Theoretical Computer Science 127 (1994) 69–98
Elsevier

Bottom-up tree pushdown automata: classification and connection with rewrite systems*,**

Jean-Luc Coquidé, Max Dauchet and Rémi Gilleron
LIFL, URA 369 CNRS, IEEA Université de Lille I, 59655 Villeneuve d'Ascq Cedex, France

Sándor Vágvölgyi
Research Group on Theory of Automata, Hungarian Academy of Sciences, Aradi Vértanuk tere 1., H-6720 Szeged, Hungary

Communicated by M. Nivat
Received September 1991
Revised December 1992

Abstract

Coquidé, J.-L., M. Dauchet, R. Gilleron and S. Vágvölgyi, Bottom-up tree pushdown automata: classification and connection with rewrite systems, Theoretical Computer Science 127 (1994) 69–98.

We define different types of bottom-up tree pushdown automata and study their connections with rewrite systems. Along this line of research we complete and generalize the results of Gallier, Book and Salomaa. We define the notion of a tail-reduction-free (trf) rewrite system. Using the decidability of ground reducibility, we prove the decidability of the trf property. Monadic rewrite systems of Book, Gallier and Salomaa become a natural particular case of trf rewrite systems. We associate a deterministic bottom-up tree pushdown automaton with any left-linear trf rewrite system. Finally, we generalize monadic rewrite systems by introducing the notion of a semi-monadic rewrite system and show that, like a monadic rewrite system, it preserves recognizability.

1. Introduction

Equations and rewrite systems have been extensively used to specify programs and data types, see Goguen et al. [13] for one of the seminal papers, Huet and Oppen [17],

Correspondence to: M. Dauchet, LIFL, URA 369 CNRS, IEEA Université de Lille I, 59655 Villeneuve d'Ascq Cedex, France. Email: dauchet@lifl.fr.

* This research was carried out while S. Vágvölgyi was visiting the department of Computer Science of the University of Lille, and was partially supported by "PRC/GDR Mathématiques et Informatique" and ESPRIT Basic Research Action 6317 ASMICS2.

** A preliminary version of this paper was published in the proceedings of the 4th RTA Conference on Rewriting Techniques and Applications.

SSDI 0304-3975(93)E0008-R

Dershowitz and Jouannaud [7] for overviews. The paradigm of rewrite systems models evaluation in logic programming as well as interpreters in functional programming. On the one hand, most of the interesting properties of rewrite systems are undecidable, while on the other, several interesting results were obtained on special kinds of rewrite systems. The most celebrated result is the Knuth–Bendix completion algorithm, the successful termination of which results in a convergent rewrite system. Thus, researchers obtain decidable properties of fragments of theories or of subclasses of rewrite systems to supply tools for software engineering.

At the same time, stacks are a basic data structure in computer science, for example in syntactical analysis or for recursive procedure calls. For strings, the equivalence of context-free grammars and pushdown automata is well known, and connections between string rewriting systems and pushdown automata have been studied as well (see [1] for a survey).

The aim of this paper is to study connections between bottom-up tree pushdown automata and rewrite systems following the works of Gallier and Book [10] and Salomaa [19]. To this end we distinguish three purposes which are mixed in the preceding papers.

- Introduce new types of tree pushdown automata (tpda's) and compare the classes of tree transformations induced by them.
- Given a rewrite system S, construct a tpda computing normal forms of ground terms with respect to S. From this point of view, tree pushdown automata can be seen as the algorithmic aspect of problems specified by rewrite systems. Henceforth, tree pushdown automata can be used for normalization of rewrite systems.
- Find rewrite system classes with "good" properties (decision and complexity results, relation with context-free and recognizable tree languages,...).

In the first research area, we introduce several types of bottom-up tree pushdown automata and compare their transformational power. All reasonable definitions, when restricted to strings, should, of course, be equivalent to ordinary pushdown automata. In the tree case, two "normalization" problems arise: the first one is about the depth of the popped terms, and the second is about the rank of the states. These problems are not too deep, but it is useful to carefully study the situation.

In the word case, a transition rule of a pushdown automaton is of the form $(q, a, b) \to (q', w)$, where q and q' are states, a is a letter of the input alphabet or the empty word ε, b is a letter of the stack alphabet and w is a string over the stack alphabet. During the corresponding move, a is read, b is popped and w is pushed. It is well known that if we generalize the definition, allowing b to be a string, then we do not modify the power of the pushdown automaton. The reason is that we can pop letters of b step-by-step, memorizing it in the state.

The situation is quite different in the tree case for the reason illustrated by the following example. Consider the following reduce rule of a tpda: $q(b(b(x, y), z)) \to q'(c(x, y, z))$, where q and q' are states, b and c are stack symbols, and x, y, z are variables. If, as usual, we consider states of rank 1, the rule cannot be simulated by a sequence of rules popping only one letter, because when we pop the

first b, we delete one of the subtrees at its two sons. A way to overcome this problem is accepting any rank for the states. Thus, using an intermediate state q'' of rank 2, we simulate the above rule by the rules $q(b(x,y))\to q''(x,y)$ and $q''(b(x,y),z)\to q'(c(x,y,z))$ where q, q' and q'' are states, b and c are stack symbols, and x, y, z are variables. We see that this construction is quite natural. We define different types of tree pushdown automata depending on the rank of the states and on the depth of the popped terms. We can note that both the input and the stacks are trees. The rules and the computation of a bottom-up tree pushdown automaton with states of rank 1 are described in the Fig. 1.

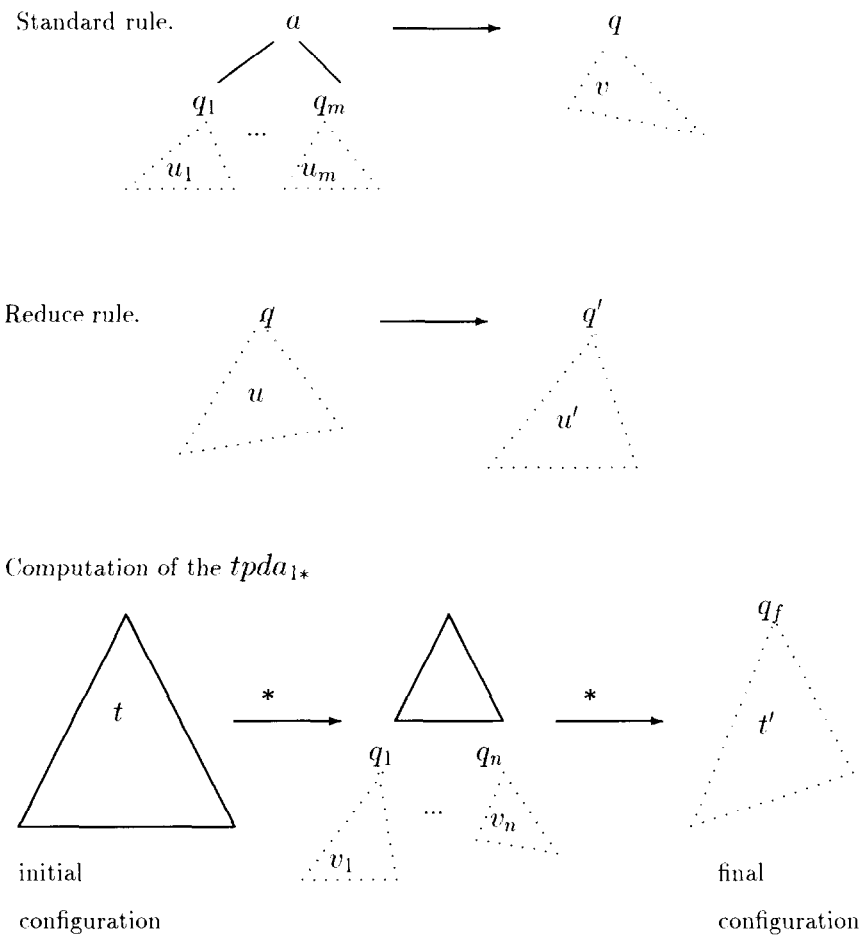

Fig. 1. The rules and the computation of a tpda_{1*}. Stack trees are indicated by dotted lines. The states $q, q', q_f, q_1, \ldots, q_m, q_1, \ldots, q_n$ are of rank 1, the stack trees appearing in the left-hand sides of the rules, $u, u_1, \ldots, u_m$ are of any depth.

Let T be a tpda, the maximum rank of the states of T is called the rank of T and is denoted by *rank*(T), the maximum depth of the stack trees appearing in the left-hand sides of the rules of T is called the depth of T and is denoted by *depth*(T). Let tpda_{**}, tpda_{n*}, tpda_{*k} and tpda_{nk} denote, respectively, a tpda of any rank and any depth, of rank n and any depth, of any rank and depth k, of rank n and depth k. Moreover, from [10] we introduce the tree pushdown automaton with read rules and reduce rules (tpda^{rr} for short). Let $TPDA_{**}$, $TPDA_{n*}$, $TPDA_{*k}$, $TPDA_{nk}$ and $TPDA^{\text{rr}}$ be the classes of all tpda_{**}'s, tpda_{n*}'s, tpda_{*k}'s, tpda_{nk}'s and tpda^{rr}'s, respectively. The corresponding tree transformation classes will be denoted by $TPDT_{**}$, $TPDT_{n*}$, $TPDT_{*k}$, $TPDT_{nk}$ and $TPDT^{\text{rr}}$, respectively.

We shall show that the classes $TPDA_{**}$, $TPDA_{1*}$, $TPDA_{*1}$, $TPDA_{12}$ and $TPDA^{\text{rr}}$ are equivalent to each other in a strong sense: the constructions permit to simulate in real time a tpda T of some class with a tpda T' of another class. More precisely, any step of T is mimicked by a bounded length sequence of steps of T'. We are interested in real-time simulation because if we do not impose such a restriction the distinctions between different classes disappear (we will make this remark more precise with the simulation of rewrite systems by trf rewrite systems in the sequel of the introduction, see also Section 4.2). On the other hand, there is a tree transformation induced by a tpda_{12} such that no tpda_{11} induces it. In other words, $TPDT_{11} \subset TPDT_{12}$ and $TPDA_{12} \equiv_{\text{rt}} TPDA_{1*} \equiv_{\text{rt}} TPDA_{*1} \equiv_{\text{rt}} TPDA_{**} \equiv_{\text{rt}} TPDA^{\text{rr}}$.

Moreover we say that a tree pushdown automaton is deterministic if its rewrite rule set is left-linear and is without critical pairs. This condition ensures the confluence of the induced move relation (see [15, 16]). We compare the deterministic tree pushdown automata classes as well, and obtain the deterministic versions of the inclusions and equalities holding for the nondeterministic classes. That is to say, we shall show that the classes $DTPDA_{**}$, $DTPDA_{1*}$, $DTPDA_{*1}$, $DTPDA_{12}$ and $DTPDA^{\text{rr}}$ are real-time equivalent to each other, and that there is a tree transformation induced by a dtpda_{12} such that no dtpda_{11} induces it.

In the second research area, we introduce the *tail-reduction-free* (trf) *property*. The tail of a term is the sequence of subterms that we obtain by erasing the head. We say that a rewrite system S is tail-reduction-free if for every right-hand side r of S and for any ground substitution σ being irreducible for S, $\sigma(tail(r))$ is irreducible as well. In other words, $\sigma(r)$ can only be reduced by matching at the head of r whenever r is a right-hand side of a rule in S and σ is an irreducible substitution for S. We prove that this property is decidable, reducing it to the decidability of ground reducibility (cf. [18]).

In order to illustrate this formal definition, consider the following two rewrite systems.

Example 1.1 (*Peano rules*). The rewrite system S_1 is defined by

$$+(s(x), y) \to +(x, s(y)),\ +(0, x) \to x.$$

As no $+$ occurs in an irreducible ground term, rewrite system S_1 is trf.

Example 1.2. Consider the ranked alphabet $\{a, b, c\}$, where a, b are of rank 1 and c is of rank 0. S_2 contains only the rule

$$b(a(x)) \to a(b(x)).$$

Rewrite system S_2 is not trf, considering the irreducible ground substitution $\sigma(x) = a(c)$, we obtain that $\sigma(tail(a(b(x)))) = b(a(c))$ is reducible.

It is easy to simulate any rewrite system by a trf rewrite system (over an enlarged alphabet), see Section 4.2; the simulation is not a real-time simulation because, in order to simulate a rewrite step, we can visit arbitrarily large subterms. Hence, this result is like saying that trf rewrite systems can simulate Turing machines.

