INFORMATION AND COMPUTATION 88, 187–201 (1990)

Decidability of the Confluence of Finite Ground Term Rewrite Systems and of Other Related Term Rewrite Systems

MAX DAUCHET*

Laboratoire d'Informatique Fondamentale de Lille, UA 369 du CNRS, Université de Lille-Flandres-Artois, 59655 Villeneuve D'Ascq, France

THIERRY HEUILLARD†

Laboratoire de Recherche en Informatique, Bat. 490, Université de Paris-Sud, 91405 Orsay, France

PIERRE LESCANNE‡

Centre de Recherche en Informatique de Nancy, CNRS UA 262 and LORIA, Campus Scientifique, BP 239, 54506 Vandoeuvre-les-Nancy, France

AND

SOPHIE TISON*

Laboratoire d'Informatique Fondamentale de Lille, UA 369 du CNRS, Université de Lille-Flandres-Artois, 59655 Villeneuve d'Ascq, France

The aim of this paper is to propose an algorithm to decide the confluence of finite ground term rewrite systems. Actually a more general class of possibly infinite ground term rewrite systems is studied. It is well known that the confluence is not decidable for general term rewrite systems, but this paper proves it is for ground term rewrite systems following a conjecture made by Huet and Oppen in their survey. The result is also applied to the confluence of left-linear and right-ground term rewrite systems. We also sketch an algorithm for checking this property. This algorithm is based on tree automata and tree transducers. Here, we regard them as rewrite systems and specialists in automata theory would translate that easily in their language.

* Supported by Greco Mathématique et Informatique.
† This work was done as a student at the Ecole des Mines de Nancy.
‡ Supported by Greco Programmation.

0890-5401/90 $3.00

1. INTRODUCTION

The aim of this paper is to propose a simple algorithm to decide the confluence of ground term rewrite systems. This algorithm is derived from decidability results presented in (Dauchet and Tison, 1985). Here a simplified view of the problem and its solution is given. Let us recall that a ground term rewrite system is a term rewrite system where each rule is a pair of ground terms, i.e., a pair of terms without variables. Confluence is the property that asserts that $u \overset{*}{\leftarrow} s \overset{*}{\rightarrow} v$ implies there exists a term t such that $u \overset{*}{\rightarrow} t \overset{*}{\leftarrow} v$. For example, the system $\{f(f(a)) \rightarrow f(a),\ g(f(a)) \rightarrow f(g(f(a))),\ g(f(a)) \rightarrow f(f(a))\}$ is confluent and the system $\{a \rightarrow f(a, b),\ f(a, b) \rightarrow f(b, a)\}$ is not. It is well known that confluence is not decidable for general term rewrite systems, but this paper proves it is for finite ground term rewrite systems following a conjecture made by Huet and Oppen (1980) in their survey. We also sketch an algorithm for checking this property. Actually we prove a more general result, namely the confluence of a family of term rewrite systems with infinitely many left-hand and right-hand sides is decidable. The general result induces the decidability of the confluence of left-linear and right-ground term rewrite systems. This algorithm is based on tree automata and tree transducers. Here, we regard them as rewrite systems and specialists in automata theory would translate that easily in their language.

Sketch of the Proof

The idea of the proof is to associate with ground term rewrite systems a kind of machine or automaton working on trees which enjoys nice properties related to rationality. These automata are called *ground tree transducers* or GTT in short. They describe transformations computed by different classes of term rewrite systems and provides a new tool whose algebraic study supplies our decision results. GTT can be seen as the combination of two bottom-up tree automata; one automaton works upward and the other downward. Since they are stable by inverse, by composition, and by semi-congruence closure, i.e., by precongruence, transitive, and reflexive closure (Section 3), the reflexive and transitive closure of the rewrite relation can be associated with a ground term rewrite system and, since the confluence is a problem of inclusion of relations, its check is just a check of the inclusion of two relations described by ground tree transducers. This last property can be tested through the inclusion of two relational tree languages, for which an algorithm is known. Thus a key of the proof is to build a natural correspondence between a ground tree transducer and a rational tree language with the same inclusion problem (Theorem 1).

Related Works

Recently, Oyamaguchi proposed a proof of the decidability of the confluence of ground term rewrite systems (Oyamagushi, 1987a) and a proof of the decidability of the confluence of left-linear and right-ground term rewrite systems (Oyamagushi, 1987b), which he calls "quasi-ground term rewriting systems." Both proofs are much longer that this presented here for a restricted result. Indeed, this paper gives a proof and a unique decision algorithm for a more general problem which contains as an instance the two results. Actually the ground tree transducers generalize the two classes of term rewrite systems considered by Oyamaguchi including infinite rewrite systems and algebraic properties of ground tree transducers improve specification and resolution of decision problems.

