$, the function symbol of s at o' is the same as the function symbol of t at o' and t/o is not a ground term. When we want to emphasize the occurrence o, we say t sticks out of s *at occurrence* o. If t/o is not a variable, then we say that t *properly* sticks out of s (at o). If t (properly) sticks out of s, then we say that s is (properly) *short of* t.

Example 5.1 Let $t_1 = f(g(x), g(y)), t_2 = f(g(x), c)$ and $t_3 = f(g(g(x)), c)$. The term t_2 sticks out of t_1 at the occurrence $1 \cdot 1$ since the function symbols of t_1 at the occurrences λ and 1 are the same as those of t_2. Similarly, t_3 properly sticks out of t_1 at $1 \cdot 1$. □

Definition 5.2 For a TRS R, define the directed graph G_R whose edges have weights. A vertex of G_R is a rewrite rule in R. The edges of G_R and their weights are defined as follows. Consider any pair of rules $l_1 \to r_1$ and $l_2 \to r_2$ (two rules can be the same).

(i) If r_2 properly sticks out of a subterm of l_1 (Fig. 3(i)), then G_R has an edge from $l_2 \to r_2$ to $l_1 \to r_1$ with weight 1.

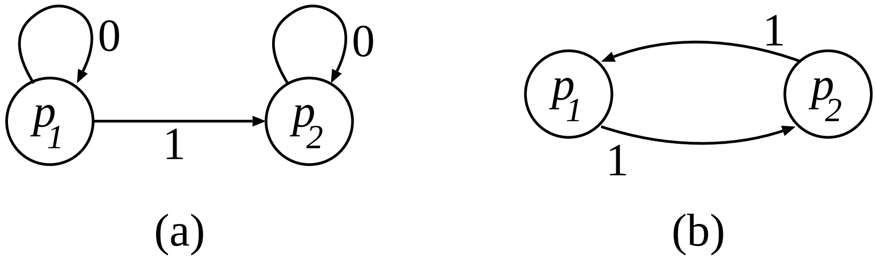

Figure 4: Examples of sticking-out graphs.

(ii) If a subterm of r_2 properly sticks out of l_1 (Fig. 3(ii)), then G_R has an edge from $l_2 \to r_2$ to $l_1 \to r_1$ with weight 1.

(iii) If r_2 is short of a subterm of l_1 (Fig. 3(iii)), then G_R has an edge from $l_2 \to r_2$ to $l_1 \to r_1$ with weight 0.

(iv) If a subterm of r_2 is short of l_1 (Fig. 3(iv)), then G_R has an edge from $l_2 \to r_2$ to $l_1 \to r_1$ with weight 0.

The graph G_R is called the *sticking-out graph of* R.□

By using a sticking-out graph, a sufficient condition to terminate Procedure 4.1 is described as follows.

Condition 5.1 The sticking-out graph G_R of R has no cycle with positive weight where the weight of a cycle is the sum of the weights of edges that constitute the cycle.□

An intuitive meaning of this condition will appear in the proof of the termination of Procedure 4.1 in the next section (Theorem 6.2).

Example 5.2 Let R be a TRS that has two rewriting rules p_1: $g(x) \to f(g(x), b)$ and p_2 : $f(x, a) \to f(a, x)$ where p_1 and p_2 are the labels of the rules. For simplicity, the left- and the right-hand sides of p_i $(i = 1, 2)$ are referred to as l_i and r_i, respectively.

The sticking-out graph G_R has two vertices p_1 and p_2 and the edges shown in Fig. 4 (a). For example, the subterm $g(x)$ of r_1 is short of l_1 and hence p_1 has a simple cycle with weight zero. One the other hand, r_1 properly sticks out of l_2 at the occurrence 1, and thus there is an edge from p_1 to p_2 with weight 1. Procedure 4.1 halts for this rewrite system R since there is no cycle with positive weight. □

Example 5.3 Let R be a TRS that has two rewrite rules p_1: $f(x) \to g(g(x))$ and p_2: $g(x) \to f(f(x))$. As in the previous example, the left- and right-hand sides of p_i $(i = 1, 2)$ are referred to as l_i and r_i, respectively. The sticking-out graph for R is shown in Fig. 4 (b). Since r_1 and r_2 properly stick out of l_2 and l_1, respectively, the edges with weight 1 are defined from p_1 to p_2 and p_2 to p_1. Remark that there is a cycle with weight 2. Actually, Procedure 4.1 does not halt for this R. □