We shall investigate how the trf property is related to bottom-up tree pushdown automata (see Section 4). This notion, both, generalizes preceding studies and illuminates the deep connection between tree pushdown automata and rewrite systems. More precisely, Gallier and Book [10] and Salomaa [19] associated tree pushdown automata only with what they call *monadic rewrite systems*. A rewrite system on a ranked alphabet Σ is monadic if each left-hand side is of depth at least 1 and each right-hand side is of depth at most 1. Monadic rewrite systems are obviously trf, since the tail of a right-hand side contains variables and nullary function symbols only, and the nullary symbols are irreducible. In this paper we assign tree pushdown automata to trf rewrite systems and in this way we generalize the results of Book, Gallier and Salomaa. A tree pushdown automaton with reduce priority is a tpda which applies a read rule rewriting the subtree $\gamma(q_1(u_1), \ldots, q_n(u_n))$ into the tree $q(\delta(u_1, \ldots, u_n))$ only when no reduce rule is applicable for the trees $q_i(u_i)$, $1 \leqslant i \leqslant n$. We show that for every convergent trf rewrite system S, one can construct a bottom-up tree pushdown automaton A with reduce priority such that, for an arbitrary ground term, A computes (in real time) its normal form. Furthermore, to any left-linear convergent trf rewrite system S, we assign a deterministic tree pushdown automaton computing (in real time) the normal forms of ground terms.

In the third research area, we introduce the class of *semi-monadic rewrite systems* (see Sections 4.2 and 5.1). This class keeps the same good properties as the monadic one, and embeds both the monadic class and the class of ground rewrite systems, the theory of which is decidable (see [5]). In Section 4.2 we show that for each convergent semi-monadic rewrite system S, there exists a convergent trf semi-monadic rewrite system S' such that each ground term has the same normal form for S as for S'. In Section 5.1, we study semi-monadic rewrite systems from the formal language point of view. Extending a result of Salomaa [19], we prove that linear semi-monadic rewrite systems preserve recognizability.

2. Preliminaries

We recall and invent some notations, basic definitions and terminology which will be used in the rest of the paper. Nevertheless, the reader is assumed to be familiar with

the basic concepts of rewrite systems and of tree language theory (see e.g. [7, 10, 11, 17, 19]). Hence, we sometimes only recall the notations of objects.

2.1. Terms and substitutions

Σ is a set, Σ^* is the free monoid generated by Σ with empty word ε as identity element. The length of a word $w \in \Sigma^*$, denoted by $length(w)$, is defined as usual. The set of nonnegative integers is denoted by ω, and ω^* stands for the free monoid generated by ω with empty word ε as identity element.

A ranked alphabet is a finite set Σ in which every symbol has a unique rank in ω. For $m \geqslant 0$, Σ_m denotes the set of all elements of Σ which have rank m. For any set Y and ranked alphabet Σ, the set $T_\Sigma(Y)$ of Σ-terms (or Σ-trees) over Y is the smallest set satisfying

(a) $\Sigma_0 \cup Y \subseteq T_\Sigma(Y)$ and

(b) $b(t_1, \ldots, t_m) \in T_\Sigma(Y)$ whenever $m \geqslant 1$, $b \in \Sigma_m$ and $t_1, \ldots, t_m \in T_\Sigma(Y)$.

If $Y = \emptyset$, then $T_\Sigma(Y)$ is written as T_Σ. A term $t \in T_\Sigma(Y)$ is a ground term if $t \in T_\Sigma$ also holds. We specify a countable set $X = \{x_1, x_2, \ldots\}$ of variables which will be kept fixed in this paper. Moreover, we put $X_m = \{x_1, \ldots, x_m\}$ for $m \geqslant 0$. Hence, $X_0 = \emptyset$.

We need a few functions on terms. For a term $t \in T_\Sigma(X)$, the depth $depth(t) \in \omega$, the size $|t| \in \omega$, the frontier $fr(t) \in X^*$, the set $var(t) \subseteq X$, and the set of occurrences $O(t) \subseteq \omega^*$ are defined by induction:

(a) if $t \in \Sigma_0 \cup X$, then

$$depth(t) = 0,$$

$$|t| = 1,$$

$$fr(t) = \varepsilon \text{ if } t \in \Sigma_0 \quad \text{and} \quad fr(t) = t \text{ if } t \in X,$$

$$var(t) = \emptyset \text{ if } t \in \Sigma_0 \quad \text{and} \quad var(t) = t \text{ if } t \in X,$$

$$O(t) = \{\varepsilon\};$$

(b) if $t = b(t_1, \ldots, t_m)$, with $m \geqslant 1$ and $b \in \Sigma_m$, then

$$depth(t) = 1 + max\{depth(t_i) \mid 1 \leqslant i \leqslant m\},$$

$$|t| = 1 + |t_1| + \cdots + |t_m|,$$

$$fr(t) = fr(t_1) \ldots fr(t_m), \qquad var(t) = var(t_1) \cup \cdots \cup var(t_m),$$

$$O(t) = \{\varepsilon\} \cup \{i\alpha \mid 1 \leqslant i \leqslant m \text{ and } \alpha \in O(t_i)\}.$$

We note that $depth(t) = max\{length(\alpha) \mid \alpha \in O(t)\}$.

For occurrences $\alpha, \beta \in O(t)$, β is said to be a (proper) successor of α if α is a (proper) prefix of β.

Moreover, for any tree $t \in T_\Sigma(X)$, $path(t)$ denotes the set of all strings of symbols in $\Sigma \cup X$ that occur as labels of a path from the head to a leaf of t. More formally,

(a) if $t \in \Sigma_0 \cup X$, then $path(t) = \{t\}$,

(b) if $t = b(t_1, \ldots, t_m)$ with $m \geqslant 1$ and $b \in \Sigma_m$, then

$$path(t) = \{b\gamma \mid 1 \leqslant i \leqslant m,\ \gamma \in path(t_i)\}.$$

For each $t \in T_\Sigma(X)$ and $\alpha \in O(t)$, we introduce the subterm $str(t, \alpha) \in T_\Sigma(X)$ of t at α and the label $lab(t, \alpha) \in \Sigma \cup X$ in t at α as follows:

(a) for $t \in \Sigma_0 \cup X$, $str(t, \varepsilon) = t$ and $lab(t, \varepsilon) = t$;

(b) for $t = b(t_1, \ldots, t_m)$ with $m \geqslant 1$ and $b \in \Sigma_m$, if $\alpha = \varepsilon$ then $str(t, \alpha) = t$ and $lab(t, \alpha) = b$; otherwise, if $\alpha = i\beta$ with $1 \leqslant i \leqslant m$, then $str(t, \alpha) = str(t_i, \beta)$ and $lab(t, \alpha) = lab(t_i, \beta)$.

The head $Head(t)$ of a tree t is defined by the equation $Head(t) = lab(t, \varepsilon)$.

A tree $t \in T_\Sigma(X)$ is called linear if each x_i appears at most once in the string $fr(t)$.

For each $m \geqslant 0$, we distinguish a subset $\bar{T}_\Sigma(X_m)$ of the set of linear terms in $T_\Sigma(X_m)$ as follows: for $t \in T_\Sigma(X_m)$, $t \in \bar{T}_\Sigma(X_m)$ if and only if $fr(t) = x_1 \ldots x_m$. We call $\bar{T}_\Sigma(X_m)$ the set of contexts over Σ.

A substitution is a mapping $\sigma : X \to T_\Sigma(X)$ which is different from the identity only for a finite subset $Dom(\sigma)$ of X. We do not distinguish σ from its canonical extension to $T_\Sigma(X)$. For $V \subset X$, we define the restriction $\sigma_{/V}$ of σ to V as $\sigma_{/V}(x) = \sigma(x)$ if $x \in V$, $\sigma_{/V}(x) = x$ otherwise. For any trees $t \in T_\Sigma(X_m)$, $t_1, \ldots, t_m \in T_\Sigma(X)$ and for the substitution σ with $Dom(\sigma) = X_m$ and $\sigma(x_i) = t_i$ for $i = 1, \ldots, m$, we denote the term $\sigma(t)$ by $t(t_1, \ldots, t_m)$ as well. A substitution σ is ground if for every variable $x \in Dom(\sigma)$, $\sigma(x)$ is a ground term. A term t matches a term s if $\sigma(s) = t$ for some substitution σ; in that case we also say that t is an instance of s.

A unifier of two terms $t_1, t_2 \in T_\Sigma(X)$ is a substitution θ such that $\theta(t_1) = \theta(t_2)$. A most general unifier of t_1 and t_2 is a unifier θ of t_1 and t_2 such that, for each unifier σ of t_1 and t_2, there is a substitution σ' satisfying that $\sigma'(\theta(t_1)) = \sigma(t_1)$ and $\sigma'(\theta(t_2)) = \sigma(t_2)$. Finally, note that if t_1 and t_2 are unifiable, then there exists a most general unifier of t_1 and t_2.

We adopt the concept of *k-normal tree* from [9]. For an integer $k \geqslant 0$, we say that a tree $t \in T_\Sigma(X)$ is a k-normal tree over Σ if it satisfies the following conditions:

(a) $t \in \bar{T}_\Sigma(X_l)$ for some $l \geqslant 0$,

(b) for every $\alpha \in O(t)$, either ($length(\alpha) = k$ and $lab(t, \alpha) \in X_l$) or ($length(\alpha) < k$ and $lab(t, \alpha) \in \Sigma$).

We denote by $T_\Sigma(X, k)$ the set of k-normal trees over Σ. Moreover, let $T_\Sigma(X_m, k) = T_\Sigma(X_m) \cap T_\Sigma(X, k)$. We adopt the following result from Fülöp and Vágvölgi [9].

Proposition 2.1. *For any $k \geqslant 0$ and $t, t' \in T_\Sigma(X, k)$, there exist substitutions σ and σ' such that $\sigma(t) = \sigma'(t')$ if and only if $t = t'$. In other words, different k-normal trees have no common instances. Moreover, for every ground term u, there exists one and only one k-normal tree t such that u matches t.*

2.2. Rewrite systems

Let Σ be a ranked alphabet. Then a rewrite system S over Σ is a finite subset of $T_\Sigma(X) \times T_\Sigma(X)$ such that, for each $(l, r) \in S$, each variable of r also occurs in l. Elements (l, r) of S are called rules and are denoted by $l \to r$.

S induces a binary relation $\to_S$ over $T_\Sigma(X)$ defined as follows: for any $t, t' \in T_\Sigma(X)$, $t \to_S t'$ if and only if there exists a context c in $\bar{T}_\Sigma(X_1)$, a rule $l \to r$ in S and a substitution σ such that $t = c(\sigma(l))$ and $t' = c(\sigma(r))$.

$\to_S^*$ is the reflexive and transitive closure of $\to_S$, $\to_S^n$ denotes the n-fold composition of $\to_S$.

A ground rewrite system is one of which all rules are ground (i.e. elements of $T_\Sigma \times T_\Sigma$). A left-linear (right-linear, linear) rewrite system is one in which no variable occurs more than once on any left-hand side (right-hand side, right-hand side and left-hand side).

A term $t \in T_\Sigma(X)$ is called irreducible for S if there does not exist t' with $t \to_S t'$. The set of all irreducible, ground terms for S is denoted by $IRR(S)$. It is not hard to prove the following result.

Proposition 2.2. *Let S be a rewrite system over some ranked alphabet Σ. Then $IRR(S) \neq \emptyset$ if and only if*

(a) *there is no rule $l \to r$ in S such that $l \in X$, and*

(b) *there exists a symbol $\delta \in \Sigma_0$ such that there is no rule in S with left-hand side δ.*

For ground terms c, $t \in T_\Sigma$, we say that t is a normal form of c with respect to S if $c \to_S^* t$ and $t \in IRR(S)$.

A substitution σ is irreducible for S if for every variable $x_i \in Dom(\sigma)$, $\sigma(x_i)$ is irreducible for S.

Let $\to$ be a binary relation on a set A. We say that $\to$ is

(i) confluent if, for every $u, v_1, v_2 \in A$, it holds that if $u \to^* v_1$ and $u \to^* v_2$, then there exists a $v_3 \in A$ such that $v_1 \to^* v_3$ and $v_2 \to^* v_3$;

(ii) noetherian if there is no infinite sequence $v_1 \to v_2 \to v_3 \to \cdots$;

(iii) convergent if $\to$ is confluent and noetherian.

A rewrite system S is confluent (noetherian, convergent) if the induced rewrite relation is confluent (noetherian, convergent).