In (Gallier *et al.*, 1988) Gallier, Narendran, Snyder, and Plaisted proposed a polynomial $O(n^3)$ algorithm for building a confluent and noetherian set of rules associated to a set of identities and more recently Snyder (1980) proposed an $O(n \cdot \log(n))$ algorithm for the same problem. This obviously can be used to prove confluence and therefore provide a more efficient algorithm, but this works only for noetherian systems. On the other hand, termination of ground term rewrite systems was proved decidable in (Huet and Lankford, 1978).

This paper is an extension, with minor corrections, of (Dauchet *et al.*, 1987) which was intended to prove the decision of the confluence only for ground term rewrite systems, here it is shown how the same proof generalizes to a larger class of term rewrite systems we call GTT term rewrite systems.

Structure of the Paper

In the second part of this paper, a new class of tree transducers is introduced: *ground tree transducers.* In the third part, stability of GTT relation w.r.t., inverse, composition, transitive, and precongruence closure of union is proved. In the fourth part, each finite ground term rewrite system is associated with a GTT and the confluence of ground term rewrite system is reduced to the inclusion of GTT relations. Part 5 solves the decision problem by encoding GTT into rational tree languages.

2. Ground Tree Transducers

In this section, we define a special type of tree automata called ground tree transducers, to simulate the behavior of ground term rewrite systems. Notice that to stick to some kind of tradition we use the name "tree" with transducers and "term" with rewrite, but for us "labeled trees" and "terms"

are synonyms. The intuitive idea behind a ground tree transducer is to transform ground terms into ground terms in two steps, with a memory that uses a finite number of values or states. In the first step, one nibbles the term and in the second step one restores a new term. The nibbling consists of transforming some subterms of the term into constants, the states, and the restoration consists of transforming the states into subterms. These operations are rational, this means the relations they define satisfy properties expected for rational relations, namely stability w.r.t. union, composition, and iteration.

The steps of nibbling and restoration are described by a tree automaton which is nothing but a ground term rewrite system that works on a set of ground terms enriched by the states as constants. In the following, the set of states is named E (or E_G if one wants to refer to a ground term rewrite system G) and its elements are written e. Let F be a finite ranked alphabet where constants are nullary operators. $T(F)$ denotes the set of terms (or trees) on F. In what follows, it will always be supposed that $F \cap E_G =$ for all considered sets of states E_G.

DEFINITION 1. A relation $\rightarrow$ is *F-compatible* or is a *precongruence* if

$$(\forall f \in F)[(\forall 1 \leqslant i \leqslant arity(f))\ s_i \rightarrow t_i] \Rightarrow f(s_1, ..., s_i, ...) \rightarrow f(t_1, ..., t_i, ...,).$$

A *semi-congruence* is a reflexive and transitive precongruence.

Let us define bottom-up or frontier-to-root tree automata, regarding them as specific ground term rewrite systems. Specialists in automata theory would remark that the e_i's are states (Gécseg and Steinby, 1984).

DEFINITION 2. A *tree automaton* G is a rewrite system on $T(F \cup E_G)$ which contains only rules of the form:

$$f(e_1, ..., e_n) \rightarrow e_0 \text{ called } \textit{reduction transitions}$$

and

$$e \rightarrow e' \text{ called } \varepsilon\textit{-transitions.}$$

for $f \in F$ and $e_1, ..., e_n, e_0, e, e'$ in E_G.

As usual, it is understood that if $n = 0$ in the previous definition we have $f(\,) \rightarrow e_0$ that we also freely write $f \rightarrow e_0$. In other words, a state is associated with each constant. In the following, the possibility of n being equal to 0 is implicitly assumed. Usually if $\rightarrow$ denotes the transitions of a tree automaton, $\rightarrow$ is also used for denoting its precongruence closure, i.e., the smallest precongruence that contains it. The relation $\overset{*}{\rightarrow}$ is the *semi-*

congruence closure of $\rightarrow$; that is, the least semi-congruence on ground terms that contains $\rightarrow$. The concept of semi-congruence closure is important in this paper and is strongly connected with that of congruence closure (Downey *et al.*, 1980; Nelson and Oppen, 1980). The relation $\xrightarrow{*}_G$ is the natural extension of the tree automaton G. Reciprocally, given a relation R on $T(F \cup E)$, the tree automaton $\mathrm{Aut}(R)$ is defined by

$$f(e_1, ..., e_n) \rightarrow e_0 \in \mathrm{Aut}(R) \quad \text{if and only if} \quad f(e_1, ..., e_n)\, R e_0$$

$$e \rightarrow e' \in \mathrm{Aut}(R) \quad \text{if and only if} \quad eRe'.$$

If R is defined by a tree automata, i.e., $R \equiv \xrightarrow{*}_G$, one gets $\xrightarrow{*}_{\mathrm{Aut}(R)} \equiv \xrightarrow{*}_G$ and $G \subseteq \mathrm{Aut}(R)$.

