JOURNAL OF COMPUTER AND SYSTEM SCIENCES 37, 367–394 (1988)

Deterministic Tree Pushdown Automata and Monadic Tree Rewriting Systems*

KAI SALOMAA

Department of Mathematics, University of Turku, SF-20500 Turku, Finland

Received March 30, 1987; revised February 2, 1988

We show that the deterministic tree pushdown automata of J. H. Gallier and R. V. Book (*Theoret. Comput. Sci.* 37 (1985), 123–150) are strictly more powerful than the corresponding automata of K. M. Schimpf (Ph. D. dissertation, University of Pennsylvania, 1982). In fact, even one of the additional features of the former automata, the capability to delete or to duplicate subtrees of the tree stack increases the recognition power. Also we show that finite unions of congruence classes of canonical monadic tree rewriting systems can be recognized by deterministic tree pushdown automata without the additional acceptance conditions used in *op. cit.* For right-linear monadic tree rewriting systems the same is true for unions of congruence classes over regular tree languages. © 1988 Academic Press, Inc.

1. INTRODUCTION

Replacement systems are an important tool, for instance, in theorem proving, algebraic simplification, and language theory. Usually they take the form of term rewriting systems, cf., e.g., [13] for an overview or when dealing with string Thue systems, cf. [6].

If the rewriting system is canonical, i.e., it is terminating and has the so called Church–Rosser property, then normal forms of objects (in our case, trees) can be computed using an arbitrary strategy. Here we obtain a representation of congruence classes of canonical monadic tree (i.e., term) rewriting systems using tree pushdown automata. Monadic tree rewriting systems were defined in [9] by lifting the concept of a monadic Thue system (cf. [5, 7]) to trees. Following [11, 17, 18] we define also tree pushdown automata as tree rewriting systems, and the power of the pushdown automaton model will depend on the form of rewrite rules that are allowed.

Tree pushdown automata generalize ordinary pushdown automata by having trees both as inputs and stacks. Bottom-up tree pushdown automata were defined

* This research was performed while the author was visiting the Department of Mathematics, University of California, Santa Barbara. It was supported in part by the Academy of Finland, the ASLA/Fulbright Foundation, and by the National Science Foundation under Grant DCR83-14977.

first in [17] and a simplified model was considered in [18]. The models of both [17 and 18] were shown to recognize exactly the family of context-free forests (we call tree languages "forests"). Context-free forests are forests generated by context-free tree grammars, cf. [2, 3, 8, 16], and as the yields of context-free forests one obtains the indexed languages, cf. [1]. Top-down tree pushdown automata were defined in [11] and were shown to recognize the family of context-free forests. Top-down tree pushdown automata simulate more closely than bottom-up automata the behaviour of context-free tree grammars.

However, there are many ways also to define a bottom-up tree pushdown automaton (all reasonable models when restricted to strings should, of course, be equivalent to ordinary pushdown automata), and another model was considered in [9]. It differs from the models of [17, 18] essentially in that it can duplicate and delete arbitrarily large subtrees of the tree stacks. This is caused by the fact that the right-hand sides of the rewrite rules of the automaton can have multiple occurrences of variables and do not necessarily contain all variables of the left-hand side of the rule.

Here in Section 4 we consider the (left-) linear deterministic versions of these automata and show that in this case even one of the capabilities, to be able to duplicate or to delete subtrees strictly increases the recognition power of the tree pushdown automata. In the case of the latter feature this might seem to be unexpected. The intuitive reason for this result is that tree pushdown automata without the deletion capability can use only the information in a part of a given tree stack the size of which is bounded by a polynomial function of the height of the tree stack.

In [5] it is shown that finite unions of congruence classes of monadic Church–Rosser Thue systems are deterministic context-free languages. It was shown in [9] that finite unions of congruence classes of canonical monadic tree rewriting systems can be recognized by a deterministic tree pushdown automaton that has the additional capability to check whether the final contents of the tree stack belong to a given regular forest. In Section 5 we show that this stronger acceptance condition is not necessary.

For right-linear monadic systems we get the stronger result that the union of congruence classes can be indexed by a regular forest. As a corollary of the proof we see that for an arbitrary (possibly infinite) Thue system the set of descendants of a regular set is always regular. This is a strenghtening of a result of [7], where it was shown for context-free monadic Thue systems.