We adopt the concept of a critical pair from [15]. Let S be a rewrite system and assume that $(l_1 \to r_1), (l_2 \to r_2) \in S$. Let us define the rule $l_2' \to r_2'$ from $l_2 \to r_2$ by renaming the variables such that $var(l_1) \cap var(l_2') = \emptyset$. (More precisely, take a substitution $\theta : X \to X$ such that the restriction of θ to $var(l_2)$ is injective and that for each $x \in var(l_2)$, $\theta(x) \notin var(l_1)$. Then let $l_2' = \theta(l_2)$ and $r_2' = \theta(r_2)$.)

Let us assume that there is a tree $t = str(l_1, \alpha)$, where $\alpha \in O(l_1)$, such that $t \notin X$, t and l_2' are unifiable. Let σ be a most general unifier of t and l_2'. Let $v_1 = \sigma(r_1)$ and define v_2

from $\sigma(l_1)$ by substituting $\sigma(r'_2)$ for the subterm $\sigma(t)=\sigma(l'_2)$ at the occurrence α. Then we call (v_1, v_2) a critical pair of S.

We recall that if the rewrite system S is without critical pairs and is left-linear, then S is confluent (see [15, 16]).

Let A be a set. We say that the relation $<\subseteq A\times A$ is a partial order on A if $<$ is irreflexive and transitive. Let $<$ be a partial order on A; we say that $<$ is a linear order if, for every $u, v\in A$ with $u\neq v$, it holds that either $u<v$ or $v<u$.

Next we generalize the concept of the innermost rewrite relation with respect to a linear order, introduced in [10] and [19]. For a rewrite system S and a partial order $<$ on S, we define the *innermost rewrite relation induced by S with respect to* $<$, denoted by $\rightarrow_{i,S,<}$, as follows:

Let $t, t'\in T_\Sigma$. Then $t\rightarrow_{i,S,<} t'$ if $t=c(\sigma(l))$ and $t'=c(\sigma(r))$, where $c\in\bar{T}(X_1)$ with $lab(c,\alpha)=x_1$ for a unique $\alpha\in O(c)$, $(l\rightarrow r)\in S$ and σ is a substitution, such that the following conditions hold:

(i) For all proper successors β of α such that $\beta\in O(t)$, $str(t,\beta)$ is not an instance of any left-hand side of a rule in S.

(ii) If (l', r') is greater than (l, r) with respect to the partial order $<$, then $str(t,\alpha)$ is not an instance of l'.

In other words, $t\rightarrow_{i,S,<} t'$ if $t\rightarrow_S t'$ by a rule that is applied at an occurrence α such that no rule is applicable below α and there is no greater rule applicable at occurrence α.

If the partial order $<$ is equal to the empty set $\emptyset$, then $\rightarrow_{i,S,\emptyset}$ is the innermost rewrite relation induced by S and is the innermost strategy of rewriting. We shall write $\rightarrow_{i,S}$ rather than $\rightarrow_{i,S,\emptyset}$. Note that we can also consider a linear order $<$ on S.

2.3. Tree languages

A bottom-up tree automaton is a quadruple $A=(\Sigma, Q, Q_f, R)$, where Σ is a ranked alphabet, Q is a finite set of states of rank 0, Q_f $(\subseteq Q)$ is the set of final states, R is a finite set of rules of the following two types:

(i) $\delta(q_1,\ldots,q_n)\rightarrow q$ with $n\geqslant 0$, $\delta\in\Sigma_n$, $q_1,\ldots,q_n, q\in Q$.

(ii) $q\rightarrow q'$ with $q, q'\in Q$(ε-rules).

We consider R as a ground rewrite system over $\Sigma\cup Q$. The move relation $\rightarrow_A$ of A is the rewrite relation $\rightarrow_R$, i.e., $\rightarrow_A=\rightarrow_R$. The tree language recognized by A is $L(A)=\{t\in T_\Sigma \mid (\exists q\in Q_f) t\rightarrow_A^* q\}$.

We say that A is deterministic if R has no ε-rule and no two rules with the same left-hand side. A tree language L is recognizable if there exists a bottom-up tree automaton A such that $L(A)=L$.

The automaton $A=(\Sigma, Q, Q_f, R)$ is connected if for every $q\in Q$ there exists $t\in T_\Sigma$ such that $t\rightarrow_A^* q$. Every recognizable tree language can be recognized by a bottom-up tree automaton without ε-rules, by a (deterministic) connected bottom-up tree automaton (see [11]).

3. Bottom-up tree pushdown automata

3.1. General definitions and basic results

Definition 3.1. A bottom-up tree pushdown automaton (tpda for short) is a quintuple $T=(\Sigma, \Gamma, Q, Q_f, R)$, where Σ is a ranked alphabet of input symbols, Γ is a ranked alphabet of stack symbols, Q is a ranked alphabet of states, $Q_f \subseteq Q_1$ is the set of final states, R is a finite set of rewrite rules over $T_{\Sigma\cup\Gamma\cup Q}$ of the following two types.

(a) *Standard rules*: $\delta(q_1(u_{11}, \ldots, u_{1n_1}), \ldots, q_m(u_{m1}, \ldots, u_{mn_m})) \to q(u_1, \ldots, u_n)$, where $\delta \in \Sigma_m$, $m \geqslant 0$, $q_i \in Q_{n_i}$, $n_i \geqslant 0$ for $i=1, \ldots, m$, $q \in Q_n$, $n \geqslant 0$, u_{ij}, $u_k \in T_\Gamma(X)$, for $i=1, \ldots, m$ and $j=1, \ldots, n_i$, $k=1, \ldots, n$. We refer to $u_{11}, \ldots, u_{mn_m}$ as stack trees appearing in the left-hand side.

(b) *Reduce rules*: $q(u_1, \ldots, u_n) \to q'(u'_1, \ldots, u'_{n'})$, where $q \in Q_n$, $n \geqslant 0$, $q' \in Q_{n'}$, $n' \geqslant 0$, $u_i, u'_j \in T_\Gamma(X)$ for $i=1, \ldots, n$, $j=1, \ldots, n'$. We refer to $u_1, \ldots, u_n$ as stack trees appearing in the left-hand side.

The maximum rank of the states in Q is called the rank of T and denoted by $rank(T)$. The maximum depth of the stack trees in $T_\Gamma(X)$ appearing in the left-hand side of the rules of T is called the depth of T and is denoted by $depth(T)$.

Definition 3.2. Given a tpda T, a *configuration* c of T is a term in $T_{\Sigma\cup\Gamma\cup Q}$ such that $path(c) \subset \Sigma^* Q \Gamma^* \cup \Sigma^*$. The *move relation* $\to_T$ is the rewrite relation $\to_R$ restricted to the set of configurations of T, i.e., $\to_T = \to_R \cap \{(c_1, c_2) \mid c_1, c_2 \text{ are configurations of } T\}$.

The *computation relation* $\to_T^*$ is the reflexive and transitive closure of $\to_T$.

An *initial configuration* c of T is simply a tree in T_Σ.

A *final configuration* c is of the form $c=q(v)$ for some final state $q \in Q_f$ and tree stack $v \in T_\Gamma$.

Intuitively speaking, if $c=t_0(t_1, \ldots, t_m)$, where $t_0 \in \bar{T}_\Sigma(X_m)$ and $t_1, \ldots, t_m \in T_\Sigma$, $c \to_T^* c'$ and $c'=t_0(q_1(u_{11}, \ldots, u_{1n_1}), \ldots, q_m(u_{m1}, \ldots, u_{mn_m}))$, then $t_1, \ldots, t_m$ are scanned and the ith read head is in state q_i with associated tree stacks $u_{i1}, \ldots, u_{in_i} \in T_\Gamma$, $1 \leqslant i \leqslant m$.

One may view a tpda as a transducer that outputs its pushdown. From this point of view, we shall study in Section 4, the connection between trf rewrite systems and tpda's. For a given trf rewrite system S, we shall construct a tpda computing normal forms of ground terms with respect to S. To this end we now define the tree transformation induced by a bottom-up tree pushdown automaton.

Definition 3.3. The *tree transformation induced* by T is the relation

$$\tau_T = \{(u, v) \in T_\Sigma \times T_\Gamma \mid (\exists q \in Q_f) u \xrightarrow[T]{*} q(v)\}.$$

Definition 3.4. We say that tpda's T, T' are *equivalent* if $\tau_T = \tau_{T'}$.

Given a tpda $T=(\Sigma, \Gamma, Q, Q_f, R)$ and a partial order $<$ on R, the rewrite relation with respect to R, denoted by $\rightarrow_{<,R}$, is defined to be $\rightarrow_{i,R,<}$, the innermost rewrite relation induced by R with respect to $<$. The move relation with respect to $<$, $\rightarrow_{<,T}$ is the restriction of $\rightarrow_{<,R}$ to the set of configurations of T. We can note that, with this definition, standard rules are applied after all possible reduce rules have been applied.

Intuitively a tpda T' simulates a tpda T if each computation step of T is simulated by a piece of computation with T'. The simulation is real-time if for each computation step of T the number of necessary simulation steps is bounded by a constant number. We often slightly modify the configurations (adding new symbols, renaming symbols, ...) in order to simulate a tpda, so we have the following formal definition.

Definition 3.5. A tpda T' *real-time simulates* a tpda T if and only if T and T' are equivalent and there is some integer k, some injective mapping h from the configurations of T into the configurations of T' such that

(i) if $c \rightarrow_T c'$ then $h(c) \overset{k'}{\rightarrow}_{T'} h(c')$ with $k' \leqslant k$,

(ii) if c is an initial configuration of T, then $h(c)$ is an initial configuration of T',

(iii) if c is a final configuration of T, then $h(c)$ is a final configuration of T'.

A class C' *real-time simulates* a class C if and only if any tpda T of C is real-time simulated by some tpda T' of C'. If the classes C and C' real-time simulate each other, we say that they are *real-time equivalent*. This equivalence is denoted by $\equiv_{rt}$.

Let tpda_{**}, tpda_{n*}, tpda_{*k} and tpda_{nk} $(n, k \geqslant 0)$ denote, respectively, a tpda of any rank and any depth, of rank n and any depth, of any rank and depth k and of rank n and depth k. Moreover, let $TPDA_{**}$, $TPDA_{n*}$, $TPDA_{*k}$ and $TPDA_{nk}$ denote the classes of all tpda_{**}'s, tpda_{n*}'s, tpda_{*k}'s and tpda_{nk}'s, respectively. Finally, $TPDT_{**}$, $TPDT_{n*}$, $TPDT_{*k}$ and $TPDT_{nk}$ denote the classes of tree transformations induced by tpda_{**}'s, tpda_{n*}'s, tpda_{*k}'s and tpda_{nk}'s, respectively.

Proposition 3.6. *For each* $\text{tpda}_{**}T$, *there exists a* $\text{tpda}_{1*}T'$ *which real-time simulates* T.

Proof. Let $T=(\Sigma, \Gamma, Q, Q_f, R)$ be a tpda_{**}. We construct tpda $T'=(\Sigma, \Gamma', Q', Q_f, R')$ as follows. With each state $q \in Q$ of rank $n \geqslant 2$, we associate a new state $\langle q, n\rangle$ of rank one. In this way we obtain a new set of states Q'_1. Let $Q'=Q_0 \cup Q_1 \cup Q'_1$. The ranked stack alphabet Γ' is defined by adding new symbols $\langle \natural, k\rangle$ of rank k, $2 \leqslant k \leqslant rank(T)$, to the ranked stack alphabet Γ. In other words, $\Gamma' = \Gamma \cup \{\langle \natural, k\rangle \mid 2 \leqslant k \leqslant rank(T)\}$, where the pushdown symbol $\langle \natural, k\rangle$ is of rank k for $k=2, \ldots, rank(T)$. We define the rewrite rule set R' from R by replacing every term $q(u_1, \ldots, u_n)$ (with $n \geqslant 2$, $q \in Q_n$, $u_1, \ldots, u_n \in T_\Gamma(X)$) appearing in a left-hand side or a right-hand side of a rule in R, with the term $\langle q, n\rangle(\langle \natural, n\rangle(u_1, \ldots, u_n))$. It should be clear that T' is a $\text{tpda}_{1(depth(T)+1)}$.

With each configuration $c = t_0(q_1(u_{11}, \ldots, u_{1n_1}), \ldots, q_m(u_{m1}, \ldots, u_{mn_m}))$ of T, we associate configuration $h(c)$ of T', where $h(c)$ is obtained from c by replacing the term $q_i(u_{i1}, \ldots, u_{in_i})$ with the term $\langle q_i, n_i\rangle(\langle \natural, n_i\rangle(u_{i1}, \ldots, u_{in_i}))$ if $n_i \geqslant 2$, for $i = 1, \ldots, n$.