6 Proof of the termination of Procedure 4.1

At first, we define *the number of layers in a packed term (state).* If there exists an upper-bound of the number of layers in a state, then there exists an upper-bound of the number of states

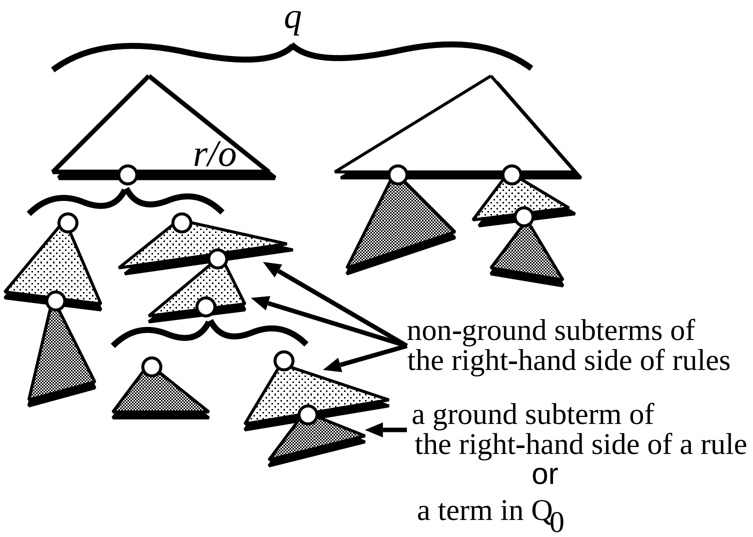

Figure 5: The structure in a new state.

(Lemma 6.1). If there exists an upper-bound of the number of states, then Procedure 4.1 halts. Therefore, we will provide a sufficient condition to upper-bound the number of layers in a state as a sufficient condition to terminate Procedure 4.1. The number of layers in a state q (denoted layer(q)) is defined as follows. Those states which belong to $\mathcal{Q}_0$ are defined to have only one layer. That is, layer(q) $= 1$ ($q \in \mathcal{Q}_0$). If q_i defined in Step 4 (4.3) is a new state, then layer(q_i) $= \max\{\text{layer}(q_i^j) \mid 1 \leq j \leq \gamma_i\}$. Similarly, the number of layers in a new state which is introduced in Step 3 of Procedure 4.3 is defined as the maximum of the number of layers in a state which constitutes the new state.

Next, consider the state $\langle r\rho/o\rangle$ which is introduced in Step 4 (a) or (b) of Procedure 4.1. Let $j = \max\{\text{layer}(q_i) \mid \text{for } i \text{ such that } x_i \text{ occurs in } r/o\}$. If r/o is ground, then let $j = 0$. When the state $\langle r\rho/o\rangle$ is added to $\mathcal{Q}_{k+1}$ in Step 4 (a) or (b) of Procedure 4.1, the number of layers in $\langle r\rho/o\rangle$ is associated as follows.

1. If $\langle r\rho/o\rangle$ has not yet belonged to $\mathcal{Q}_{k+1}$, then $\langle r\rho/o\rangle$ is defined to have $j+1$ layers.
2. If $\langle r\rho/o\rangle$ has already belonged to $\mathcal{Q}_{k+1}$ and is defined to have $j+1$ or more layers, then the number of layers in $\langle r\rho/o\rangle$ is decreased to $j+1$.
3. If $\langle r\rho/o\rangle$ has already belonged to $\mathcal{Q}_{k+1}$ and is defined to have j or less layers, then the number of layers in $\langle r\rho/o\rangle$ is not changed.

In general, there are many combinations of layers which constitute $\langle r\rho/o\rangle$. The number of layers in $\langle r\rho/o\rangle$ is defined to be the minimum among them. Remark that for any new state $\langle r\rho/o\rangle$, the number of layers in $\langle r\rho/o\rangle$ cannot be smaller than the number of layers in $\langle r\rho/oo'\rangle$ for any $o' \in \mathcal{O}cc(r\rho/o)$.