Now we define a relation on terms $t \overset{T}{\rightarrow\leftarrow} t'$ which shows how ground rewrite steps can be simulated by a certain kind of composite automaton.

DEFINITION 3. A *ground tree transducer* on $T(F)$ (a GTT in short) is the relation T or (G, D) associated with two tree automata G and D and defined as

$$t \overset{T}{\rightarrow\leftarrow} t' \quad \text{iff there exists} \quad u \in T(F \cup (E_G \cap E_D)) \text{ such that } t \xrightarrow{*}_G u \xleftarrow{*}_D t'.$$

In order to produce actual pairs of terms, the set E_G and E_D are supposed non-disjoint. $E_G \cap E_D$ is called the *interface*. If we define a *context* as a linear term, i.e., a term with one occurrence of each variable numbered from left to right, it could be more intuitive to see u as a term of the form $c(e_1, ..., e_n)$, where $c(x_1, ..., x_n)$ is a common context of t and t' with $t = c(t_1, ..., t_n)$, $t' = c(t'_1, ..., t'_n)$, and $t_1 \xrightarrow{*}_G e_1 \xleftarrow{*}_D t'_1, ..., t_n \xrightarrow{*}_G e_n \xleftarrow{*}_D t_n$ (see Fig. 1.) A relation associated with a ground tree transducer is called a *GTT-relation.*

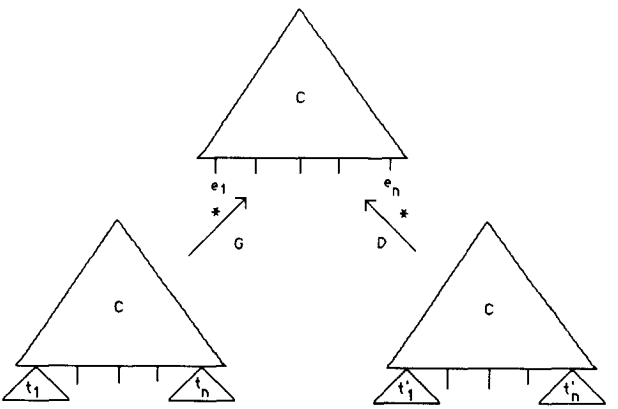

FIG. 1. A ground tree transduction.

EXAMPLE 1. A GTT associated with the ground term rewrite system

$$R = \{a \to f(a, b), f(a, b) \to f(b, a)\}$$

is defined as follows. The reason for choosing this precise GTT will be given in Section 4,

$$E_G = E_D = \{e_a, e_b, e_{f(a,b)}, e_{f(b,a)}\},$$

Recognition of subterms for G,

$$\begin{aligned} a(\,) &\to_G e_a \\ b(\,) &\to_G e_b \\ f(e_a, e_b) &\to_G e_{f(a,b)} \\ f(e_b, e_a) &\to_G e_{f(b,a)}. \end{aligned}$$

Recognition of subterms for D,

$$\begin{aligned} a(\,) &\to_D e_a \\ b(\,) &\to_D e_b \\ f(e_a, e_b) &\to_D e_{f(a,b)} \\ f(e_b, e_a) &\to_D e_{f(b,a)}. \end{aligned}$$

The rules of R as ε-transition,

$$\begin{aligned} e_a &\to_G e_{f(a,b)} \\ e_{f(a,b)} &\to_G e_{f(b,a)} \\ e_{f(a,b)} &\to_D e_a \\ e_{f(b,a)} &\to_D e_{f(a,b)}. \end{aligned}$$

Notice that $f(a, a) \overset{G,D}{\to\leftarrow} f(f(a, b), a)$ because

$$f(a,a) \to_G f(e_a, a) \to_G f(e_{f(a,b)}, a) \leftarrow_D f(f(e_a, e_b), a) \leftarrow_D f(f(a, e_b), a) \leftarrow_D f(f(a, b), a).$$

Indeed, a common context of $f(a, a)$ and $f(f(a, b), a)$ is $f(x, a)$ and one has $a \overset{*}{\to}_G e_{f(a,b)} \overset{*}{\leftarrow}_D f(a, b)$.

3. STABILITY OF GTT RELATIONS

From the definition, it is really easy to see that the inverse of a GTT-relation is still a GTT-relation, commuting the rôle of G and D. The union

of two GTT-relations R_1 and R_2, is not actually a GTT-relation. However the *precongruence closure* of $R_1 \cup R_2$, which is the least precongruence that contains R_1 and R_2, is associated with the GTT (G_3, D_3) defined by $G_3 \equiv G_1 \cup G_2$ and $D_3 \equiv D_1 \cup D_2$, where (G_1, D_1) and (G_2, D_2) are the GTT associated with R_1 and R_2. This supposes the set of states are disjoint, that is $(E_{G_1} \cup E_{D_1}) \cap (E_{G_2} \cup E_{D_2}) = \varnothing$. So we may state the following proposition.