It can easily be seen that for all configurations c_1, c_2 of T and their associated configurations $h(c_1)$ and $h(c_2)$, $c_1 \to_T c_2$ if and only if $h(c_1) \to_{T'} h(c_2)$. It should be clear that we assign itself to each initial configuration t of T, and t is an initial configuration of T' as well. Moreover, we assign itself to each final configuration $q(v)$ of T, and $q(v)$ is a final configuration of T' as well. We deduce $\tau_T = \tau_{T'}$ and one can easily prove that T' real-time simulates T with the mapping h and $k = 1$. □

Choosing the value of $depth(T)$ to be 1 in the proof of Proposition 3.6, we obtain the proof of the following result.

Proposition 3.7. *For each* tpda_{*1} T, *there exists a* tpda_{12} T' *which real-time simulates* T.

Proposition 3.8. *For each* tpda_{1*} T, *there exists a* tpda_{*1} T' *which real-time simulates* T.

Proof. Let $T = (\Sigma, \Gamma, Q, Q_f, R)$ be a tpda_{1*}.

Definition of T'. We construct tpda_{*1} $T' = (\Sigma, \Gamma, Q', Q_f, R')$ as follows: Q is included in Q' and we define the rest of Q' and the rules of R' in the following way. Consider an arbitrary subterm $q(u)$ ($u \in T_\Gamma(X_m)$, $m \geqslant 0$) of the left-hand side of a rule in R with $depth(u) \geqslant 2$ (if $depth(u) \leqslant 1$, there is no modification); we introduce the state q_0^u of Q' and the following rule of R':

$$q(x) \to q_0^u(x) \quad \text{(guessing rule)}.$$

Let k, $0 \leqslant k \leqslant depth(u) + 1$, be an integer and let $u_k \in \bar{T}_\Gamma(X_h)$ ($h \geqslant 0$) be defined by the following requirements:

$$O(u_k) = \{\alpha \in O(u) \mid length(\alpha) \leqslant k\} \text{ and, for every occurrence } \alpha \in O(u_k),$$

$$\text{if } length(\alpha) < k \text{ and } lab(u, \alpha) \in \Gamma, \text{ then } lab(u_k, \alpha) = lab(u, \alpha) \text{ else } lab(u_k, \alpha) \in X.$$

Intuitively, one can consider the tree u_k as the truncation of u at depth k. Let us introduce the states $q_1^u, \ldots, q_{depth(u)+1}^u \in Q$. Moreover, for every k, $0 \leqslant k \leqslant depth(u)$, consider the equation $u_{k+1} = u_k(z_1, \ldots, z_h)$, where $u_{k+1} \in \bar{T}_\Gamma(X_i)$ for some $i \geqslant 0$, $h \geqslant 0$ and for each $j \in \{1, \ldots, h\}$, $z_j \in T_\Gamma(X)$, $depth(z_j) \leqslant 1$. Note that z_j can be a variable, a symbol of rank 0 or a tree of depth 1. Then, we introduce the rule of R'

$$q_k^u(z_1, \ldots, z_h) \to q_{k+1}^u(x_1, \ldots, x_i) \quad \text{(checking rule)}.$$

It should be clear that, for every two different subterms $q(u)$, $q'(u')$, we introduce disjoint sets of new states.

For each reduce rule $q(u)\to q'(u')$ in R, if $depth(u)\leqslant 1$, then we place in R' the rule $q(u)\to q'(u')$ itself; otherwise we place in R' the rule

$$q^u_{depth(u)+1}(x_{k_1},\dots,x_{k_j})\to q'(u'),$$

where $q^u_{depth(u)+1}\in Q'_j$, $x_{k_1}\dots x_{k_j}=fr(u)$.

Moreover, let $\delta(q_1(u_1),\dots,q_m(u_m))\to q'(u')$ $(m\geqslant 0)$ be a standard rule of R. For each $i\in\{1,\dots,m\}$, if $depth(u_i)\geqslant 2$, then replace the subterm $q_i(u_i)$ in $\delta(q_1(u_1),\dots,q_m(u_m))$ with the term $q^{u_i}_{depth(u_i)+1}(x_{k_1},\dots,x_{k_j})$, where $q^{u_i}_{depth(u_i)+1}\in Q'_j$, $x_{k_1}\dots x_{k_j}=fr(u_i)$). In this way we obtain the tree t from $\delta(q_1(u_1),\dots,q_m(u_m))$. We place in R' the rule

$$t\to q'(u').$$

Example. Let us consider the term $q(u)=q(c(x,b(y,b(a,x)),z))$. Now $depth(u)=3$, and

$$u_0=x_1,\qquad u_1=c(x_1,x_2,x_3),\qquad u_2=c(x_1,b(x_2,x_3),x_4),$$

$$u_3=c(x_1,b(x_2,b(x_3,x_4)),x_5),\qquad u_4=c(x_1,b(x_2,b(a,x_3)),x_4).$$

We introduce the states $q^u_0,q^u_1,q^u_2,q^u_3,q^u_4$ and the following rules:

$$q(x)\to q^u_0(x),\qquad q^u_0(c(x_1,x_2,x_3))\to q^u_1(x_1,x_2,x_3),$$

$$q^u_1(x_1,b(x_2,x_3),x_4)\to q^u_2(x_1,x_2,x_3,x_4),$$

$$q^u_2(x_1,x_2,b(x_3,x_4),x_5)\to q^u_3(x_1,x_2,x_3,x_4,x_5),$$

$$q^u_3(x_1,x_2,a,x_3,x_4)\to q^u_4(x_1,x_2,x_3,x_4).$$

Let us consider the reduce rule $q(u)=q(c(x,b(y,b(a,x)),z))\to q'(u')$. Now we introduce the rule

$$q^u_4(x,y,x,z)\to q'(u').$$

Correctness: We now show that $\tau_T\subseteq\tau_{T'}$. Each configuration, starting configuration and final configuration of T is a configuration, a starting configuration and a final configuration of T', respectively. Moreover, if $c\to_T c'$, then $c\to^*_{T'}c'$ holds as well. Thus, $\tau_T\subseteq\tau_{T'}$.

Conversely, we show that $\tau_{T'}\subseteq\tau_T$. Define the mapping $\phi: T_{\Sigma\cup\Gamma\cup Q'}\to T_{\Sigma\cup\Gamma\cup Q}$ in the following way. For each tree $t\in T_{\Sigma\cup\Gamma\cup Q'}$, let the tree $\phi(t)$ be defined from t by replacing each occurrence of the subtree $q^u_k(t_1,\dots,t_m)$ with $q(u_k(t_1,\dots,t_m))$ for each state $q^u_k\in Q'_m-Q$ with $m\geqslant 0$, and for all trees $t_1,\dots,t_m\in T_\Gamma$. It should be clear that for all configurations c_1, c_2 of T', $c_1\to_{T'}c_2$ implies that either $\phi(c_1)=\phi(c_2)$ or $\phi(c_1)\to_T\phi(c_2)$.

Assume that $(t,t')\in\tau_{T'}$ and consider a reduction $t\to^*_{T'}q(t')$, where $t\in T_\Sigma$ is an initial configuration of T', and t' is a final configuration of T', i.e., $q\in Q_f$, $t'\in T_\Gamma$. Now $\phi(t)\to^*_T\phi(q(t'))$ holds as well. Obviously $\phi(t)=t$ is an initial configuration of T and $\phi(q(t'))=q(t')$ is a final configuration of T. Thus, we obtain that $t\to^*_T q(t')$ and $q(t')$ is a final configuration of T. Hence, $(t,t')\in\tau_T$ as well. The above argument implies that

$\tau_{T'} \subseteq \tau_T$. One can also easily prove that T' real-time simulates T with $h = id$ and $k = depth(T) + 3$. □

By Propositions 3.6, 3.7 and 3.8, we have proved that the classes $TPDA_{**}$, $TPDA_{1*}$, $TPDA_{*1}$ and $TPDA_{12}$ are real-time equivalent to each other. The most usual class is $TPDA_{1*}$. From [10] we adopt another class: the class of bottom-up tree pushdown automata with read rules and reduce rules (or tree stack update rules).

Definition 3.9 (*Gallier and Book* [10]). A *bottom-up tree pushdown automaton with read rules and reduce rules* (tpda^{rr} for short) is a tpda$_{1*}$ $T = (\Sigma, \Gamma, Q, Q_f, R)$ such that R is a finite set of rewrite rules over $T_{\Sigma \cup \Gamma \cup Q}$ of the following two types:

(a) *Read rules*: $\delta(q_1(x_1), \ldots, q_k(x_k)) \to q(\gamma(x_1, \ldots, x_k))$, where $k \geqslant 0$, $\delta \in \Sigma_k$, $q \in Q$, $q_i \in Q$ for $i \in \{1, \ldots, k\}$, $\gamma \in \Gamma_k$.

(b) *Reduce rules* (tree stack update rules): $q(u) \to q'(v)$, *where* $q, q' \in Q$, $u, v \in T_\Gamma(X)$.

We denote the class of bottom-up tree pushdown automata with read rules and reduce rules by $TPDA^{rr}$. The class of tree transformations induced by tpda^{rr}'s is denoted by $TPDT^{rr}$. Now we show how to simulate a tpda$_{1*}$ by a tpda^{rr}.

Proposition 3.10. *For each* tpda$_{1*}$ T, *there exists a* tpda^{rr} A *which real-time simulates* T.

Proof. Let $T = (\Sigma, \Gamma, Q, Q_f, R)$ be a tpda$_{1*}$.

If $Q_0 \neq \emptyset$, then first we modify T as follows. For each state q of rank 0, we introduce the state $\bar{q}$ of rank 1 and replace each occurrence of q in the rules of T by the term $\bar{q}$(*emptystack*), where *emptystack* is a new stack symbol of rank 0; then we discard the state q.

We construct the tpda^{rr} $A = (\Sigma, \Gamma', Q', Q_f, R')$ as follows. The rule set R' contains each reduce rule of T. Moreover, for each standard rule r: $\delta(q_1(u_1), \ldots, q_m(u_m)) \to q(u)$ of T, we introduce the state $\langle q, r \rangle$, the symbol $\bar{\delta}$ ($\bar{\delta} \in \Gamma'_m$) and the following rules of R':

$$\delta(q_1(x_1), \ldots, q_m(x_m)) \to \langle q, r \rangle(\bar{\delta}(x_1, \ldots, x_m)) \text{ and } \langle q, r \rangle(\bar{\delta}(u_1, \ldots, u_m)) \to q(u).$$

It should be clear that A real-time simulates T with $h = id$ and $k = 2$. □

Proposition 3.11. $TPDT_{11} \subset TPDT_{12}$.

Proof. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{c\}$, $\Sigma_1 = \{b\}$, $\Sigma_2 = \{a\}$. Consider the relation $\rho = \{(a(b^n(c), b^k(c)), a(b^n(c), b^k(c))) \mid n \geqslant 0, k \geqslant 0, n \neq k\}$ over T_Σ. Thus, ρ is the identity relation on those forks which have two different main branches.

Consider the tpda$_{12}$ $T = (\Sigma, \Gamma, Q, Q_f, R)$, where $\Gamma = \Sigma$, $Q = \{q_1, q_2, q_3\}$, $Q_f = \{q_3\}$, and R consists of the following rules:

$c \to q_1(c)$,

$b(q_1(x)) \to q_1(b(x))$,

$a(q_1(x_1), q_1(x_2)) \to q_2(d(x_1, x_2, x_1, x_2))$,

$q_2(d(b(x_1), b(x_2), x_3, x_4)) \to q_2(x_1, x_2, x_3, x_4)$,
$q_2(d(c, b(x_1), x_3, x_4)) \to q_3(a(x_3, x_4))$,
$q_2(d(b(x_1), c, x_3, x_4)) \to q_3(a(x_3, x_4))$.

It should be clear that $\tau_T = \rho$. Hence, ρ is in $TPDT_{12}$.

We now show that ρ cannot be induced by a tpda_{11}. On the contrary, let us suppose that ρ can be induced by a tpda_{11} A. Let the relation $\eta \subseteq \{(b^n(c), b^m(c)) \mid n, m \geqslant 0\}$ contain all pairs $(b^n(c), b^m(c))$ such that for each tree $q(u)$ appearing in the left-hand side of a standard rule with $q \in Q$, $u \in T_\Gamma(X)$,

$$b^n(c) \xrightarrow[A]{*} \sigma_n(q(u)) \text{ if and only if } b^m(c) \xrightarrow[A]{*} \sigma_m(q(u)),$$

where σ_n and σ_m are substitutions. It should be clear that η is an equivalence relation on the set $\{b^n(c) \mid n \geqslant 0\}$ and η has finitely many classes. Hence, one can provide infinitely many pairs $(b^n(c), b^m(c))$ of trees such that $n \neq m$ and $(b^n(c), b^m(c)) \in \eta$. For each of these pairs consider the derivation

$(*)$ $a(b^n(c), b^m(c)) \to_A^* a(q_1(\sigma_{n1}(u_1)), q_2(\sigma_{m2}(u_2))) \to_A q(\sigma_{mn}(v)) \to_A^* q_f(a(b^n(c), b^m(c)))$,

where $a(q_1(u_1), q_2(u_2)) \to q(v)$ is a rule of A, q_1, q_2, q_f are states of A and q_f is a final state, and $\sigma_{n1}, \sigma_{m2}, \sigma_{mn}$ are substitutions.