In the case of a non-packed term, it is trivial that, for a given number c, the set $\{t \mid \text{layer}(t) \leq c\}$ is finite. On the other hand, in the case of a packed term q, for a given c, $\{q \mid \text{layer}(q) \leq c\}$ is not always finite since there is no upper-bound of n for a packed term $\langle q_1, \ldots, q_n\rangle$. In other words, a packed term in $\{q \mid \text{layer}(q) \leq c\}$ does not have an upper-bound in width besides it has an upper-bound in depth. However, we have the following lemma.

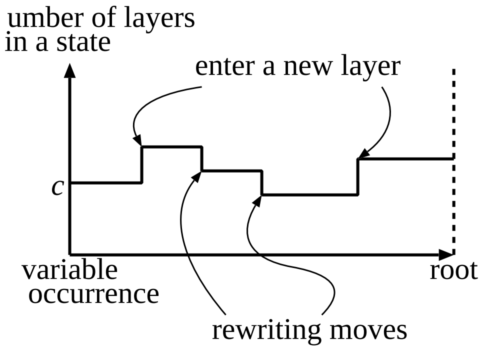

Figure 6: The number of layers in a state varies as the tree automaton moves.

Lemma 6.1 *Suppose that for an instance of the unification problem there exists a number c such that* $\text{layer}(q) \leq c$ *for any k and $q \in \mathcal{Q}_k$. Then there exists a number c' such that the number of the states in $\mathcal{Q}_k$ is not greater than c' for any k.*

Proof. There are four cases when a new state is added in Procedure 4.1.

1. In Step 4 of Procedure 4.1, a state which is defined as $q_i = \bigcup_{1\leq j\leq \gamma_i} q_i^j \cup \{S_i\}$ in (4.3) is added.
2. In Step 4 (a) or (b) of Procedure 4.1, a new state $\langle r\rho/o\rangle$ is added. In this case, the number of layers in a state may increases according to the definition of the number of layers in a state.
3. In Case 3 (i) of Procedure 4.3, a new state $\tau'' = \tau' \cup \tau \setminus \langle t_i\rangle$ is added.
4. In Case 3 (ii) of Procedure 4.3, a new state $\tau_j = \tau_{1j} \cup \cdots \cup \tau_{nj}$ is added.

Assume that there exists a number c such that $\text{layer}(q) \leq c$ for any k and $q \in \mathcal{Q}_k$. Then there exists a number k' such that case 2 does not take place at any loop counter k'' for $k'' \geq k'$ in Procedure 4.1. A new state which is added as in case 1, 3, or 4 is a combination of states in $\mathcal{Q}_{k'}$. Since $\mathcal{Q}_{k'}$ is finite, the number of combinations of members in $\mathcal{Q}_{k'}$ is also finite. Hence the lemma holds. □

Using this lemma, we can prove the termination of the unification procedure by showing that there is an upper-bound of the number of layers in a state if a given instance satisfies Condition 5.1.

We first observe how the number of layers in a state increases as the tree automaton moves as (4.1). For a term s, consider the path from a variable occurrence of s to the root of s, and observe how the number of layers in a state varies as the head of the tree automaton goes up the path (Fig. 6). When the tree automaton moves its head, a move defined by **ADDTRANS** is used. Thereby, the number of layers in the attached state is not changed, or increased by one if the head enters a new layer. On the other hand, when the tree automaton makes a rewriting move, the number of layers in the attached state does not increase.

For the later discussion, it is worthwhile to introduce some new notions concerning the behavior of a tree automaton. Let s be an input term and α be a sequence of moves of the tree automaton for s. An occurrence o of s is called an *ε-move occurrence in α* if an ε-move takes place at o, that is, if the sequence α contains a move $I_1 \vdash I_2$ such that $I_1/o = q_1$ and $I_2/o = q_2$ where q_1 and q_2 are states of the tree automaton. An occurrence $o_1 \in \mathcal{O}cc(s)$ is