It is assumed that the reader is familiar with trees and tree rewriting systems. In Section 2 we present briefly and informally some notation and definitions used later, for a more comprehensive treatment cf., e.g., [9, 10, 12, 13, 15, 17]. In Section 3 we introduce the notion of a k-passable tree. Results about k-passable trees will be used in Section 4 to see how some models of tree pushdown automata are able to use their tree stacks.

2. Preliminaries

Let A be a set and $m > 0$. Then A^m denotes the Cartesian product of m copies of A, ${}^{\#}A$ denotes the cardinality of A and $\mathscr{P}(A)$ the set of subsets of A. If $\delta \subseteq A \times A$ is a binary relation, the reflexive transitive closure of δ is denoted by δ^*. The letter N denotes the set of natural numbers (not containing zero) and λ denotes the empty word. The symbol "$\subset$" is used for the strict inclusion relation of sets.

In the following $X = \{x_1, x_2, ...\}$ always denotes a denumerable set of variables, and $X_m = \{x_1, x_2, ..., x_m\}$, $m \geqslant 0$. The symbols Σ, Ω, Γ, and Δ denote finite *ranked alphabets*. The rank function of a given ranked alphabet Σ is usually denoted by r. The set of m-ary ($m \geqslant 0$) symbols of Σ is denoted by Σ_m.

2.1. Definition. A *tree domain* is a finite nonempty subset D of N^* such that

(i) If $u \in D$ and v is a prefix of u, then $v \in D$.

(ii) If $ui \in D$, where $u \in N^*$ and $i \in N$, then $uj \in D$ for every j, $1 \leqslant j \leqslant i$.

The *length* of a word $u \in N^*$ is denoted by $|u|$.

Let Y be a finite or denumerable set of variables. Then the set of *Σ-trees with variables Y* (*or ΣY-trees*), $F_\Sigma(Y)$, is the smallest set A such that

(i) $\Sigma_0 \cup Y \subseteq A$ and

(ii) if $m > 0$, $\sigma \in \Sigma_m$, and $t_1, ..., t_m \in A$, then $\sigma(t_1, ..., t_m) \in A$.

The set $F_\Sigma(\varnothing)$ is denoted by F_Σ. Subsets of $F_\Sigma(Y)$ (F_Σ) are called *ΣY- (Σ-) forests*. The tree domain of a Σ-tree t, $\mathrm{dom}(t)$, is defined in the usual way. If u, $v \in \mathrm{dom}(t)$, then the node u is said to be a *predecessor* of v if u is a prefix of v. Also in this case v is said to be a *successor* of u. The nodes u and v are *independent* if neither one is the predecessor of the other. The relation $<_{\mathrm{lr}} \subseteq \mathrm{dom}(t) \times \mathrm{dom}(t)$ denotes the natural left to right ordering of pairwise independent nodes of t. ($u <_{\mathrm{lr}} v$ iff we can write $u = wj_1u'$ and $v = wj_2v'$, where w, u', $v' \in N^*$, $j_1, j_2 \in N$, and $j_1 < j_2$.)

Let t, $t_1 \in F_\Sigma(X)$ and $u \in \mathrm{dom}(t)$. Then $\mathrm{lab}(t, u)$ ($\in \Sigma \cup X$) denotes the label of the node u of t, $t(u)$ denotes the subtree of t at node u and $t(u \leftarrow t_1)$ denotes the Σ-tree that is obtained from t by replacing the subtree at node u with t_1. The height of t is denoted by $\mathrm{hg}(t)$. The *size* of the tree t is defined to be ${}^{\#}\mathrm{dom}(t)$ and it is denoted simply by ${}^{\#}t$.

The node u is said to be a *leaf* of t if ui does not belong to $\mathrm{dom}(t)$ for any $i \in N$. The set $\mathrm{path}(t) \subset \Sigma^+$ consists of all strings of elements of Σ that occur as labels of a path in $\mathrm{dom}(t)$ from the *root* (the node with the address λ) to a leaf of t. The tree t is said to be *balanced* if all strings in $\mathrm{path}(t)$ have equal length. The set $\mathrm{var}(t) \subset X$ consists of all elements of X that occur as a label of some leaf of t. The tree t is said to be *linear* (in variables of X) if no variable of X labels more than one node of t.