PROPOSITION 1. 1. *The inverse of a GTT-relation is a GTT-relation.*

2. *The precongruence closure of the union of two GTT-relations is a GTT-relation.*

We are now going to address two other properties of GTT-relations, namely stability by composition and stability by semi-congruence closure.

PROPOSITION 2. *The semi-congruence closure of a GTT-relation is a GTT-relation.*

Proof. Let (G, D) be a given GTT associated with the relation R. Let us construct the GTT associated with the semi-congruence closure of R. Consider the GTT (G^*, D^*), where $E_{G^*} = E_{D^*} = E_G \cup E_D$ and

- $e \to_{G^*} e'$ if and only if $e(\overset{G, D}{\to\leftarrow})^* e'$.
- $f(e_1, ..., e_n) \to_{G^*} e_0$ if and only if $f(e_1, ..., e_n)(\overset{G, D}{\to\leftarrow})^* e_0$.
- $f(e_1, ..., e_n) \to_{D^*} e_0$ if and only if $e_0(\overset{G, D}{\to\leftarrow})^* f(e_1, ..., e_n)$.

(G^*, D^*) is associated with the semi-congruence closure of the relation R associated with (G, D). Indeed $(\overset{G, D}{\to\leftarrow})^*$ contains $\to_{G^*} \circ \leftarrow_{D^*}$ and since $(\overset{G, D}{\to\leftarrow})^*$ is a semi-congruence it contains $\overset{G^*, D^*}{\to\leftarrow}$. In the other direction, let $u(\overset{G, D}{\to\leftarrow})^* v$ then

$$u \overset{*}{\to}_G c_1(\cdots) \overset{*}{\leftarrow}_D \cdots \overset{*}{\to}_G c_i(\cdots) \overset{*}{\leftarrow}_D \cdots c_q(\cdots)\, v.$$

Consider $c(\cdots)$ the common part of the $c_i(\cdots)$, i.e., $\mathrm{Dom}(c) = \bigcap_i \mathrm{Dom}(c_i)$ and c/α is a variable if there exists an i such that c_i/α is a variable. At this occurrence α, c_i gets a state e_j. The term $c(e_1, ..., e_j, ..., e_p)$ plays a central rôle. By construction, c is a context of u and v. Therefore, $u \equiv c(u_1, ..., u_p)$ and $v \equiv c(v_1, ..., v_p)$. One has for each $j \in [1, p]$, $u_j(\overset{G, D}{\to\leftarrow})^* e_j$, and $e_j(\overset{G, D}{\to\leftarrow})^* v_j$. By definition of the GTT (G^*, D^*), this means $u_j \overset{*}{\to}_G e_j$ and $e_j \overset{*}{\leftarrow}_D v_j$ and, by putting all this results together, one gets

$$u \equiv c(u_1, ..., u_p)(\overset{G, D}{\to\leftarrow})^* c(e_1, ..., e_p)(\overset{G, D}{\to\leftarrow})^* c(v_1, ..., v_p) \equiv v$$

or

$$u \overset{*}{\to}_G c(e_1, ..., e_p) \overset{*}{\leftarrow}_D v;$$

therefore

$$u \overset{G^*, D^*}{\rightarrow\leftarrow} v$$

which means that $\overset{G^*, D^*}{\rightarrow\leftarrow}$ contains $(\overset{G, D}{\rightarrow\leftarrow})^*$. ∎

PROPOSITION 3. *The composition of two GTT-relations is a GTT-relation.*

Proof. Suppose that the two relations are associated with the GTT's (G, D) and (Γ, Δ) and that the sets of states satisfy the condition $E_D \cap E_\Gamma = \varnothing$. The relation C which is the composition of the relations associated with the two GTT's is given by $\overset{*}{\rightarrow}_G \circ \overset{*}{\leftarrow}_D \circ \overset{*}{\rightarrow}_\Gamma \circ \overset{*}{\leftarrow}_\Delta$. Let A be $\mathrm{Aut}(\overset{*}{\leftarrow}_D \circ \overset{*}{\rightarrow}_\Gamma)$ and B be $\mathrm{Aut}(\overset{*}{\leftarrow}_\Gamma \circ \overset{*}{\rightarrow}_D)$. C is equal to $\overset{*}{\rightarrow}_G \circ \overset{*}{\rightarrow}_A \circ \overset{*}{\leftarrow}_B \circ \overset{*}{\leftarrow}_\Delta$. By a renaming of the states this relation can be easily described as a GTT. ∎