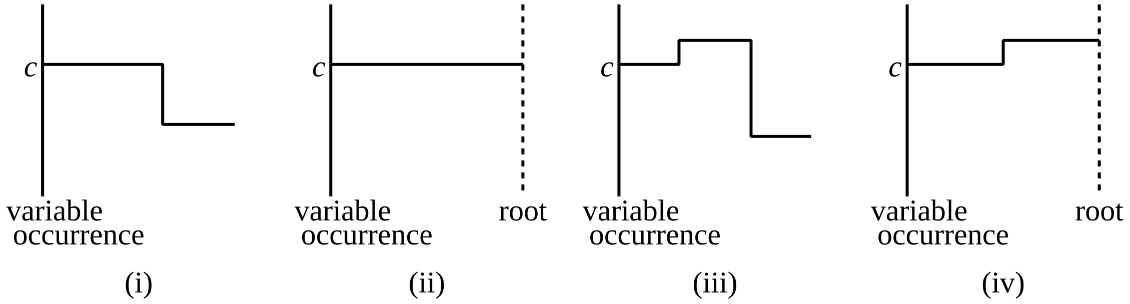

Figure 7: Four cases in which the number of layers in an associated state was c.

the *first ε-move occurrence with respect to a variable occurrence* o_2 if $o_1 \prec o_2$, o_1 is an ε-move occurrence and there is no ε-move occurrence o' such that $o_1 \prec o' \preceq o_2$.

Consider the sequence of moves (4.1) of a tree automaton in Step 4 of Procedure 4.1, and assume that the number of layers in state q_i^j is c. In this case, a newly introduced state in Step 4 of Procedure 4.1 may have $c+1$ layers. There are four different cases in which this situation occurs.

Case I: The head does not enter a new layer before the first rewriting move (Fig. 7(i)).

Case II: The head does not enter a new layer and there is no rewriting move (Fig. 7(ii)).

Case III: The head enters a new layer before the first rewriting move (Fig. 7(iii)).

Case IV: The head enters a new layer and there is no rewriting move (Fig. 7(iv)).

If the tree automaton M_k behaves as Case I or Case II and if c was the maximum number of layers in a state in $\mathcal{Q}_k$, then the newly introduced state will have larger number of layers than any other states which have belonged to $\mathcal{Q}_k$. In other words, if the tree automaton behaves as Case I or Case II, then the maximum number of layers in a state possibly increases. Condition 5.1 is to prohibit that the number of layers infinitively increases. To show that, we associate each state in $\mathcal{Q}_k$ with a nonnegative integer called the *rank* which is defined based on the sticking-out graph G_R. Then it is shown that an Ω-term with rank j has at most $j+1$ layers. First we associate each rule in R with an integer which is also called the *rank* as follows.

Step 1. The *rank* of every rule is initialized to one.

Step 2. For every rewrite rule v in R (i.e. vertex of G_R), the rank of v is changed to be the maximum of $\{$ the rank of $v' + w(v',v) \mid v'$ is a vertex of G_R from which there is an edge to v with weight $w(v',v)\}$.

Step 3. Repeat Step 2 until the rank of any rule is not changed.

In Example 5.2, the rule p_1 has rank 1 and the rule p_2 has rank 2. If R satisfies Condition 5.1, then it can be easily shown that each rule in R is associated with a unique rank which is not greater than the number of rules in R. A state is said to have *rank j* if it is introduced in Step 4 of Procedure 4.1 by using a rewrite rule with rank j. Those states that have belonged to $\mathcal{Q}_0$ is defined to have rank zero.

Consider that a state $\langle t \rangle$ is introduced in Step 4(a) of Procedure 4.1 by using a rewrite rule $l \to r$ with rank j and a substitution ρ. In the following Theorem 6.2, we show that every state in the co-domain of ρ has j or less layers. This implies that there are $j+1$ or less layers in $\langle t \rangle$.

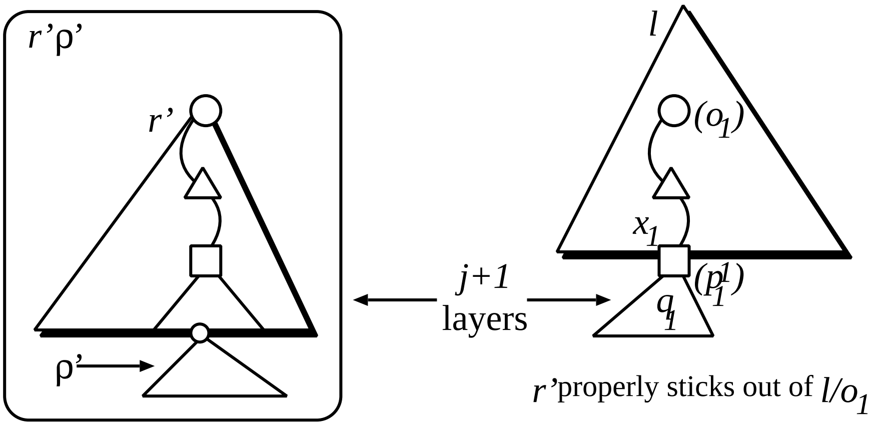

Figure 8: The number of layers and the sticking-out relation (Case I).