If t, $t_1, ..., t_m \in F_\Sigma(X)$ and $z_1, ..., z_m \in X$, then

$$t(z_1 \leftarrow t_1, ..., z_m \leftarrow t_m)$$

denotes the ΣX-tree that is obtained from t by replacing every occurrence of a variable z_i in t by t_i, $i=1, ..., m$.

2.2. DEFINITION. Let Σ and Ω be ranked alphabets. For every $m \geqslant 0$ let $\Xi_m = \{\xi_1, ..., \xi_m\}$ denote a set of variables (disjoint from X, Σ, and Ω). Suppose that for every $m \geqslant 0$ we are given a mapping

$$h_m: \Sigma_m \to F_\Omega(\Xi_m).$$

The *tree homomorphism*

$$h: F_\Sigma(X) \to F_\Omega(X)$$

determined by the mappings h_m is defined inductively as follows:

(i) For every $x \in X$, $h(x) = x$.

(ii) Let $m \geqslant 0$, $\sigma \in \Sigma_m$, and $t_1, ..., t_m \in F_\Sigma(X)$. Then

$$h(\sigma(t_1, ..., t_m)) = h_m(\sigma)(\xi_1 \leftarrow h(t_1), ..., \xi_m \leftarrow h(t_m)).$$

(Here we consider only tree homomorphisms that map every variable of X onto itself.)

A mapping $\alpha: X \to F_\Sigma(X)$ is a *substitution* if $\mathrm{dom}(\alpha) = \{x \in X \mid \alpha(x) \neq x\}$ is finite. We say that α is a substitution $X_m \to F_\Sigma(X)$ if $\mathrm{dom}(\alpha) \subseteq X_m$. If $\alpha: X \to F_\Sigma(X)$ is a substitution, we denote also its unique homomorphic extension $\hat{\alpha}: F_\Sigma(X) \to F_\Sigma(X)$ simply by α.

Trees t_1, $t_2 \in F_\Sigma(X)$ are said to be *unifiable* if there exist substitutions $\alpha_i: X \to F_\Sigma(X)$, $i = 1, 2$, such that $\alpha_1(t_1) = \alpha_2(t_2)$. We denote this by $t_1 \downarrow_u t_2$. (Sometimes unifiability of t_1 and t_2 is defined slightly differently by requiring that there exists a substitution α such that $\alpha(t_1) = \alpha(t_2)$. In this case one always allows the renaming of variables of t_1 and t_2, so the definitions are clearly equivalent.)

2.3. DEFINITIONS. Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$. The set S is called a *tree rewriting* (*or term rewriting*) *system.* We define the *rewrite relation* $\to_S \subseteq F_\Sigma \times F_\Sigma$ determined by S as follows:

Let $t_1, t_2 \in F_\Sigma$. Then $t_1 \to_S t_2$ iff there exists a pair $(s_1, s_2) \in S$, where $\mathrm{var}(s_1) \cup \mathrm{var}(s_2) \subseteq X_m$, and a substitution $\alpha: X_m \to F_\Sigma(X)$ such that we can write

$$t_1 = p(y \leftarrow \alpha(s_1)) \qquad \text{and} \qquad t_2 = p(y \leftarrow \alpha(s_2)),$$

where $p \in F_\Sigma(\{y\})$ and exactly one node of p is labeled by y.

In the above case we say that t_1 is *reducible* using the rule (s_1, s_2). If $t \in F_\Sigma$ is not reducible using any of the rules of S, we say that t is *irreducible* mod S. The set of irreducible Σ-trees mod S is denoted by $\mathrm{IRR}(S)$. The rewriting relation S is said to be *left-* (*right-*) *linear* if all the left- (right-) hand sides of rules of S are linear in variables X.

The relation $\leftrightarrow_S$ is defined by

$$\leftrightarrow_S = \rightarrow_S \cup \rightarrow_S^{-1}.$$

The *congruence class* $[t]_S$ of a tree $t \in F_\Sigma$ is determined by

$$[t]_S = \{t' \in F_\Sigma \mid t \leftrightarrow_S^* t'\}.$$

When S is clear from the context we denote $[t]_S$ simply by $[t]$. The set of *descendants* of t is

$$S^*(t) = \{t' \in F_\Sigma \mid t \rightarrow_S^* t'\}.$$

Furthermore, if $L \subseteq F_\Sigma$ we denote

$$[L] = \{t' \in F_\Sigma \mid (\exists t \in L)\, t' \in [t]\}$$
$$S^*(L) = \{t' \in F_\Sigma \mid (\exists t \in L)\, t' \in S^*(t)\}.$$