Moreover, consider the derivation

$(**)$ $a(b^n(c), b^n(c)) \to_A^* a(q_1(\sigma_{n1}(u_1)), q_2(\sigma_{n2}(u_2)))$, where σ_{n2} is a substitution.

During $(*)$ A must transform $\sigma_{n1}(u_1)$ and $\sigma_{m2}(u_2)$ to the same tree t, and then compare the resulting copies of t by applying a reduce rule with a nonlinear left-hand side along the subderivation $q(\sigma_{mn}(v)) \to_A^* q_f(a(b^n(c), b^m c)))$. Because, otherwise, $(**)$ can be continued in the same way as $(*)$. But after transforming $\sigma_{n1}(u_1)$ and $\sigma_{m2}(u_2)$ to the same form t and comparing the resulting copies of t, A is not able to restore $b^n(c)$ and $b^m(c)$ from t because a state is the topmost symbol of the current configuration, A is a tpda_{11}, and hence does not know the symbols below the topmost pushdown symbol when applying a rule, and n, m are chosen in infinitely many ways. Hence, A cannot output the tree $a(b^n(c), b^m(c))$. □

Now we sum up our results obtained in this chapter.

Theorem 3.12. $TPDA_{12} \equiv_{rt} TPDA_{1*} \equiv_{rt} TPDA_{*1} \equiv_{rt} TPDA_{**} \equiv_{rt} TPDA^{rr}$.

3.2. Deterministic tree pushdown automata

Definition 3.13. A bottom-up tree pushdown automaton $T = (\Sigma, \Gamma, Q, Q_f, R)$ is deterministic (dtpda for short) if the set R of rewrite rules is left-linear and is without critical pairs.

Note that we restrict our definition to left-linear tree pushdown automata, since whenever the set of rewrite rules R is without critical pairs and is left-linear, $\to_R$ is

confluent (see [15, 16]). We can easily prove that the move relation $\rightarrow_T$ is confluent, too.

Let us denote the deterministic versions of tpda_{**}, tpda_{n*}, tpda_{*k}, tpda_{nk} and tpda^{rr} by dtpda_{**}, dtpda_{n*}, dtpda_{*k}, dtpda_{nk} and dtpda^{rr}, respectively. Moreover, the corresponding bottom-up tree pushdown automata classes are denoted by $DTPDA_{**}$, $DTPDA_{n*}$, $DTPDA_{*k}$, $DTPDA_{nk}$ and $DTPDA^{rr}$, and the corresponding tree transformation classes are denoted by $DTPDT_{**}$, $DTPDT_{n*}$, $DTPDT_{*k}$, $DTPDT_{nk}$ and $DTPDT^{rr}$, respectively.

Now we prove the deterministic version of the main result of the previous section, that is to say we show that $DTPDA_{12} \equiv_{rt} DTPDA_{1*} \equiv_{rt} DTPDA_{*1} \equiv_{rt} DTPDA_{**} \equiv_{rt} DTPDA^{rr}$. One can easily verify that the construction of the proofs of Propositions 3.6 and 3.7 preserves determinism.

Thus, we obtain the following two results.

Proposition 3.14. *For each* $\text{dtpda}_{**}\,T$, *there exists a* $\text{dtpda}_{1*}\,T'$ *which real-time simulates* T.

Proposition 3.15. *For each* dtpda_{*1} T, *there exists a* $\text{dtpda}_{12}\,T'$ *which real-time simulates* T.

Definition 3.16. Let T be a dtpda. We say that T has property (U) if for every states q and q' and stack trees $u, v \in T_\Gamma(X)$, if $q(u)$ occurs in the left-hand side of a standard rule and $q'(v)$ occurs in the left-hand side of a rule, then the equivalence $(q=q') \Leftrightarrow (u=v)$ holds.

We note that determinism does not imply property (U). For example, consider two standard rules of $T=(\Sigma, \Gamma, Q, Q_f, R)$ with left-hand sides $b(q(u), q_1(u_1))$ and $b(q_2(u_2), q(v))$ such that $u \neq v$. Even if u and v are unifiable, they do not induce a critical pair for R. Another example is to consider the terms $b(q(u), q_1(u_1))$ and $q(v)$ as a left-hand side of a standard rule and of a reduce rule, respectively. If u and v are distinct and are not unifiable, then they do not induce a critical pair.

For the proof of Proposition 3.19, we need the following lemma.

Lemma 3.17. *Let* T *be a* dtpda_{1*}. *Then, there exists a* $\text{dtpda}_{1*}\,T'$ *such that* T' *real-time simulates* T *and that property* (U) *holds for* T'.

Proof. Let $T=(\Sigma, \Gamma, Q, Q_f, R)$ be a dtpda_{1*}. We construct the $\text{dtpda}_{1*}\,T'=(\Sigma, \Gamma, Q', Q'_f, R')$ as follows. For each state $q \in Q$, let $V(q)$ denote the set of terms u such that $q(u)$ occurs in the left-hand side of a rule in R. Moreover, let

$$k(q) = \max\{depth(u) \mid u \in V(q)\} + 1$$

and let

$$K = \max\{k(q) \mid q \in Q\}.$$

We define the state set Q' and rewrite rule set R' of T' as follows. For each state $q \in Q$ and each K-normal tree $t \in T_\Sigma(X, K)$, we introduce the state $q_t \in Q'$ and place in R' the rule $q(t) \to q_t(t)$. Note that we introduce disjoint sets of states for distinct states of Q.

For each reduce rule $q(u) \to q'(u')$ of R, we place in R' the rule $q_t(t) \to q'(t')$, where $t, t' \in T_\Sigma(X, K)$ and there is a substitution σ such that $\sigma(u) = t$ and $\sigma(u') = t'$.

For each standard rule $\delta(q^1(u_1), \ldots, q^m(u_m)) \to q(u)$ of R, we place in R' the rule $\delta(q^1_{t_1}(t_1), \ldots, q^m_{t_m}(t_m)) \to q(t)$, where for each $i = 1, \ldots, m$, $t_i \in T_\Sigma(X, K)$ and there is a substitution σ such that $\sigma(u) = t$ and $\sigma(u_i) = t_i$ for $i = 1, \ldots, m$.

Thus, we simulate a reduce rule $q(u) \to q'(u')$ of R by the rules $q(t) \to q_t(t)$ and $q_t(t) \to q'(t')$ of R', where $t \in T_\Sigma(X, K)$, and there is a substitution σ such that $\sigma(u) = t$ and $\sigma(u') = t'$. Moreover, we simulate a standard rule $\delta(q^1(u_1), \ldots, q^m(u_m)) \to q(u)$ of R by the rules $q^i(t_i) \to q^i_{t_i}(t_i)$ for $i = 1, \ldots, m$ and the rule $\delta(q^1_{t_1}(t_1), \ldots, q^m_{t_m}(t_m)) \to q(t)$ of R', where for each $i = 1, \ldots, m$, $t_i \in T_\Sigma(X, K)$, and there is a substitution σ such that $\sigma(u) = t$ and $\sigma(u_i) = t_i$ for $i = 1, \ldots, m$. In this way we obtain a dtpda_{1*} T' with property (U). Moreover, one can easily verify that T' real-time simulates T. □

Proposition 3.18. *For each* dtpda_{1*} T, *there exists a* dtpda_{*1} T' *which real-time simulates* T.

Proof. Let $T = (\Sigma, \Gamma, Q, Q_f, R)$ be a dtpda_{1*}. We construct the dtpda_{*1} $T' = (\Sigma, \Gamma, Q', Q_f, R')$ as follows. For each state $q \in Q$, let $V(q)$ denote the set of terms u such that $q(u)$ occurs in the left-hand side of a rule in R. Moreover, let

$$k(q) = \max\{depth(u) \mid u \in V(q)\} + 1$$

and let

$$K = \max\{k(q) \mid q \in Q\}.$$

We define the state set Q' by

$$Q' = Q \cup \{q^u \mid q \in Q,\ 1 \leqslant k \leqslant K,\ u \in T_\Sigma(X, k)\}.$$

Furthermore, we define the rewrite rule set R' as follows. If $u \in T_\Sigma(X_n, k)$, $v \in T_\Sigma(X_m, k+1)$, $n, m \geqslant 0$, $1 \leqslant k < K$ and $v = u(z_1, \ldots, z_n)$ $(z_j \in T_\Gamma(X)$, $depth(z_j) \leqslant 1$ for $j \in \{1, \ldots, n\})$, then we place in R' the rule $q^u(z_1, \ldots, z_n) \to q^v(x_1, \ldots, x_m)$.

For each reduce rule $q(u) \to q'(v)$ in R, we place in R' the rule $q^t(x_1, \ldots, x_k) \to q'(t')$, where $t \in T_\Sigma(X, K)$, and there is a substitution σ such that $\sigma(u) = t$ and $\sigma(v) = t'$.

Moreover, for each standard rule $\delta(q_1(u_1), \ldots, q_m(u_m)) \to q(u)$ of R, define the tree v from $\delta(q_1(u_1), \ldots, q_m(u_m))$ by replacing, for each $i \in \{1, \ldots, m\}$, the subterm $q_i(u_i)$ with the subterm $q_i^{t_i}(x_{k_1}, \ldots, x_{k_j})$, where $t_i \in T_\Sigma(X_j, K)$, t_i matches u_i, $j \geqslant 0$, $x_{k_1} \ldots x_{k_j} = fr(u_i)$, $q_i^{t_i} \in Q'_j$. Consider a substitution σ such that $\sigma(u_i) = t_i$ for $i = 1, \ldots, m$, and let $t = \sigma(u)$.

Then we place in R' the rule

$$v \to q(t).$$

One can verify that T' real-time simulates T. □.

We now show how to simulate a dtpda_{1*} by a dtpda^{rr}.

Proposition 3.19. *For each* dtpda_{1*} T, *there exists a* dtpda^{rr} T' *which real-time simulates* T.

Proof. Let $T=(\Sigma, \Gamma, Q, Q_f, R)$ be a dtpda_{1*} with property (U). Using the construction of Proposition 3.10, we construct a tpda^{rr} $T'=(\Sigma', \Gamma', Q', Q'_f, R')$ such that T' real-time simulates T. It should be clear that T' is linear and there is no critical pair between any two reduce rules of T'.

Furthermore, let us consider an arbitrary read rule $a(q_1(x_1), \ldots, q_m(x_m)) \to \langle q, r\rangle(\bar{\delta}(x_1, \ldots, x_m))$ in R', which is obtained from the rule $r: a(q_1(u_1), \ldots, q_m(u_m)) \to q(u)$ of R, and consider an arbitrary reduce rule $q(v) \to q(v')$ in $R(\subseteq R')$, and an arbitrary integer $i \in \{1, \ldots, m\}$, then $q(v)$ and $q_i(v_i)$ are not unifiable because T is deterministic. Hence, either $q \neq q_i$ or ($q = q_i$ and v and v_i are not unifiable). Since T has property (U), the second case is impossible, and hence there is no critical pair between the read rule $a(q_1(x_1), \ldots, q_m(x_m)) \to \langle q, r\rangle(\bar{\delta}(x_1, \ldots, x_m))$ in R' and the reduce rule $q(v) \to q(v') \in R(\subseteq R')$.

Moreover, it is not hard to see that there is no critical pair between an arbitrary read rule $a(q_1(x_1), \ldots, q_m(x_m)) \to \langle q, r\rangle(\bar{\delta}(x_1, \ldots, x_m))$ and an arbitrary reduce rule of the form $\langle q, r\rangle(\bar{\delta}(x_1, \ldots, x_m)) \to q(u)$. Thus, there is no critical pair between a read rule and a reduce rule of R'.

One can show in a similar way that there is no critical pair between two read rules of T'. It should be clear that T' is deterministic, and by the proof of the Proposition 3.10, T' real-time simulates T. □

Proposition 3.20. $DTPDT_{11} \subset DTPDT_{12}$.

Proof. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{c\}$, $\Sigma_1 = \{b\}$, $\Sigma_2 = \{a\}$. Consider the relation $\rho = \{(a(b^n(c), b^n(c)), c) \mid n \geq 1\}$ over Σ.