Theorem 6.2 *Let $l \to r$ and $\rho = \{x_i \mapsto q_i \mid$ for i such that x_i occurs in $r\}$ be respectively a rewrite rule and a $\mathcal{Q}_k$-substitution that are used in Step 4 of Procedure 4.1. If the rank of $l \to r$ is j, then every q_i $(1 \leq i \leq m)$ in the co-domain of ρ has j or less layers.*

Proof. The theorem is shown by induction on the value of loop variable k of Procedure 4.1. When $k = 0$, every state belongs to $\mathcal{Q}_0$ and hence it has only one layer, thus the theorem holds. Assume that the theorem holds for $k \leq n-1$, and consider the case that $k = n$. The inductive part is shown by contradiction. Assume that the co-domain of ρ contains a state which has $j+1$ layers. Without loss of generality, let q_1 be a state with $j+1$ layers. As we have observed in this section, there are four different cases under which this situation occurs.

Assume that the number of layers in the state varies as Case I. In this case the number of layers in the associated state is $j+1$ or more and is not changed before the number of layers is decreased by the first rewriting move. The overview of the discussion in this case is as follows: We first show that there is a rewrite rule $l' \to r'$ which corresponds to the first rewriting move, and also show that the rank of the rewrite rule is j or more. By using the property that there is no rewriting move before the first rewriting move, r' is shown to properly stick out of a subterm of l, which implies that the rank of $l \to r$ must be defined to be $j+1$ or more by (i) of the definition of the sticking-out graph, a contradiction. See also Fig. 8.

Let o_1 be the first ε-move occurrence and let q be the state just before the first rewriting move. Since the number of layers in the state is $j+1$ or more before the first rewriting move, q has $j+1$ or more layers. It cannot happen that q has belonged to $\mathcal{Q}_0$ since it implies that q has only one layer. Thereby there are rewrite rule, say $l' \to r'$ and $\mathcal{Q}_k$-substitution ρ' which were used to introduce q in Step 4, and then, q can be written as $\langle r'\rho' \rangle$. Furthermore, the rank of the rule $l' \to r'$ must be j or more, otherwise $\langle r'\rho' \rangle$ cannot have $j+1$ or more layers by the inductive hypothesis. In the following, we show that r' properly sticks out of l/o_1.

Consider the moves of the tree automaton from the occurrence p_1^1 to o_1. Since o_1 is the first ε-move occurrence, all moves under o_1 are defined by **ADDTRANS**. By the construction of transition rules in **ADDTRANS**, it follows that the function symbol of l at the occurrence $o_1 \cdot o'$ is the same as the function symbol of r' at o' for every o' such that $o_1 \cdot o' \prec p_1^1$. Furthermore, it can be easily shown that when the head visits the occurrence $o_1 \cdot o'(o_1 \cdot o' \preceq p_1^1)$ of l, the state $\langle r'\rho'/o' \rangle$ is attached to that head. Thereby, at the variable occurrence p_1^1, $stater'\rho'/o'$ was attached where o' is such that $o_1 \cdot o' = p_1^1$, and this is the state q_1. Intuitively saying, the head goes up l along the path from p_1^1 to o_1 by changing the state from $\langle q' \rangle$ to $\langle f(\ldots, g', \ldots) \rangle$ where f is the scanned symbol. Since the top layer of $\langle r'\rho' \rangle$ is r', this implies that r' properly sticks out of l/o_1. We have observed that the rank of $l' \to r'$ is j or more, and thus the rank of $l \to r$ must be defined to be $j+1$ or more by (i) of the definition of the sticking-out graph, a contradiction.