The rewriting system S (or $\rightarrow_S$) is said to be *Church–Rosser* (*or confluent*) if for all t, $t' \in F_\Sigma$ such that $t \leftrightarrow_S^* t'$: $S^*(t) \cap S^*(t') \neq \varnothing$, and it is said to be *terminating* (or *Noetherian*) if there exist no infinite sequences $t_1 \rightarrow_S t_2 \rightarrow_S \cdots$. The system S is *canonical* (or *complete*) if it is terminating and Church–Rosser.

The elements (s_1, s_2) of S are usually given in the form $s_1 \rightarrow_S s_2$ and when S is clear from the context $\rightarrow_S$ ($\leftrightarrow_S$) is denoted simply by $\rightarrow$ ($\leftrightarrow$). It is well known that if S is Church–Rosser, then every congruence class has at most one irreducible element.

2.4. DEFINITIONS. A (*nondeterministic*) *bottom-up tree automaton* (*or recognizer*) is a four-tuple $\mathbf{A} = (\Sigma, Q, Q_F, g)$, where

(i) Σ is a ranked alphabet.

(ii) Q is a finite set of states.

(iii) $Q_F \subseteq Q$ is the set of accepting final states.

(iv) g is a mapping that associates with every element $\sigma \in \Sigma_k$, $k \geqslant 0$, a function $\sigma_g: Q^k \rightarrow \mathscr{P}(Q)$. If $\sigma \in \Sigma_0$, σ_g is interpreted to be an element of $\mathscr{P}(Q)$.

Define the ranked alphabet $\Omega = \Sigma \cup Q$, where $\Omega_0 = \Sigma_0 \cup Q$ and $\Omega_k = \Sigma_k$, when $k > 0$. The set $G \subseteq F_\Omega \times F_\Omega$ of rewrite rules of $\mathbf{A}$ consists of all the rules

$$\sigma(q_1, ..., q_k) \rightarrow_G q,$$

$q \in \sigma_g(q_1, ..., q_k)$, $k \geqslant 0$, $\sigma \in \Sigma_k$, $q_1, ..., q_k \in Q$. If $t \in F_\Omega$, we define $t_g = \{q \in Q \mid t \rightarrow_G^* q\}$. The forest *recognized by* $\mathbf{A}$ is

$$L(\mathbf{A}) = \{t \in F_\Sigma \mid t_g \cap Q_F \neq \varnothing\}.$$

A forest is *regular* if it can be recognized by a nondeterministic bottom-up tree automaton. The family of regular forests is denoted by *REG*.

The automaton $\mathbf{A} = (\Sigma, Q, Q_F, g)$ is said to be *deterministic* if $^{\#}\sigma_g(q_1, ..., q_k) = 1$ for all $\sigma \in \Sigma_k$, $q_1, ..., q_k \in Q$. It is well known (cf. [10]) that every regular forest can be recognized by a deterministic bottom-up automaton. If $\mathbf{A}$ is deterministic and $t_g = \{q\}$, $q \in Q$, we denote t_g simply by q.

The automaton $\mathbf{A} = (\Sigma, Q, Q_F, g)$ is *connected* if for every $q \in Q$ there exists $t \in F_\Sigma$ such that $q \in t_g$. Every regular forest can be recognized by a (deterministic) connected automaton, cf. [10].

3. k-Passable Trees

A tree t is k-passable if a top-down (or bottom-up) tree automaton can compute t using simultaneously at most k reading heads. Here we show that for every k there is a polynomial p_k such that for every k-passable tree t, $^{\#}t \leqslant p_k(\mathrm{hg}(t))$. This result will be useful in Section 4 because it turns out that for certain tree pushdown automata the part of the tree stack that can affect the computation is a k-passable supertree of the stack.