Let $T=(\Sigma, \Gamma, Q, Q_f, R)$, where $\Gamma = \Sigma, Q = Q_1 = \{q_1, q_2, q_3\}$, $Q_f = \{q_3\}$, and R contains the following five rules:

$c \to q_1(c)$,
$b(q_1(x)) \to q_1(b(x))$,
$a(q_1(x), q_1(y)) \to q_2(a(x, y))$,
$q_2(a(b(x), b(y))) \to q_2(a(x, y))$,
$q_2(a(c, c)) \to q_3(c)$.

It should be clear that T is deterministic and that $\tau_T = \rho$. Hence, ρ is in $DTPDT_{12}$. Let $T'=(\Sigma, \Gamma, Q', Q'_f, R')$, where $\Gamma = \Sigma$, $Q' = Q_1 = \{q_1, q_2, q_3\}$, $Q'_f = \{q_3\}$, and R' contains the following four rules:

$c \to q_1(c)$,
$b(q_1(x)) \to q_1(b(x))$,

$a(q_1(x), q_1(y)) \to q_2(a(x, y))$,
$q_2(a(x, x)) \to q_3(c)$.

It should be clear that $\tau_{T'} = \rho$. Hence, ρ is in $TPDT_{11}$. However, it is not hard to see that ρ is not in $DTPDT_{11}$. □

Now we sum up our results obtained in this chapter.

Theorem 3.21. $DTPDA_{12} \equiv_{rt} DTPDA_{1*} \equiv_{rt} DTPDA_{*1} \equiv_{rt} DTPDA_{**} \equiv_{rt} DTPDA^{rr}$.

In view of the results of Section 4.4, we conclude this section with the following result.

Remark. For every tpda^{rr} A and order $<$ such that τ_A is a (partial) function there exists a dtpda^{rr} B such that $\tau_B = \tau_A$. The main points of the tedious construction are the following:

(0) We traverse the input tree and put it in the stack.

(i) Every move through a term can be simulated by a sequence of reductions using reduce rules shifting parts of the term in the stack tree.

(ii) We can simulate every move of the tpda^{rr} A by a sequence of moves on the stack tree. In the case of non left-linear rewrite rules, the equalities can be tested by a sequence of moves.

(iii) We simulate all the possible moves of the tpda^{rr} A according to an arbitrary but deterministic and fair strategy.

(iv) If some move in the dtpda^{rr} B terminates with a final configuration we are done.

Moreover, we notice that this intricate construction is untractable in the precise complexity meaning and the intuitive one.

4. Tree pushdown automata and rewrite systems

In Section 4.1, we introduce the concept of a tail reduction free (trf for short) rewrite system. We prove the decidability of trf property by reducing it to the decidability of ground reducibility (cf. Theorem 4.11). We give several examples which illustrate the fact that this class is fairly powerful. We show that monadic rewrite systems, introduced by Gallier and Book [10] and Salomaa [19], are a particular case of trf rewrite systems. In Section 4.2, we also define the concept of a semi-monadic rewrite system, which is a generalization of the concept of a monadic rewrite system.

Theorem 4.13 assigns a tpda^{rr} *with reduce priority*, computing the normal forms of ground terms, to each convergent trf rewrite system. A tpda with reduce priority is a tpda which applies reduce rules before read rules, i.e., it applies a read-rule rewriting the subtree $\delta(q_1(u_1), \ldots, q_n(u_n))$ into the tree $q(u)$ only when no reduce rule is applicable for subtrees $q_i(u_i)$, $i \in \{1, \ldots, n\}$. We can note that reduce priority is a feature of innermost rewriting.

In the left-linear case we do not need the notion of priority. Theorem 4.15 associates with each convergent left-linear term rewriting system a deterministic tpda computing normal forms of ground terms.

4.1. Tail-reduction-free rewrite systems

We introduce the notion of a tail-reduction-free rewrite system with the next two examples.

Example 4.1. Let $\Sigma=\{a,b,c,d\}$ where a,b,c are of rank 1 and d is of rank 0. Consider the rewrite system $S=\{a(c(b(x)))\to c(x)\}$ and the tpda^{rr} $A=(\Sigma,\Sigma,\{q\},\{q\},R)$, where R consists of the following rules:

$$d\to q(d),\quad a(q(x))\to q(a(x)),\quad b(q(x))\to q(b(x)),\quad c(q(x))\to q(c(x)),$$
$$q(a(c(b(x))))\to q(c(x)).$$

Let $u=a(a(a(c(b(b(d))))))$. Then we have

$$u\underset{A}{\overset{*}{\to}} q(u),\quad u\underset{A}{\overset{*}{\to}} q(a(a(c(b(d))))))\ \text{and}\ u\underset{A}{\overset{*}{\to}} q(a(c(d))).$$

Thus, if A is given the input tree u, then A outputs not only the normal form $a(c(d))$ of u with respect to S. For any partial order $<$ on R, we have $u\to^*_{<,A} q(a(c(d)))$. Thus, in this case tpda^{rr} A with reduce priority computes the normal form of u.

Example 4.2. Let the ranked alphabet Σ be the same as in Example 4.1. Consider the rewrite system $S=\{a(b(x))\to b(a(x))\}$ and the tpda^{rr} $A=(\Sigma,\Sigma,\{q\},\{q\},R)$, where R consists of the following rules:

$$d\to q(d),\quad a(q(x))\to q(a(x)),\quad b(q(x))\to q(b(x)),\quad q(a(b(x)))\to q(b(a(x))).$$

Then for any order $<$ and for $u=a(a(b(b(d))))$, we have

$$u\underset{<,A}{\overset{*}{\longrightarrow}} q(b(a(a(b(d)))))$$

and $b(a(a(b(d))))$ is not a normal form of u with respect to S. Hence, in this case we cannot compute the normal forms of ground terms with respect to S using a tpda^{rr} with reduce priority.

Definition 4.3. Let S be a rewrite system over some ranked alphabet Σ with $IRR(S)\neq\emptyset$. We say that S has the trf property if for each rule $l\to r$ in S with $r=\delta(r_1,\ldots,r_n)$, $\delta\in\Sigma_n$ $(n\geqslant 1)$, $r_1,\ldots,r_n\in T_\Sigma(X)$, for each irreducible, ground substitution σ, and for each i, $1\leqslant i\leqslant n$, the term $\sigma(r_i)$ is irreducible.

Note that, by Proposition 2.2 the condition $IRR(S)\neq\emptyset$ implies that no left-hand side of a rule is a variable. The rewrite system of Example 4.1 has the trf property. On the other hand, the rewrite system of Example 4.2 has not the trf property. For example, consider the ground substitution $\sigma(x)=b(d)$. Then $\sigma(a(x))=a(b(d))$ is reducible.

4.2. Examples of trf rewrite systems

We adopt the concept of a monadic rewrite system from Gallier and Book [10] and Salomaa [19].

Definition 4.4. A rewrite system S is monadic if, for every rule $(l,r)\in S$, $depth(l)\geqslant 1$ and $depth(r)\leqslant 1$.

Example. The rewrite system S of Example 4.1 is monadic, the rewrite system S of Example 4.2 is not.

The definitions of the trf property and the monadic rewrite system imply the following result.

Proposition 4.5. *Each monadic rewrite system has the trf property.*

Now we introduce the semi-monadic rewrite system which is a generalization of both the monadic rewrite system and the ground rewrite system.

Definition 4.6. A rewrite system S over some ranked alphabet Σ is semi-monadic if, for every rule $l\to r$ in S, $depth(l)\geqslant 1$ and either $depth(r)=0$, or $r=\delta(y_1,\ldots,y_k)$, where $\delta\in\Sigma_k$, $k\geqslant 1$, and for each $i\in\{1,\ldots,k\}$, either y_i is a variable (i.e., $y_i\in X$) or y_i is a ground term (i.e., $y_i\in T_\Sigma$).

A semi-monadic rewrite system is trf if and only if every ground term y_i in a right-hand side $r=\delta(y_1,\ldots,y_k)$ is irreducible. We now show that for each convergent semi-monadic rewrite system S, there exists a convergent trf semi-monadic rewrite system S' such that each ground term can be reduced to the same normal form by S as by S'.

Proposition 4.7. *Let S be a convergent semi-monadic rewrite system. Then there exists a convergent semi-monadic rewrite system S' such that S' has the trf property and for all ground terms $u,v\in T_\Sigma$,*

$$\left(u \underset{S}{\overset{*}{\to}} v \text{ and } v\in IRR(S)\right)\Leftrightarrow\left(u \underset{S'}{\overset{*}{\to}} v \text{ and } v\in IRR(S')\right).$$

Proof. We define S' from S by replacing every ground subterm in the right-hand side of each rule with its normal form for S. □

Definition 4.8. A rewrite system S, over some ranked alphabet Σ is head separating if there is a partition (Σ',Σ'') of Σ such that, for every rule $l\to r$ in S, $Head(l)$, $Head(r)$ are in Σ' and all other symbols are in Σ''.

The following result is obvious.

Proposition 4.9. *Each head separating rewrite system S with $IRR(S)\neq\emptyset$ has the trf property.*

We now prove that we can simulate any rewrite system with a trf rewrite system over an enlarged alphabet. The idea is as follows: we simulate the traverse of the tree by means of a well-known method of reversing links that have been traversed. Thus, any node x can be moved to the head, and the path from the head to x is reversed. Note that the simulation is not a real-time simulation.

Theorem 4.10. *For every rewrite system S over some ranked alphabet Σ there exists a trf rewrite system S' over some enlarged alphabet Σ' such that*

$$\forall t,u\in T_\Sigma,\quad \left(f(t,a)\underset{S'}{\overset{*}{\rightarrow}}f(u,a)\right)\Leftrightarrow\left(t\underset{S}{\overset{*}{\rightarrow}}u\right)$$

where f is a new binary symbol and a is a new constant.

Proof. To every $b\in\Sigma_n$ and $i\in\{1,\ldots,n\}$, we assign a new symbol b_i in Σ'_n. We define the rules of S' as follows:

Redex selection rules:

$$\forall n\geqslant 1,\ \forall b\in\Sigma_n,\ \forall i\in\{1,,\ldots,n\},$$
$$f(b(x_1,\ldots,x_n),y)\rightarrow f(x_i,b_i(x_1,\ldots,x_{i-1},y,x_{i+1},\ldots,x_n))),$$

Reduce rules:

$$\forall l\rightarrow r\in S,\quad f(l,x)\rightarrow f(r,x),$$

where x is a variable which does not occur in the rule $l\rightarrow r$,

Restoration rules:

$$l\rightarrow r\in S' \quad \text{whenever } r\rightarrow l \text{ is a redex selection rule.}$$

S' is a head-separating rewrite system, and hence by Proposition 4.9 is trf.

Let us consider a reduction $t\rightarrow_S t'$ with $t=c(u)$, $t'=c(v)$, $c\in\bar{T}_\Sigma(X_1)$, $u=\sigma(l)$, $v=\sigma(r)$, $l\rightarrow r\in S$.

The simulation is as follows:

$$f(c(u),a)\underset{S'}{\overset{*}{\rightarrow}}f(u,\tilde{c}(a)) \quad \text{using redex selection rules,}$$

where $\tilde{c}$ denotes c with reversed links along the path leading to the redex position,

$$f(u,\tilde{c}(a))\underset{S'}{\rightarrow}f(v,\tilde{c}(a)) \quad \text{using a reduce rule,}$$

$$f(v,\tilde{c}(a))\underset{S'}{\overset{*}{\rightarrow}}f(c(v),a) \quad \text{using restoration rules.}$$

The proof is now straightforward. □

4.3. The decidability of the trf property

Let Σ be a ranked alphabet and let S be a rewrite system over Σ.

A *term* t in $T_\Sigma(X)$ is *ground-reducible* for S if each of its ground instances in T_Σ is reducible with respect to S, that is, if for each ground substitution $\sigma: X \to T_\Sigma$ with $Dom(\sigma) = var(t)$, $\sigma(t)$ is reducible with respect to S.

We say that a *substitution* $\sigma: X \to T_\Sigma(X)$ with $Dom(\sigma) = \{x_{i_1}, \ldots, x_{i_n}\}$ $(n \geqslant 1, i_1, \ldots, i_n \geqslant 1)$ is *ground-reducible* for S if $\$(\sigma(x_{i_1}), \ldots, \sigma(x_{i_n}))$ is ground-reducible for S with $\$ \notin \Sigma$ being a new symbol of rank n $(n \geqslant 1)$. In other words, σ is ground-reducible for S if for each ground substitution σ', there exists an x_{i_j} $(j \in \{1, \ldots, n\})$ such that $\sigma'(\sigma(x_{i_j}))$ is reducible with respect to S.