4. TERM REWRITE SYSTEMS, TRANSDUCERS, AND CONFLUENCE

From the previous section, it is now easy to prove that the rewrite relation $\overset{*}{\rightarrow}_R$ deduced from a ground term rewrite system, that is the semi-congruence closure of the set of rules, can be described by a GTT. First one builds a GTT, (G, D), such that

$$E_G = E_D = \{e_s \mid s \in \mathrm{Subterm}(R)\},$$

where $\mathrm{Subterm}(R)$ is the set of all the subterms of left-hand and right-hand sides of rules in R. If $f(s_1, ..., s_n) \in \mathrm{Subterm}(R)$, then G and D contain the following rules:

$$f(e_{s_1}, ..., e_{s_n}) \rightarrow_G e_{f(s_1, ..., s_n)}$$

$$f(e_{s_1}, ..., e_{s_n}) \rightarrow_D e_{f(s_1, ..., s_n)}.$$

In addition, one adds the rules

$$e_l \rightarrow_G e_r$$

$$e_r \rightarrow_D e_l$$

for every rule $l \rightarrow r$ in R. (G, D) is now closed in the GTT (G^*, D^*) by the following properties:

- $f(e_1, ..., e_n) \rightarrow_{G^*} e_0$ if and only if $f(e_1, ..., e_n)(\overset{G, D}{\rightarrow\leftarrow})^* e_0$,
- $f(e_1, ..., e_n) \rightarrow_{D^*} e_0$ if and only if $e_0(\overset{G, D}{\rightarrow\leftarrow})^* f(e_1, ..., e_n)$,
- $e \rightarrow_{G^*} e'$ if and only if $e(\overset{G, D}{\rightarrow\leftarrow})^* e'$
- $e' \rightarrow_{D^*} e$ if and only if $e(\overset{G, D}{\rightarrow\leftarrow})^* e'$.

(G^*, D^*) is associated with the semi-congruence closure of the relation R.

This means that

$$u \overset{G^*, D^*}{\rightarrow\leftarrow} v \qquad \text{if and only if} \quad u \overset{*}{\rightarrow}_R v.$$

EXAMPLE 2. The GTT (Γ, Δ) which describes the rewrite relation associated with the relation

$$R = \{a \rightarrow f(a, b), f(a, b) \rightarrow f(b, a)\}$$

is obtained by closure of the GTT (G, D) of Example 1 as explained previously. The Γ-transitions of (Γ, Δ) that are not transitions of (G, D) are

$$\begin{aligned} a(\,) &\rightarrow_\Gamma e_{f(a,b)} \\ a(\,) &\rightarrow_\Gamma e_{f(b,a)} \\ f(e_a, e_b) &\rightarrow_\Gamma e_{f(b,a)}. \end{aligned}$$

For instance, $a(\,) \rightarrow_\Gamma e_{f(b,a)}$ comes from $a(\,) \rightarrow_G e_a \rightarrow_G e_{f(a,b)} \rightarrow_G e_{f(b,a)}$. The Δ-transitions of (Γ, Δ) that are not transitions of (G, D) are

$$\begin{aligned} f(e_a, e_b) &\rightarrow_\Delta e_a \\ f(e_b, e_a) &\rightarrow_\Delta e_{f(a,b)} \\ f(e_b, e_a) &\rightarrow_\Delta e_a. \end{aligned}$$

Actually, the previous construction works for the family of all the infinite ground term rewrite systems whose associated rewrite relation can be described by a GTT. Let us call them *GTT rewrite systems*. If the set of the instances of the rules is the Cartesian product of a rational set of ground terms (the instances of left-hand sides) by another rational set of ground terms (the instances of the right-hand sides) then the rewrite system is a GTT rewrite system. What we have built in the previous paragraph is a GTT based on automata that recognize finite sets of ground terms on both sides. Actually, this construction may work for non ground term rewrite systems provided we may prove the infinite set of ground instances of the rules fits in the previously described restrictions, namely they are a Cartesian product of rational languages. Rationality precludes non-linear terms. Indeed, it is well known that the set of instances of a term is rational if and only if the term is linear. For the same reason, a finite automaton cannot check that the right-hand sides receive the same instantiations as the left-hand sides. Therefore, the right-hand sides cannot contain occurrences of variables. A natural family that fulfills the previous requirements is the set of finite rewrite systems whose left-hand sides are all linear and whose right-hand sides are ground. We call these systems

left-linear and right-ground term rewrite systems and Oyamaguchi calls them *quasi-ground term rewriting systems* in (Oyamagushi, 1987b). Thus in (Oyamagushi, 1987a) Oyamagushi gives a proof for finite ground term rewrite systems and then in (Oyamagushi, 1987b) he gave a new proof for finite left-linear and right-ground term rewrite systems; in our case the proof intended to be a proof for ground term rewrite systems extends easily to a generalization of left-linear and right-ground term rewrite systems.