For other Cases II through IV, we can derive a contradiction in a similar way. Thereby, it cannot happen that q_1 has $j+1$ or more layers and the induction completes. □

Theorem 6.2 shows that if R satisfies Condition 5.1, then Procedure 4.1 halts.

Corollary 6.3 *Procedure 4.1 halts for instances of the order-sorted unification problem which satisfy Condition 5.1 in addition to Condition 4.1.*

Proof. By Theorem 6.2, if a given instance of the problem satisfies Condition 5.1, then every state in $\mathcal{Q}_k$ has $n+1$ or less layers where n is the number of rewrite rules in R. Using Lemma 6.1 with the finiteness of the number of layers in a state, we can say that there is an integer k_1 such that $\mathcal{Q}_0 \subseteq \cdots \subseteq \mathcal{Q}_{k_1} = \mathcal{Q}_{k_1+1} = \cdots$. Once the set of states has been fixed, then Procedure 4.1 only adds ε-transitions Δ_k with $k \geq k_1$ which saturates at last and hence the procedure halts under the condition. □

The above corollary provides a sufficient condition for the order-sorted unification problem to be decidable.

Corollary 6.4 *A class of order-sorted unification problems which satisfy Condition 4.1 and Condition 5.1 is decidable.*

Proof. If a given instance of the order-sorted unification problem satisfies Condition 4.1 and Condition 5.1, then Procedure 4.1 halts for the instance by Corollary 6.3 and answers correctly by Theorem 4.1. □

7 Conclusion

In this paper, we have proposed a sufficient condition for order-sorted unification problems to be decidable. We have proved that for a given instance of the unification problems which satisfies the condition the presented unification procedure halts and answers correctly.

The presented Procedure 4.1 returns yes/no answer only. However, the authors conjecture that if we analyze the tree automata constructed by Procedure 4.1 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 4.1 and Condition 5.1) 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] Gécseq, F., Steinby, M.: *Tree Automata*, Akadémiai Kiadó, 1984.

[5] Kaji, Y., Fujiwara, T., Kasami, T.: "Solving a unification problem under constrained substitutions using tree automata," *J. of Symbolic Computation,* **23**, *1*, pp. 79–117, 1997.

[6] Ohta, Y., Oyamaguchi, M., Toyama, Y.: "On the Church-Rosser Property of Simple-Right-Linear TRS's," *Transactions of IEICE,* **D-I**, *Vol. J78-D-I, No. 3*, pp. 263–268, 1995 (in Japanese).

[7] Smolka, G., Nutt, W., Goguen, J.A., and Meseguer, J.: "Order-sorted equational computation," *Resolution of equations in algebraic structure, Vol. 2*, pp. 297–367, Academic Press, 1989.

[8] Takai, T., Kaji, Y., Tanaka, T., Seki, H.: "A Procedure for Solving an Order-Sorted Unification Problem — extension for left nonlinear system," *Technical Report of IEICE*, **COMP98-44**, 1998.

[9] Werner, A.: "A Semantic Approach to Order-Sorted Rewriting," *LNCS* **690**, pp. 47–61, 1993.

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

APPENDIX: Proof of Theorem 6.2

We have observed in Section 6 that the number of layers in a state cannot varies as in Case I. In this appendix, we see that similar discussions hold for other cases and the theorem holds.

Case II

In this case, the initial state of the tree automaton has $j+1$ or layers, and the number of layers is not changed at all. That is, the last state which associated at the root occurrence has the same number of layers in the initial state. The overview of the discussion in this case is as follows: We first show that there is a rewrite rule $l' \to r'$ such that the last state is written as an instance of a subterm of r', and also show that the rank of the rewrite rule is j or more. By using property that there is no rewriting move, the subterm of r' is shown to properly stick out of l, which implies that the rank of $l \to r$ must be defined to be $j+1$ or more by (ii) of the definition of the sticking-out graph, a contradiction. If one replaces r' and o_1 in Fig. 8 with r'/o_2 and the root occurrence λ, respectively, then the figure sketches the following discussion.