Theorem 4.11. *Let S be a rewrite system over some ranked alphabet Σ. Then it is decidable whether or not S has the trf property.*

Proof. First we decide whether or not $IRR(S) = \emptyset$ (cf. Proposition 2.2). If $IRR(S) = \emptyset$, then S is not trf. If $IRR(S) \neq \emptyset$, then we shall show that it is decidable whether or not S is trf by reducing this problem to the decidability of the predicate $P(t)$: for each irreducible, ground substitution σ, the tree $\sigma(t)$ is irreducible with respect to S.

$P(t)$ is true if and only if for each ground-irreducible substitution σ, the tree $\sigma(t)$ is irreducible with respect to S. Assume that $var(t) = \{x_1, \ldots, x_m\}$ and $var(t)$ is disjoint from the set of variables of S. Let $\{\tau_1(t), \ldots, \tau_n(t)\}$ be the results of all possible superpositions of the left-hand sides of S into t (that is every τ_i is a most general unifier of some $s \in S$ and a subterm of t). Assume $\tau_i = \langle x_1 \leftarrow t_{i_1}, \ldots, x_m \leftarrow t_{i_m} \rangle$ for every i, $0 \leqslant i \leqslant n$. Obviously, if σ is an irreducible substitution, then $\sigma(t)$ is reducible if and only if $\sigma = \tau_i \circ \rho$ for some i, $0 \leqslant i \leqslant n$ and substitution ρ. Then $P(t)$ is true if and only if for any ρ, $\tau_i \circ \rho$ is reducible i.e. the tuple $(t_{i_1}, \ldots, t_{i_m})$ is ground-reducible. Since ground reducibility is decidable (cf. [18]), we obtain that the predicate $P(t)$ is decidable as well. Finally, we observe that the rewrite system S has the trf property if and only if for each rule $l \to r$ in S with $r = \delta(r_1, \ldots, r_n)$, $\delta \in \Sigma_n$, $n \geqslant 1$ and for each $i \in \{1, \ldots, n\}$, $P(r_i)$ holds. □

4.4. Trf rewrite systems and tpda

In this section, we show that for every convergent trf rewrite system S, one can construct a bottom-up tree pushdown automaton A with reduce priority such that for an arbitrary ground term, A computes (in real time) its normal form. Furthermore, to any left-linear convergent trf rewrite system S, we assign a deterministic tree pushdown automaton computing (in real time) the normal forms of ground terms.

Proposition 4.12. *For every trf rewrite system S over a ranked alphabet Σ, there exist a* tpda^{rr} $A = (\Sigma, \Sigma, Q, Q_f, R)$ *and a partial order $<$ on R such that for every $t, t' \in T_\Sigma$ the*

equivalence

(*) $(t \to^{n}_{i,S} t' \text{ and } (t' \in IRR(S))) \Leftrightarrow (\exists q \in Q_f)(t \to^{n+|t|}_{<,A} q(t'))$ *holds.*

Proof. First we define the tpda^{rr} A and the partial order $<$. Let $A=(\Sigma, \Sigma, \{q, q'\}, \{q'\}, R)$, where R is the set of rules of the following three types:

(i) $q(l) \to q(r)$, for each rule $l \to r$ in S,

(ii) $q(x) \to q'(x)$,

(iii) $\delta(q'(x_1), \ldots, q'(x_k)) \to q(\delta(x_1, \ldots, x_k))$ for each $\delta \in \Sigma_k$, $k \geqslant 0$.

Let the partial order $<$ on R be defined as follows. For each rule $q(l) \to q(r)$,

$$(q(x) \to q'(x)) < (q(l) \to q(r)).$$

Roughly speaking, the move relation $\to_{<,A}$ with respect to $<$ is defined such that if A applies a read rule, then all the corresponding states plus tree stacks are irreducible for the rules (i) and (ii), hence the tree stacks are irreducible for S. In fact the actual order on R is only needed to ensure that the tpda ends with an irreducible tree stack because reduce priority is already forced by innermost rewriting.

We now show that the left-hand side of (*) implies its right-hand side. For this, we prove the following implication:

(**) For every $n \geqslant 0$, $t_0 \in T_\Sigma(X_n)$, $t_1, \ldots, t_n$, $u_1, \ldots, u_n \in T_\Sigma$, if

(1) $t = t_0(t_1, \ldots, t_n) \to^*_{i,S} u = t_0(u_1, \ldots, u_n)$, and

(2) for each $i \in \{1, \ldots, n\}$, the derivation $t_i \to^*_{i,S} u_i$ holds such that the head of t_i is rewritten during the subderivation $t_i \to^*_{i,S} u_i$, then

(a) if u_i $(i = 1, \ldots, n)$ is reducible for S and $u_i = \delta(v_1, \ldots, v_k)$ for some $k \geqslant 1$, $\delta \in \Sigma_k$ and $v_1, \ldots, v_k \in T_\Sigma$, then for each $j \in \{1, \ldots, k\}$, v_j is irreducible for S, and there exists a configuration c of A such that

(b) $t \to^*_{<,A} c = t_0(q_1(u_1), \ldots, q_n(u_n))$,

(c) if u_i is irreducible for S, then $q_i = q'$ for $i \in \{1, \ldots, n\}$,

(d) if u_i is reducible for S, then $q_i = q$ for $i \in \{1, \ldots, n\}$.

The induction proof on the length of the derivation $t \to^*_{i,S} u$ is left to the reader. It is not hard to prove, using (**), that the left-hand side of (*) implies its right-hand side. The trf property is needed to prove (a).

Conversely, we now show that the right-hand side of (*) implies its left-hand side. To this end, we prove the following statement:

(***) For every $n \geqslant 0$, $t_0 \in T_\Sigma(X_n)$, $t_1, \ldots, t_n$, $u_1, \ldots, u_n \in T_\Sigma$, $q_1, \ldots, q_n \in \{q, q'\}$, if $t = t_0(t_1, \ldots, t_n) \to^*_{<,A} c = t_0(q_1(u_1), \ldots, q_n(u_n))$, then

(e) for every $i \in \{1, \ldots, n\}$, if $q_i = q'$, then u_i is irreducible for S,

(f) for each $i \in \{1, \ldots, n\}$, if $q_i = q$ and $u_i = \delta(r_1, \ldots, r_k)$ for some $k \geqslant 1$, $\delta \in \Sigma_k$ and $r_1, \ldots, r_k \in T_\Sigma$, then for each $j \in \{1, \ldots, k\}$, r_j is irreducible for S, and

(g) $t \to^*_{i,S} t_0(u_1, \ldots, u_n)$.

The induction proof on the length of the computation $t \to^*_{<,A} c$ is left to the reader. One can easily show, using (***), that the right-hand side of (*) implies its left-hand side. The trf property is needed to prove (f).

Moreover, the length of the computation of t in $q(t')$ with the tpda^{rr} A is bounded by the maximal length of derivations of t in t' with S plus $|t|$ because applications of reduce rules are in one-to-one correspondence with rewriting steps and we use $|t|$ to traverse t from the frontier up to the root. □

For convergent rewrite systems one obtains the following result.

Theorem 4.13. *For every convergent trf rewrite system S, there exists a* tdpa^{rr} $A=(\Sigma, \Sigma, Q, Q_f, R)$ *and a partial order* $<$ *on* R *such that for every* $t, t' \in T_\Sigma$,

$$\left(t \underset{S}{\overset{*}{\rightarrow}} t' \text{ and } t' \in \mathrm{IRR}(S)\right) \Leftrightarrow (\exists q \in Q_f)\left(t \underset{<,A}{\overset{*}{\rightarrow}} q(t')\right),$$

moreover, the length of the computation of t in $q(t')$ with the tpda^{rr} A *is bounded by the maximal length of derivations of t in t' with S plus the number of symbols in t.*

Proof. As S is convergent, for any irreducible term t', we have $(t \rightarrow_S^* t') \Leftrightarrow (t \rightarrow_{i,S}^* t')$.

Proposition 4.14. *For every left-linear trf rewrite system S and for each linear order* $<$ *on* S, *there exists a* dtpda^{rr} B *such that for every* $t, t' \in T_\Sigma$, $(t \rightarrow_{i,S,<}^n t'$ *and* $(t' \in \mathrm{IRR}(S))) \Leftrightarrow (\exists q \in Q_f)(t \rightarrow_B^{n+|t|} q(t'))$ *holds.*

Proof. Let S be a left-linear trf rewrite system, and let

$$k = \max\{depth(l) \mid (l \rightarrow r) \in S\} + 1.$$

Let $B=(\Sigma, \Sigma, Q, Q_f, R)$ where $Q=\{q_t \mid t \in T_\Sigma(X,k)\} \cup \{q\}$, $Q_f=\{q_t \mid t$ does not match any left-hand side of a rule in $S\}$ and the set of rules R is defined as follows:

(1) $\delta(q_1(x_1), \ldots, q_n(x_n)) \rightarrow q(\delta(x_1, \ldots, x_n))$, for every $n \geqslant 0$, $\delta \in \Sigma_n$ and for every $i \in \{1, \ldots, n\}$, $q_i \in Q_f$,

(2) $q(t) \rightarrow q_t(t)$, for each $t \in T_\Sigma(X,k)$,

(3) $q_t(l) \rightarrow q(r)$, for each $q_t \notin (Q_f \cup \{q\})$, where $l \rightarrow r$ is the greatest rule with respect to $<$ such that t matches l.

It is obvious from Proposition 2.1 that B is deterministic. One can show the equivalence of the proposition in a similar way as (*) in Proposition 4.12. □

Note that in the non-left-linear case, we cannot use a finite set of states (or terms) to express that a term is irreducible because we have to check the equality of arbitrarily deep subterms.

For a convergent rewrite system, each ground term has exactly one normal form, hence one obtains the following result.

Theorem 4.15. *For every convergent left-linear trf rewrite system S, there exists a* dtpda^{rr} $B=(\Sigma, \Sigma, Q, Q_f, R)$ *such that for every* $t, t' \in T_\Sigma$,

$$\left(t \underset{S}{\overset{*}{\rightarrow}} t' \text{ and } t' \in IRR(S) \right) \Leftrightarrow (t, t') \in \tau_B,$$

moreover, the length of the computation of t in $q(t')$ with the dtpda^{rr} B *is bounded by the maximal length of derivations of t in t' with S plus the number of symbols in t.*

We recall that any rewrite system can be simulated with a trf rewrite system but the simulation is not a real-time simulation. Hence, we can obtain results analogous to Theorems 4.13 and 4.15 for rewrite systems but without bound on the length of the computation.

5. Tree languages, pushdown automata and rewrite systems

In this section we study the connections between tree languages, tree pushdown automata and rewrite systems.

5.1. Semi-monadic rewrite systems

In this section we show that recognizability is preserved by linear semi-monadic rewrite systems. This result embeds both Salomaa's result for monadic rewrite systems [19] and the known result that ground rewrite systems preserve recognizability [2].

Let S be a rewrite system over some ranked alphabet Σ. The set of reductions of a term $t \in T_\Sigma$ is denoted by $S^*(t) = \{t' \in T_\Sigma \mid t \rightarrow_S^* t'\}$.

Furthermore, if $L \subseteq T_\Sigma$, then $S^*(L) = \{t' \in T_\Sigma \mid (\exists t \in L)\, t' \in S^*(t)\}$.

Theorem 5.1. *Let S be a linear semi-monadic rewrite system and L be a recognizable tree language. Then $S^*(L)$ is recognizable.*

Proof. Let $A=(\Sigma, Q, Q_f, R)$ be a connected bottom-up tree automaton such that $L(A)=L$. We lose no generality by assuming that for each rule $l \rightarrow r$ in S, $l \in \bar{T}_\Sigma(X_k)$ for some $k \geqslant 0$. Let SUB be the set of all ground subterms of the right-hand sides of the rules in R, and let the connected deterministic bottom-up tree automaton $A'=(\Sigma, Q', Q'_f, R')$ be defined as follows:

$$Q' = Q'_f = \{q_p \mid p \in SUB\},$$

$$R' = \{\delta(q_{p_1}, \ldots, q_{p_n}) \rightarrow q_p \mid \delta \in \Sigma_n,\ n \geqslant 0,\ p = \delta(p_1, \ldots, p_n)\}.$$

It is not hard to see that $L(A')=SUB$ and that, for every t in SUB and $q' \in Q'$, $t \rightarrow_{A'}^* q'$ if and only if $q' = q_t$.