3.1. Definitions. Let Σ be a ranked alphabet and let ∇ be a symbol of rank one not in Σ. The *one-state top-down Σ-tree automaton* is the set

$$S \subseteq F_{\Sigma \cup \{\nabla\}}(X) \times F_{\Sigma \cup \{\nabla\}}(X)$$

of rewrite rules, where for every $m \geqslant 0$ and $\sigma \in \Sigma_m$, S contains the rule

$$\nabla(\sigma(x_1, ..., x_m)) \rightarrow_S \sigma(\nabla(x_1), ..., \nabla(x_m)).$$

(If $m = 0$, this reduces to $\nabla(\sigma) \rightarrow_S \sigma$.) Let $k \geqslant 1$ and $t \in F_\Sigma$. A *k-computation* on t is a computation

$$\nabla(t) = t_0 \rightarrow_S t_1 \rightarrow_S \cdots \rightarrow_S t_n, \tag{1}$$

where $n \geqslant 0$, $t_i \in F_{\Sigma \cup \{\nabla\}}$ and t_i contains at most k symbols ∇, $i = 0, ..., n$. (Hence intuitively in a k-computation the automaton can have at most k simultaneously active reading heads.) The tree t is *k-passable* if there exists a k-computation (1), where $t_n = t$.

Let $s \in F_\Sigma(Y_{k'})$, $k' \leqslant k$, be linear in variables of $Y_{k'} = \{y_1, ..., y_{k'}\}$ and suppose that there exist $s_1, ..., s_{k'} \in F_\Sigma$ such that

$$t = s(y_1 \leftarrow s_1, ..., y_{k'} \leftarrow s_{k'}).$$

We say that s is a *k-passable supertree of t* if there exists a k-computation (1), where

$$t_n = s(y_1 \leftarrow \nabla(s_1), ..., y_{k'} \leftarrow \nabla(s_{k'}))$$

and for every $u \in \mathrm{dom}(s)$ there exists $i \in \{0, ..., n\}$ such that

$$\mathrm{lab}(t_i, u) = \nabla.$$

The latter condition means just that all subtrees of s not containing variables have been "completely read" in the k-computation (1). The set of k-passable supertrees of t is denoted by $U(t, k)$. Define the ranked alphabet $\Omega = \Sigma \cup \bar{\Sigma}$, where $\bar{\Sigma} = \{\bar{\sigma} \mid \sigma \in \Sigma\}$ and the rank of elements of $\bar{\Sigma}$ is zero. The *supremum* of the k-passable supertrees of t, $S(t, k)$, is the Ω-tree defined as follows:

(i) $\mathrm{dom}(S(t, k)) = \bigcup(\mathrm{dom}(s) \mid s \in U(t, k))$.

Let $u \in \mathrm{dom}(S(t, k))$.

(iia) If there exists $s \in U(t, k)$ such that $\mathrm{lab}(s, u) \in \Sigma$, then $\mathrm{lab}(S(t, k), u) = \mathrm{lab}(t, u)$.

(iib) If u does not belong to (iia) and $\mathrm{lab}(t, u) = \sigma$, then $\mathrm{lab}(S(t, k), u) = \bar{\sigma}$.

Clearly $\mathrm{dom}(S(t, k))$ defined by (i) is a tree domain. Note that in the case (iib) no successor of u belongs to $\mathrm{dom}(S(t, k))$ and hence u has to be labeled by an element of Ω_0. Intuitively $S(t, k)$ is just the "top-part" of t containing all the k-passable supertrees of t.

3.2. Lemma. *Let $t \in F_\Sigma$ be k-passable, $k \geqslant 1$. Then*

$$^{\#}t \leqslant k(\mathrm{hg}(t))^k + 1.$$

Proof. Let $p(n, k)$ denote the maximal size of a k-passable Σ-tree of height n. Without restriction we may assume that

$$\Sigma_m \neq \varnothing \qquad \text{for all} \quad m \leqslant k. \tag{2}$$

(Adding new symbols to Σ can only increase $p(n, k)$.) Clearly by (2),

$$p(n, k_1) < p(n, k_2) \qquad \text{and} \qquad p(n_1, k') < p(n_2, k') \tag{3}$$

for all $n \geqslant 1$, $1 \leqslant k_1 < k_2 \leqslant k$, $n_1 < n_2$, and $1 \leqslant k' \leqslant k$.