Let q be the last state which is used in the sequence (4.1) of Step 4 of Procedure 4.1 at the root occurrence of l. Since the number of layers in the last state is $j+1$ or more, q has $j+1$ or more layers. It cannot happen that q has belonged to $\mathcal{Q}_0$ since it implies that q has only one layer. Thereby there are rewrite rule, say $l' \to r'$ and an occurrence o_2 such that q can be written as $\langle r'\rho'/o_2 \rangle$ where ρ' is $\mathcal{Q}_k$-substitution which is introduced when $l' \to r'$ is used at Step 4 of Procedure 4.1. Furthermore, the rank of the rule $l' \to r'$ must be j or more and r'/o_2 is a non-ground subterm of r' with size more than one, otherwise $\langle r'\rho'/o_2 \rangle$ cannot have $j+1$ or more layers by the inductive hypothesis. In the following, we show that r'/o_2 properly sticks out of l.

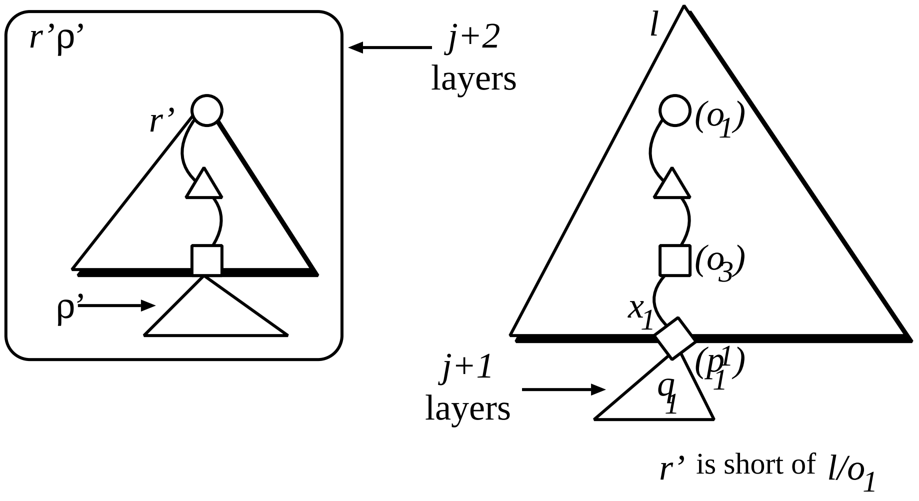

Figure 9: The number of layers and the sticking-out relation (Case III).

In this case, all moves at occurrences between the root and p_1^1 are defined by **ADDTRANS**. Similarly to the Case I, it follows that the function symbol of l at the occurrence is the same as the function symbol of r'/o_2 at o' for every o' such that $o' \prec p_1^1$. Furthermore, it can be easily shown that when the head visits occurrence o' ($o' \preceq p_1^1$) of l, the state $\langle r'\rho'/o_2 \cdot o' \rangle$ is attached to that head. Thereby, at the variable occurrence p_1^1, $q_1 = \langle r'\rho'/o_2 \cdot p_1^1 \rangle$ was attached. Intuitively saying, the head goes up l along the path from p_1^1 to the root by changing the state from $\langle q' \rangle$ to $\langle f(\ldots, g', \ldots) \rangle$ where f is the scanned symbol. Since the top layer of $\langle r'\rho'/o_2 \rangle$ is r'/o_2, this implies that r'/o_2 properly sticks out of l. We have observed that the rank of $l' \to r'$ is j or more, and thus the rank of $l \to r$ must be defined to be $j+1$ or more by (ii) of the definition of the sticking-out graph, a contradiction.

Case III

In this case, the number of layers in the associated state is increased to $j+2$ ore more by entering a new layer, and the number of layers is decreased by the first rewriting move. The overview of the discussion in this case is as follows: We first show that there is a rewrite rule $l' \to r'$ which corresponds to the first rewriting move, and also show that the rank of the rewrite rule is $j+1$ or more. By using the property that the number of layers is increased before the first rewriting move, r' is shown to be short of l, which implies that the rank of $l \to r$ must be defined to be $j+1$ or more by (iii) of the definition of the sticking-out graph, a contradiction. See Fig 9.

Let o_1 be the first ε-move occurrence and let $\langle r'\rho' \rangle$ be the state just before the first rewriting move where $l' \to r'$ and ρ' are the rewrite rule and the $\mathcal{Q}_k$-substitution which were used to introduce $\langle r'\rho' \rangle$ at Step 4. Since the number of layers is $j+2$ or more just before the rewriting move, $\langle r'\rho' \rangle$ has $j+2$ or more layers. The rank of the rule $l' \to r'$ must be $j+1$ or more, otherwise $\langle r'\rho' \rangle$ cannot have $j+2$ or more layers by the inductive hypothesis. In the following, we show that r' is short of l/o_1.