The Confluence Is an Inclusion Problem

Confluence of a rewrite relation $\rightarrow$ can be translated into the inclusion

$$\overset{*}{\leftarrow} \circ \overset{*}{\rightarrow} \subseteq \overset{*}{\rightarrow} \circ \overset{*}{\leftarrow}.$$

If $\overset{*}{\rightarrow}$ is a GTT-relation, from the previous results the two relations $\overset{*}{\leftarrow} \circ \overset{*}{\rightarrow}$ and $\overset{*}{\rightarrow} \circ \overset{*}{\leftarrow}$ are associated with a GTT. The decidability of this property will be a consequence of the following section which proves that the inclusion of two GTT-relations is decidable.

5. Decidability of the Inclusion of Two GTT-Relations

The idea for deciding the inclusion of two GTT-relations is to map each GTT-relation onto a regular tree language. Since the map is one-to-one, a language will be included in another language if and only if the associated relations are included one in the other. The map is defined as follows: With each pair,

$$t = c(t_1, ..., t_n) \overset{*}{\rightarrow}_G c(e_1, ..., e_n) \overset{*}{\leftarrow}_D t' = c(t'_1, ..., t'_n),$$

one associates the term $v\downarrow_{\#}$, where $v\downarrow_{\#}$ is the normal form of the term $v = \#(t, t')$ for the rewrite system $R_{\#}$ defined by the rules

$$\#(f(x_1, ..., x_p), f(y_1, ..., y_p)) \rightarrow_{R_\#} f(\#(x_1, y_1), ..., \#(x_p, y_p)).$$

Let us write $\mathscr{L}(T)$ for the language of the $v\downarrow_{\#}$'s and let us call it the *tensorial product* of the GTT T. Let us show that $\mathscr{L}(T)$ is recognized by a tree automaton $\mathscr{A}(T)$. The set of states is divided into four parts $E_G \times F$, $E_D \times F$, $\{ok\}$ and $E_G \times E_D$. The goal of taking the Cartesian product of the states with the set of functional symbols is to check that when a term is recognized it is actually in $R_{\#}$-normal form, so one has to remember the last read symbol and allow proceeding above the $\#$ only when the last symbol read on the left is different from the last symbol read on the right (rule (3) below). A state in $E_G \times F$ or in $E_D \times F$ will be written $e \cdot f$ and a

state in $E_G \times E_D$ will be written $\langle e, e' \rangle$. ok is the unique final state; in other words, $v \in \mathscr{L}(T)$ if and only if $v \xrightarrow{*}_{\mathscr{A}(T)} ok$. The transitions are of the form

$$f(e_1 \cdot g_1, ..., e_n \cdot g_n) \to e \cdot f \tag{1}$$

$$f(ok, ..., ok) \to ok \tag{2}$$

$$\#(e \cdot f, e' \cdot g) \to \langle e, e' \rangle \tag{3}$$

$$f(\langle e_1, e'_1 \rangle, ..., \langle e_n, e'_n \rangle) \to \langle e, e' \rangle \tag{4}$$

$$\langle e, e \rangle \to ok. \tag{5}$$

(3) describes a family of rules for all f and g such that $f \not\equiv g$. The family of rules of type (1) corresponds to the rules

$$f(e_1, ..., e_n) \to e$$

of G and D; one remembers, in addition, that the last read symbol is f. The family of rules (4) corresponds to the rules

$$f(e_1, ..., e_n) \to_G e$$
$$f(e'_1, ..., e'_n) \to_D e'.$$

The transitions of type (1) translate transitions of E_G and E_D; the transitions of type (2) are just for transmitting previous acknowledgments through the context. Since the system $R_\#$ pushes down the $\#$'s through terms with the same top, it could be the case that a pair is not completely recognized when a $\#$ appears. The transitions of type (3) allow the recognition of s and t to continue beyond $\#(s, t)$ and at the same time check that the term $\#(s, t)$ is in $R_\#$-normal form, as shown by the fact that s and t have different top symbols. Since, beyond this point, the not fully recognized terms are supposed to have the same symbols, the recognition of the rest of the two terms of the pairs has to be done on both terms together and has to work on a pair of states. This is the rôle of the rules of type (4). The last rule means that what has been recognized till this point is the $R_\#$-normal form of a term $\#(s, t)$, where $s \overset{T}{\to\leftarrow} t$. Notice that $\mathscr{A}(T)$ is non-deterministic. The following theorem says that $\mathscr{L}(T)$ and $\mathscr{A}(T)$ are correctly associated.

THEOREM 1. *The three following statements are equivalent:*

(i) $c(\#(t_1, t'_1), ..., \#(t_n, t'_n))\downarrow_\# \in \mathscr{L}(T)$

(ii) $c(t_1, ..., t_n) \overset{T}{\to\leftarrow} c(t'_1, ..., t'_n)$

(iii) $c(\#(t_1, t'_1), ..., \#(t_n, t'_n))\downarrow_{\#}$ *is recognized by the automaton* $\mathscr{A}(T)$; *in other words*

$$c(\#(t_1, t'_1), ..., \#(t_n, t'_n))\downarrow_{\#} \xrightarrow{*}_{\mathscr{A}(T)} ok$$

and only those terms reduce to ok.