From (2) and (3) it follows that a k-passable Σ-tree of height n and maximal size is of the form

$$\sigma(t_1, ..., t_k),$$

where $\sigma \in \Sigma_k$ and t_i is an i-passable Σ-tree of height $n-1$ with maximal size, $i = 1, ..., k$. Here the order of the subtrees t_i can, of course, be arbitrary. Hence we have

$$p(n, k) = 1 + p(n-1, 1) + p(n-1, 2) + \cdots + p(n-1, k). \tag{4}$$

From (4) we see that

$$p(n, k) = p(n, k-1) + p(n-1, k). \tag{5}$$

By (5) and (3) and noticing that $p(0, k) = 1$ and $p(n, 1) = n+1$ we have, furthermore,

$$\begin{aligned} p(n, k) &= p(n, k-1) + p(n-1, k-1) + p(n-2, k) = \cdots \\ &= p(n, k-1) + p(n-1, k-1) + \cdots + p(1, k-1) + p(0, k) \leqslant np(n, k-1) + 1 \\ &\leqslant n^2 p(n, k-2) + n + 1 \leqslant \cdots \leqslant n^{k-1} p(n, 1) + n^{k-2} + \cdots + n + 1 \\ &= n^k + n^{k-1} + \cdots + n + 1 \leqslant kn^k + 1. \quad \blacksquare \end{aligned}$$

3.3. Lemma. *Let Σ be a ranked alphabet and define*

$$M = \max\{r(\sigma) \mid \sigma \in \Sigma\} + 2.$$

Define $g(n, k) = Mn^{Mk} + 1$, $n \geqslant 0$, $k \geqslant 1$. Then for all $t \in F_\Sigma$,

$$^{\#}S(t, k) \leqslant g(\mathrm{hg}(t), k). \tag{6}$$

Proof. If $\mathrm{hg}(t) = 0$ then $^{\#}S(t, k) = 1$, and if $\mathrm{hg}(t) = 1$ then $^{\#}S(t, k) \leqslant M + 1$. Suppose then that for all $k \geqslant 1$, (6) holds for all Σ-trees of height at most n, $n \geqslant 1$. Let t be a Σ-tree of height $n+1$,

$$t = \sigma(t_1, \ldots, t_m),$$

$m \geqslant 1$, $\sigma \in \Sigma_m$, and let $k \geqslant 1$ be arbitrary.

(i) Suppose that $m-1$ of the subtrees $t_1, \ldots, t_m$ are k-passable, without restriction we may assume that these are $t_1, \ldots, t_{m-1}$. By Lemma 3.2 and the induction assumption we have

$$\begin{aligned} ^{\#}S(t, k) &\leqslant 1 + {}^{\#}t_1 + \cdots + {}^{\#}t_{m-1} + {}^{\#}S(t_m, k) \\ &\leqslant 1 + (m-1)(kn^k + 1) + Mn^{Mk} + 1. \end{aligned}$$

On the other hand,

$$g(n+1, k) = M(n+1)^{MK} + 1 \geqslant Mn^{Mk} + (M^2 k)\, n^{Mk-1} + 1 \tag{7}$$

and (6) holds since $M \geqslant 2$ and $M \geqslant m$.

(ii) Suppose then that (i) does not hold. This means that $m \geqslant 2$ and at least two of the trees $t_1, \ldots, t_m$ are not k-passable. If $k = 1$, then $^{\#}S(t, k) = 1$ because $m \geqslant 2$. Hence we assume in the following that $k > 1$. When computing an arbitrary k-passable supertree of t the top-down automaton is able to use at most $k-1$ heads on any one of the subtrees $t_1, ..., t_m$. Hence by the induction assumption and (7) we have

$$\begin{aligned} {}^{\#}S(t, k) &\leqslant 1 + {}^{\#}S(t_1, k-1) + \cdots + {}^{\#}S(t_m, k-1) \\ &\leqslant 1 + m(Mn^{M(k-1)} + 1) \leqslant g(n+1, k). \quad \blacksquare \end{aligned}$$

The upper bound g in the above lemma was chosen just to make the proof easy and is not optimal. The essential part in the proof is to divide the induction step into the two cases and thereby be able to use the fact that if the automaton can use all k reading heads on one of the subtrees $t_1, ..., t_m$, then the other subtrees must be "very small."

4. Tree Pushdown Automata

In the following we define extended tree pushdown automata that are able to simulate the computations of the automata of [9]. As a restriction of these we define the basic tree pushdown automata that are equivalent to the model of [17].