We now define a (nondeterministic) automaton B such that $L(B)=S^*(L)$:

- The set of states of B is $Q\cup Q'$.
- The set of final states of B is Q_f.
- The rule set of B is the smallest set satisfying conditions (1)–(3).

(1) Each rule in $R\cup R'$ is a rule of B,

(2) $(l(x_1,\ldots,x_k)\to\delta(r_1,\ldots,r_{k'})\in S$, and $l(s_1,\ldots,s_k)\to_B^* s$, where $l\in\bar{T}_\Sigma(X_k)$, $s,s_1,\ldots,s_k$ are states of B, $\delta\in\Sigma_{k'}$)

$\Rightarrow$

$(\delta(s'_1,\ldots,s'_{k'})\to s$ is a rule of B with $s'_j=q_t$ if $r_j=t\in T_\Sigma$ and $s'_j=s_i$ if $r_j=x_i$ for $j\in\{1,\ldots,k'\}$),

(3) $(l(x_1,\ldots,x_k)\to x_j\in S$, and $l(s_1,\ldots,s_k)\to_B^* s$, where $l\in\bar{T}_\Sigma(X_k)$, $s,s_1,\ldots,s_k$ are states of B, $\delta\in\Sigma_{k'}$)

$\Rightarrow$

$(s_j\to s$ is a rule of $B)$.

Note that inference rules (2) and (3) induce rules which mimic, on the right-hand side of a rule of S, the moves of A on the corresponding left-hand side. An inference rule of type (3) corresponds to a collapsing rule of S and yields an ε-rule of B. Any algorithm using these inference rules with a fair control halts and builds the rule set $\bar{R}$ of B via a sequence of rule sets

$$R_0=R\cup R', R_1,\ldots,R_{m-1},R_m=\bar{R},$$

where $m\geqslant 0$ and for each i, $0\leqslant i\leqslant m-1$, R_{i+1} is obtained by adding exactly one rule to R_i.

We obviously get $S^*(L)\subseteq L(B)$ from the following observation. If $t\to_S t'$ and there exists an accepting computation of B on t, then there exists an accepting computation on t' as well. Note that here we need the information that A and A' are connected.

Let us prove the converse inclusion. For each $0\leqslant i\leqslant m$, let us take the tree automaton $B_i=(\Sigma, Q\cup Q', Q_f, R_i)$. We need the following two results.

Lemma 5.2. *For each integer i, $0\leqslant i\leqslant m$, for each state $q_p\in Q'$, and for each tree $p'\in T_\Sigma$, if $p'\to_{B_i}^* q_p$, then $p'\in S^*(p)$.*

Proof. We proceed by an induction on i. Let us assume that $i=0$. By the definition of B_0, the computation $p'\to_{B_0}^* q_p$ implies that $p'=p$.

Let us assume that the statement has been proved for i, $0\leqslant i\leqslant m-1$, and let us consider the rule $r\in R_{i+1}-R_i$, where r corresponds to the rule $\bar{r}$ of S. If B_{i+1} applies the rule r, $n(\geqslant 1)$ times along the derivation $p'\to_{B_{i+1}}^* q_p$, then consider an innermost (i.e., downmost) application of r at occurrence α and define t from p' by replacing $str(p',\alpha)$ by a suitable instance of the left-hand side of $\bar{r}$. It is not hard to see that t can be chosen such that along some computation $t\to_{B_{i+1}}^* q_p$, B_{i+1} applies the rule r, $n-1$ times and that t reduces to p' by (i) applying the rule $\bar{r}$ of S and (ii) if the right-hand side of $\bar{r}$ contains a ground subterm, then by reducing each ground subterm $u\in SUB$,

appearing on the right-hand side of $\bar{r}$ to the corresponding subtree v of p' which was evaluated by B_{i+1} applying only rules in R_i into the state q_u, see induction hypothesis. Hence, we can construct a term t' such that there is a computation $t' \to^*_{B_{i+1}} q_p$, where B_{i+1} does not apply the rule r, and that $t' \to^*_S p'$. Thus, the derivation $t' \to^*_{B_i} q_p$ holds as well. By the induction hypothesis we obtain that $p \to^*_S t'$, hence $p \to^*_S p'$. □

Lemma 5.3. *For each* i, $1 \leqslant i \leqslant m$, $L(B_i) \subseteq S^*(L)$.

Proof. We proceed by an induction on i. For $i=0$, $L(B_i)=L(A)=L \subseteq S^*(L)$. Let us assume that $L(B_i) \subseteq S^*(L)$, $0 \leqslant i \leqslant m-1$, and let us consider the rule $r \in R_{i+1}-R_i$. If a term t is accepted by B_{i+1} applying the rule r, $n(\geqslant 1)$ times, then it is easy to construct a term t' which is accepted by B_{i+1} applying the rule r, $n-1$ times and that t' reduces to t by (i) applying the rule $\bar{r}$ of S which corresponds to r, and (ii) if the right-hand side of $\bar{r}$ contains a ground subterm, then by reducing each ground subterm $p \in SUB$, appearing on the right-hand side of $\bar{r}$, to the corresponding subtree p' of t which was evaluated by B_{i+1} into the state q_p occurring in the left-hand side of r (see Lemma 5.2).

Hence, we can construct a term t'' which is accepted by B_i and t'' reduces to t by applying the rules of S. By the induction hypothesis, $t'' \in S^*(L)$, hence $t \in S^*(L)$. Thus, we obtained that $L(B_{i+1}) \subseteq S^*(L)$.

Since $B=B_m$, we have $L(B) \subseteq S^*(L)$. □

Consider the semi-monadic rewrite system S consisting of the rule $a(x) \to b(x, x)$ and the recognizable tree language $L=\{a(c^n(\$)) \mid n \geqslant 0\}$. It is clear that S is not right-linear and that $S^*(L)$ is not recognizable. On the other hand, we conjecture that for a right-linear semi-monadic rewrite system S and a recognizable tree language L, $S^*(L)$ is recognizable. The construction and the proof of Theorem 5.1 do not hold for the right-linear case because of the following reasons. In the left-linear case when proving the inclusion $S^*(L) \subseteq L(B)$, it is easy to see that if $t \to_S t'$ and there exists an accepting computation of B on t, then there exists an accepting computation on t' as well. On the other hand, if S is not left-linear, then two occurrences of a subtree of t that would correspond to different occurrences of a variable in a nonlinear left-hand side of a rule of S could be computed differently in the accepting computation of t.

5.2. Languages and tree pushdown automata

Let A be a bottom-up tree pushdown automaton one can define the tree language $L(A)$ recognized by A by

$$L(A)=\left\{ u \in T_\Sigma \;\middle|\; (\exists q \in Q_f) u \xrightarrow[T]{*} q(v) \right\}.$$

If we precise a special symbol *emptystack* one can also define the tree language $Le(A)$ recognized by A with emptystack by

$$Le(A)=\left\{u\in T_\Sigma \;\middle|\; u \underset{T}{\overset{*}{\rightarrow}} q(emptystack)\right\}.$$

Schimpf presents in [20] and [21] a class of bottom-up tree pushdown automata such that the class of tree languages recognized by such automata is identical to the class of context-free tree languages. We also mention the paper of Guessarian [14], where the author defines a class of topdown tree pushdown automata verifying the same property.

For our purpose it is easy to design a tpda_{12} and a tpda_{21} which real-time simulates any Turing machine ([3]). On the other hand, this simulation cannot be carried out by a tpda_{11}. We conjecture that the class of tree languages recognized by tpda_{11}'s is a proper subclass of context-free tree languages and that it can be analyzed in real time.

6. Conclusions

We have introduced several types of bottom-up tree pushdown automata and compared their transformational capability. We introduced the concept of the trf property, showed that it is decidable whether or not a rewrite system is trf, and associated to a convergent trf rewrite system a tpda^{rr} with reduce priority. If the rewrite system is left-linear, then the normal form can be computed by a dtpda^{rr}.

We now raise some open problems

(1) Introduce new special types of bottom-up tree pushdown automata. One can impose restrictions on the number of states, the number of rules or the maximal rank of pushdown symbols, one can define the concept of look-ahead [11] for tree pushdown tree automata. Moreover, one can consider totally defined tpda's. Mixing these concepts with the already introduced restrictions, one can introduce several types of tpda's. For example, one may consider a one-state totally defined tpda_{12} or a two-state tpda_{21} having unary stack symbols. It would be worthwhile studying the transformational and recognizing power of these devices. Finally, the compositions of tree transformation classes induced by tpda's were studied. There is a considerable interest in composing and decomposing tree transformation classes, (see [8, 11, 6]).

(2) Considering the hierarchies

$$TPDT_{11}\subseteq TPDT_{21}\subseteq TPDT_{31}\subseteq\cdots$$

$$DTPDT_{11}\subseteq DTPDT_{21}\subseteq DTPDT_{31}\subseteq\cdots$$

It is still open whether or not these hierarchies are proper.

(3) Generalize the concept of the semi-monadic rewrite system such that it still preserves recognizability. Note that, in general, it is undecidable whether or not a rewrite system preserves recognizability (see [4, 12]).

(4) Introduce the concept of a top-down tree pushdown automaton having one common tree stack for the states and study connection with rewrite systems.

Acknowledgment

We would like to thank the anonymous referees for much constructive criticism.

References

[1] R.V. Book, Thue systems as rewriting systems, *J. Symbolic Comput.* **3** (1987) 39–68.
[2] W.S. Brainerd, Tree generating regular systems, *Inform. and Control* **14** (1969) 217–231.
[3] J.L. Coquidé, M. Dauchet, R. Gilleron and S. Vágvölgyi, Bottom-up tree pushdown automata and rewrite systems, *Proc. RTA'91*, Lecture Notes in Computer Science, Vol. 488 (1991) 287–298.
[4] J.L. Coquidé, M. Dauchet and S. Tison, About connections between syntactical and computational complexity, *Proc. FCT'89*, Lecture Notes in Computer Science, Vol. 380 (1989) 105–115.
[5] M. Dauchet and S. Tison, The theory of ground rewrite systems is decidable, *Proc. 5th IEEE Symp. on Logic in Computer Science*, Philadelphia (1990) 242–248.
[6] M. Dauchet, P. Heuillard, P. Lescanne and S. Tison, Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems, *Inform. and Control* **88** (1990) 187–201.
[7] N. Dershowitz and J.P. Jouannaud, Rewrite systems, in: J.V. Leeuwen, eds., *Handbook of Theoretical Computer Science, Vol. B* (North-Holland, Amsterdam, 1990) 243–320.
[8] J. Engelfriet, Bottom-up and top-down tree transformations. A comparison, *Math. Systems Theory* **9** (1975) 198–231.
[9] Z. Fülöp and S. Vágvölgyi, A characterization of irreducible sets modulo left-linear term rewriting systems by tree automata, *Fund. Inform.* **13** (1990) 211–226.
[10] J.H. Gallier and R.V. Book, Reductions in tree replacement systems, *Theoret. Comput. Sci.* **37** (1985) 123–150.
[11] F. Gécseg and M. Steinby, *Tree Automata* (Akadémiai Kiadó. Budapest, 1984).
[12] R. Gilleron, Decision problems for term rewriting systems and recognizable tree languages, *Proc. STACS'91*, Lecture Notes in Computer Science, Vol. 480 (1991) 148–159.
[13] J. Goguen, J.W. Thatcher, E. Wagner and E. Wright, *An Initial Algebra Approach to the Specification Correctness, and Implementation of Abstract Data Types, Current Trends in Programming Methodology, Vol. 4* (Prentice-Hall, Englewood Cliffs, NJ, 1977) 80–149.
[14] I. Guessarian, Pushdown tree automata, *Math. Systems Theory* **16** (1983) 237–263.
[15] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems, *J. ACM* **27** (1980) 797–821.
[16] G. Huet and J.-J. Levy, Call by need computations in non-ambiguous linear term rewriting systems, Technical Report No. 359, INRIA, Le Chesnay, France, 1979.
[17] G. Huet and D.C. Oppen, Equations and rewrite rules: a survey, in R.V. Book, ed., *Formal Language Theory: Perspectives and Open Problems* (Academic Press, New York, 1980) 309–405.
[18] D.A. Plaisted, Semantic confluence tests and completion methods, *Inform. and Control* **65** (1985) 182–215.
[19] K. Salomaa, Deterministic tree pushdown automata and monadic tree rewriting systems, *J. Comput. System Sci.* **37** (1988) 367–394.
[20] K.M. Schimpf, A parsing method for context-free tree languages, Ph.D. Thesis, University of Pennsylvania, Pennsylvania, 1982.
[21] K.M. Schimpf and J.H. Gallier, Tree pushdown automata, *J. Comput. System Sci.* **30** (1985) 25–40.