Proof. (i) $\Leftrightarrow$ (ii) is a direct consequence of the definition of $\mathscr{L}(T)$. For the proof of (ii) $\Rightarrow$ (iii), let us look at the $\#(t_i, t'_i)$ part of $c(\#(t_1, t'_1), ..., \#(t_n, t'_n))$. Its $R_{\#}$-normal form $\#(t_i, t'_i)\downarrow_{\#}$ is of the form $c_i(\#(t_{i,1}, t'_{i,1}), ..., \#(t_{i,n}, t'_{i,n}))$. If $c(t_1, ..., t_n) \xrightarrow{T} \leftarrow c(t'_1, ..., t'_n)$, we have $t_i \xrightarrow{*}_G e_i \xleftarrow{*}_D t'_j$ for a state e_i of $\mathscr{A}(T)$. To recognize $\#(t_i, t'_i)\downarrow_{\#}$; i.e., to get $\#(t_i, t'_i)\downarrow_{\#} \xrightarrow{*}_{\mathscr{A}(T)} ok$ one uses the rules of type (1) for deriving $t_{i,j} \xrightarrow{*}_{\mathscr{A}(T)} e_{i,j} \cdot f_{i,j}$ and $t'_{i,j} \xrightarrow{*}_{\mathscr{A}(T)} e'_{i,j} \cdot g_{i,j}$, then the rules of type (3) to get $\#(e_{i,j} \cdot f_{i,j}, e'_{i,j} \cdot g_{i,j}) \rightarrow_{\mathscr{A}(T)} \langle e_{i,j} e'_{i,j} \rangle$, then the rules of type (4) to get

$$c_i(\#(t_{i,1}, t'_{i,1}), ..., \#(t_{i,n}, t'_{i,n})) \xrightarrow{*}_{\mathscr{A}(T)} \langle e_i, e_i \rangle,$$

then the rule (5) to get $\langle e_i, e_i \rangle \rightarrow_{\mathscr{A}(T)} ok$. So

$$c(\#(t_1, t'_1), ..., \#(t_n, t'_n))\downarrow_{\#} \xrightarrow{*}_{\mathscr{A}(T)} ok.$$

by using rule (2).

For the proof (iii) $\Rightarrow$ (ii), the problem is to rebuild the terms $c(t_1, ..., t_n)$ and $c(t'_1, ..., t'_n)$ from the recognition of

$$c(\#(t_1, t'_1), ..., \#(t_n, t'_n))\downarrow_{\#},$$

in other words to rebuild $c(\#(t_1, t'_1), ..., \#(t_n, t'_n))$ from its $R_{\#}$-normal form. The idea is just to push up the $\#$ in the term till the point where the rules $\langle e_i, e_i \rangle \rightarrow_{\mathscr{A}(T)} ok$ are used. ∎

Since the correspondence between $\mathscr{L}(T)$ and $R(T)$ is one-to-one, the following corollary is an easy consequence of the theorem.

COROLLARY 1. $R(T) \subseteq R(T')$ *if and only if* $\mathscr{L}(T) \subseteq \mathscr{L}(T')$.

The tree language $\mathscr{L}(T)$ is rational, since it is recognized by the bottom up or frontier-to-root automaton $\mathscr{A}(T)$. The presentation is a little different of that used in the automata literature, but the reader familiar with this approach can easily convince himself that $\mathscr{A}(T)$ is actually a bottom-up automaton. It is well known that the inclusion of rational tree languages is decidable (Gécseg and Steinby, 1984). This can be done by extending to bottom-up tree automaton the decision procedure for the inclusion of string rational languages. So we have the following corollary.

COROLLARY 2. *The inclusion of GTT-relations is decidable.*

From the previous section we get also the following main corollary.

COROLLARY 3. *The confluence of GTT term rewrite systems is decidable.*

From this general result we deduce the following result.

COROLLARY 4. *The confluence of finite ground tree rewrite systems is decidable.*

The construction of the GTT associated with a ground term rewrite system in Section 4 can be extended, especially instead of adding transitions

$$e_l \to_G e_r$$

$$e_r \to_D e_l;$$

for every rule $l \to r$ in R, we add the transitions

$$e_{\mathscr{L}_i} \to_G e_{\mathscr{R}_i}$$

$$e_{\mathscr{R}_i} \to_D e_{\mathscr{L}_i},$$

where $e_{\mathscr{L}_i}$ (resp. $e_{\mathscr{R}_i}$) is the final state for the automaton that recognizes $\mathscr{L}_i$ (resp. $\mathscr{R}_i$). Therefore we can state the following corollary.