4.1 Definition. A *(bottom-up) extended (nondeterministic) tree pushdown automaton, ex-tpa*, is a four-tuple $\mathbf{A} = (\Sigma, \Gamma, \Gamma', \delta)$, where

(i) Σ is a finite ranked alphabet of input symbols,

(ii) Γ is a finite ranked alphabet of stack symbols disjoint from Σ,

(iii) $\Gamma' \subseteq \Gamma$ is the set of accepting stack symbols, and

(iv) δ is a partial transition function with the properties:

(a) The domain of δ, dom(δ), is a finite subset of

$$\bigcup(\Sigma_k \times \Gamma^k \mid k \geqslant 0) \cup F_\Gamma(X).$$

(b) For all $k \geqslant 0$, δ maps elements of $\Sigma_k \times \Gamma^k$ to subsets of Γ_k. (Here Γ^k is the Cartesian product of k copies of Γ.)

(c) If $t \in F_\Gamma(X) \cap \text{dom}(\delta)$, then $\delta(t)$ is a finite set of trees of the form

$$f(z_1, ..., z_k),$$

$m, k \geqslant 0$, $t \in F_\Gamma(X_m)$, $f \in \Gamma_k$, $z_i \in X_m \cup \Gamma_0$, $i = 1, ..., k$. Note that t needs not have occurrences of all variables of X_m.

The automaton $\mathbf{A}$ is said to be a *deleting tree pushdown automaton, del-tpa*, if (c) is replaced by

(c′) If $t \in F_\Gamma(X) \cap \text{dom}(\delta)$, then $\delta(t)$ is a finite set of trees of the form

$$f(x_1, ..., x_n),$$

$0 \leqslant n \leqslant m$, $t \in F_\Gamma(X_m)$, $f \in \Gamma_n$.

The automaton **A** is a *basic tree pushdown automaton, ba-tpa*, if (c) is replaced by

(c″) If $t \in F_\Gamma(X) \cap \mathrm{dom}(\delta)$, then $\delta(t)$ is a finite set of trees of the form

$$f(x_1, ..., x_m),$$

$m \geqslant 0$, $t \in F_\Gamma(X_m)$, $f \in \Gamma_m$.

The automaton **A** is said to be *linear* (or left-linear), if all ΓX-trees belonging to $\mathrm{dom}(\delta)$ are linear in variables of X. The class of linear basic (deleting, extended) tree pushdown automata is denoted by *lin-ba-tpa* (*lin-del-tpa*, *lin-ex-tpa*).

Let Δ be the ranked alphabet $\Delta = \Sigma \cup \Gamma \cup \{\nabla\}$, where ∇ is not in $\Sigma \cup \Gamma$ and $r(\nabla) = 1$. Intuitively ∇ corresponds to the reading heads of the pushdown automaton **A**. Now we are ready to define the set of rewrite rules that describe the computation of **A**.

4.2. Definitions. Let $\mathbf{A} = (\Sigma, \Gamma, \Gamma', \delta)$ be some pushdown automaton of Definition 4.1. The set of rewrite rules of **A**, $\Delta \subseteq F_\Delta(X) \times F_\Delta(X)$, consists of the rules defined in (i)–(iii) below. (As mentioned in Section 2 we denote a relation $(s, t) \in \Delta$ by $s \rightarrow_\Delta t$.)

(i) If $f \in \delta(\sigma)$, $\sigma \in \Sigma_0$, then

$$\sigma \rightarrow_\Delta \nabla(f).$$

Let $M = \max\{r(\gamma) \mid \gamma \in \Gamma\}$. For all i, j such that $i \geqslant 1$ and $0 \leqslant j \leqslant M$ we denote

$$\bar{x}_{i,j} = (x_{(i-1)M+1}, ..., x_{(i-1)M+j}).$$

(Note that if $i \neq i'$, then for all j, j' the sets of variables occurring in the vectors $\bar{x}_{i,j}$ and $\bar{x}_{i',j'}$ are disjoint.)