Consider the moves of the tree automaton from the occurrence p_1^1 to o_1. Let o_3 be the occurrence where the number of layers is increased first time before the first rewriting move. Since o_1 is the first ε-move occurrence, all moves under o_1 are defined by **ADDTRANS**. Similarly to the previous cases, it follows that the function symbol of l at the occurrence $o_1 \cdot o'$

is the same as the function symbol of r' at o' for every o' such that $o_1 \cdot o' \prec o_3$. Furthermore, since the number of layers is increased at o_3, the head enters the top layer of $\langle r'\rho' \rangle$. This means that o' is a variable occurrence of the top layer of $\langle r'\rho' \rangle$. Intuitively saying, the head goes up l along the path from p_1^1 to o_1 by changing the state from $\langle q' \rangle$ to $\langle f(\ldots, g', \ldots) \rangle$ where f is the scanned symbol. When the head reaches o_3, the head enters the top layer of $\langle r'\rho' \rangle$. Since the top layer of $\langle r'\rho' \rangle$ is r', this implies that r' is short of l/o_1. We have observed that the rank of $l' \to r'$ is $j+1$ or more, and thus the rank of $l \to r$ must be defined to be $j+1$ or more by (iii) of the definition of the sticking-out graph, a contradiction.

Case IV

In this case, the last state of the tree automaton at the root occurrence has $j+2$ or more layers, and the number of layers has been increased. The overview of the discussion in this case is as follows: We first show that there is a rewrite rule $l' \to r'$ such that the last state at the root occurrence is written as an instance of a subterm of r', and also show that the rank of the rewrite rule is $j+1$ or more. By using the property that the number of layers have been increased, the subterm of r' is shown to be short of l, which implies that the rank of $l \to r$ must be defined to be $j+1$ or more by (iv) of the definition of the sticking-out graph, a contradiction.

Let q be the last state in the sequence (4.1) of Step 4 of Procedure 4.1 at the root occurrence of l. Since the number of layers in the final state is $j+2$ or more, q has $j+2$ or more layers. It cannot happen that q has belonged to $\mathcal{Q}_0$ since it implies that q has only one layer. Thereby there are rewrite rule, say $l' \to r'$ and an occurrence o_2 such that q can be written as $\langle r'\rho'/o_2 \rangle$ where ρ' is $\mathcal{Q}_k$-substitution which is introduced when $l' \to r'$ is used at Step 4 of Procedure 4.1. Furthermore, the rank of the rule $l' \to r'$ must be $j+1$ or more and r'/o_2 is a non-ground subterm of r' with size more than one, otherwise $\langle r'\rho'/o_2 \rangle$ cannot have $j+2$ or more layers by the inductive hypothesis. In the following, we show that r'/o_2 is short of l.

In this case, all moves at occurrences between the root and p_1^1 are defined by **ADDTRANS**. Let o_3 be the occurrence where the number of layers is increased first time. Similarly to the previous case, it follows that the function symbol of l at the occurrence o' is the same as the function symbol of r'/o_2 at o' for every o' such that $o' \prec o_3$. Furthermore, since the number of layers is increased at o_3, the head of the tree automaton enters the top layer of $\langle r'\rho'/o_2 \rangle$ when the head reaches o_3. This means that o' is a variable occurrence of the top layer of $\langle r'\rho'/o_2 \rangle$. Intuitively saying, the head goes up l along the path from p_1^1 to the root by changing the state from $\langle q' \rangle$ to $\langle f(\ldots, g', \ldots) \rangle$ where f is the scanned symbol. When the head reaches o_3, the head enters the top layer of $\langle r'\rho'/o_2 \rangle$. Since the top layer of $\langle r'\rho'/o_2 \rangle$ is r'/o_2, this implies that r'/o_2 is short of l/o_1. We have observed that the rank of $l' \to r'$ is $j+1$ or more, and thus the rank of $l \to r$ must be defined to be $j+1$ or more by (iv) of the definition of the sticking-out graph, a contradiction.