COROLLARY 5. *The confluence of infinite ground term rewrite systems of the form*

$$\bigcup_{i=1}^{n} \{l \to r \mid l \in \mathscr{L}_i \,\&\, r \in \mathscr{R}_i\},$$

where $\mathscr{L}_i$ and $\mathscr{R}_i$ are rational tree languages, is decidable.

If the left-hand sides are linear, the set of their ground instances are rational trees; therefore one can state the following corollary.

COROLLARY 6. *The confluence of finite, left-linear, and right-ground term rewrite systems is decidable.*

6. AN ALGORITHM FOR DECIDING THE CONFLUENCE OF GROUND TERM REWRITE SYSTEMS

A possible algorithm for deciding the confluence of finite ground term rewrite system can be divided into three main steps:

- build the GTT associated with the term rewrite system,
- build the GTTs associated witht the relations $\overset{*}{\leftarrow} \circ \overset{*}{\rightarrow}$ and $\overset{*}{\rightarrow} \circ \overset{*}{\leftarrow}$ and then build the rational languages associated with these relations.
- decide the inclusion of the rational tree languages, by classical tree automata techniques.

7. Conclusion

We have shown that the problem of deciding the confluence of a very general family of ground term rewrite systems can be reduced to the problem of deciding the inclusion of two rational tree languages, a problem which has a well-known decision procedure. As a particular case we prove that the confluence of finite ground term rewrite systems is decidable. It should be noticed that this decision algorithm is essentially useful for non-terminating term rewrite systems, because otherwise a method based on superposition (detection of subterms in this case) similar to the Knuth–Bendix algorithm or a method based on congruence closure like in (Gallier *et al.*, 1988) would work. Moreover, the complexity of the algorithm we propose is rather high, since it is based on inclusion of rational languages. It is disturbing to see that the problem of building a canonical set associated with a set of ground identities is much simpler, since it is in $O(n \cdot \log(n))$. Therefore, the greatest lower bound of the complexity of the decision algorithm for the confluence of finite ground terms rewriting systems is an interesting open problem.

Acknowledgments

The authors thank Jurgen Avenhaus, Wayne Snyder, Sándor Vágvölgyi, Zoltán Fülöp, and the referees for their useful comments.

Received November 13, 1987; final manuscript received June 21, 1989

References

Dauchet, M., Heuillard, T., Lescanne, P., and Tison, S. (1987), Decidability of the confluence of ground term rewriting systems, *in* "Proceedings, 2nd IEEE Symp. on Logic In Computer Science, Ithaca, New York," pp. 353–359.

Dauchet, M., and Tison, S. (1985), Tree automata and decidability in ground terms rewriting systems, *in* "FCT' 85," pp. 80–84, Lecture Notes in Computer Science, Springer-Verlag, New York/Berlin.

Downey, R., Sethi, P. J. and Tarjan, R. E. (1980), Variations on the common subexpression problem, *J. Assoc. Comput. Mach.* **27**, No. 4, 758–771.

GALLIER, J., NARENDRAN, P., PLAISTED, D., RAATZ, S., AND SNYDER, W. (1988), Finding canonical rewriting systems equivalent to a finite set of ground equations in polynomial time, *in* "Proceedings 9th Int. Conf. on Automated Deduction," (E. Lusk and R. Overbeek, Eds.), 182–196, Lecture Notes in Computer Science, Springer-Verlag, New York/Berlin.

GÉCSEG, F., AND STEINBY, M. (1984), "Tree Automata," Akad. Kiaddá, Budapest, Hungary.

HUET, G., AND LANKFORD, D. (1978), "On the Uniform Halting Problem for Term Rewriting Systems," Technical Report 283, Laboria.

HUET, G., AND OPPEN D. (1980), Equations and rewrite rules: A survey, *in* "Formal Languages Theory: Perspectives and Open Problems," pp. 349–405. Academic Press, New York/London.

NELSON, C. G., AND OPPEN, D. C. (1980), Fast decision procedures based on congruence closure, *J. Assoc. Comput. Mach.* **27**, No. 2, 356–364.

OYAMAGUCHI, M. (1987a), The Church–Rosser property for ground term rewriting systems is decidable, *Theoret. Comput. Sc.* **49**, 43–79.

OYAMAGUCHI, M. (1987b), "The Church–Rosser Property for Quasi-Ground Term Rewriting Systems," Technical Report, Faculty of Engineering, Mie University, Tsu 514, Japan.

SNYDER, W. (1989), Efficient completion: A $O(n \cdot \log(n))$ algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations E, *in* "Proceedings, 3rd Conf. on Rewriting Techniques and Applications,, (N. Dershowitz, Ed.), Lecture Notes in Computer Science, Springer-Verlag, New York/Berlin.