(ii) Let $f \in \delta(\sigma, f_1, ..., f_k)$, $k > 0$, $\sigma \in \Sigma_k$, $f_i \in \Gamma$, and denote $r(f_i) = r(i)$, $i = 1, ..., k$. Then

$$\sigma(\nabla(f_1(\bar{x}_{1,r(1)})), ..., \nabla(f_k(\bar{x}_{k,r(k)}))) \rightarrow_\Delta \nabla(f(f_1(\bar{x}_{1,r(1)}), ..., f_k(\bar{x}_{k,r(k)}))).$$

(iii) If $r \in \delta(t)$, $t \in F_\Gamma(X)$, then

$$\nabla(t) \rightarrow_\Delta \nabla(r).$$

The forest *recognized by* **A**, $L(\mathbf{A})$, and the forest *recognized by* **A** *with empty tree stack*, $L_e(\mathbf{A})$, are defined as

$$L(\mathbf{A}) = \{t \in F_\Sigma \mid (\exists t' \in F_\Gamma)\ t \rightarrow_\Delta^* \nabla(t'), \mathrm{lab}(t', \lambda) \in \Gamma'\},$$

$$L_e(\mathbf{A}) = \{t \in F_\Sigma \mid (\exists \gamma \in \Gamma_0 \cap \Gamma')\ t \rightarrow_\Delta^* \nabla(\gamma)\}.$$

If K is a class of tree pushdown automata, then the family of forests recognized (with empty tree stack) by the automata of K is denoted by $\mathscr{L}(K)$ (resp. $\mathscr{L}_e(K)$).

Note that the set Δ of rewrite rules is finite since $\mathrm{dom}(\delta)$ is finite. Rules (i) read a leaf of the input tree and initiate a tree stack. The assumption $\Sigma \cap \Gamma = \emptyset$ guarantees that **A** cannot create new stacks within an existing tree stack. Rules (ii) read a nonleaf symbol of the input tree and rules (iii) change the contents of a tree stack. The rules (i) could be obtained from (ii) by making the appropriate interpretation for the case $k = 0$. Rules (i) and (ii) are called *shift-rules* of **A** and rules (iii) are called *reductions* of **A**. Intuitively the forest $L(\mathbf{A})$ consists of such Σ-trees t that **A** is able to read t arriving at the root of t with the top of the tree stack labeled by an element of Γ'.

Essentially the basic tree pushdown automaton is a "stateless tree pushdown automaton" of [17] that is further restricted not to have the capability to "look back" at the symbols labeling the roots of the subtrees of the tree stack corresponding to variables of a reduction rule. On the other hand, the basic tree pushdown automata are clearly more powerful than the automata of [18] and, since these as well as the stateless automata of [17] recognize the context-free forests, it follows that also $\mathscr{L}_e$(ba-tpa) equals to the family of context-free forests. (The automata of [17] must at the end of the computation have in the stack a fixed nullary symbol denoting the empty stack but clearly it does not make a difference if we have a finite number of accepting symbols of rank zero.) The ba-tpa differ from the tree pushdown automata of [17, 18] also in that in the computation the automaton does not carry along the subtrees of the input tree that have been processed (and hence the rank of ∇ is one). This is, of course, only a notational simplification.

The tree pushdown automaton model of [18] is still simpler than the ba-tpa but it is not suitable for our purposes since we are here mainly concerned with the deterministic versions of the automata. The automata of [18] can always make a shift-move and there seems to be no natural way to define a deterministic computation. If one, for instance, requires the deterministic automata to first perform all the shift-moves, then it is easy to see that in the string case (i.e., when both the input and the stack consist only of symbols of rank at most one) these automata would recognize only the regular languages. Hence they could not be considered as a natural generalization of deterministic string pushdown automata.

It is straightforward to see that the ex-tpa and the tree pushdown automata of [9] are equivalent. To simulate a computation of an automaton of [9] the ex-tpa stores the states to the root symbols of the tree stacks and similarly the automaton of [9] can use the internal states to remember the top symbols of the tree stacks of an ex-tpa that it is simulating. (Note that an ex-tpa can simulate a reduction of the automaton of [9] of the form $s \to x$, $x \in \mathrm{var}(s)$, by using all possible rules $\nabla(s(x \leftarrow \gamma(x_1', ..., x_k'))) \to \nabla(\gamma(x_1', ..., x_k'))$, where $k \geq 0$, $\gamma \in \Gamma_k$, and the variables $x_1', ..., x_k'$ are not in $\mathrm{var}(s)$.)

Since we may always choose the set of accepting stack symbols to consist only of nullary symbols it is clear that for all automata classes of Definition 4.1, $\mathscr{L}_e(K) \subseteq \mathscr{L}(K)$. The extended and deleting tree pushdown automata can easily be made capable of reducing an arbitrary tree stack to height zero, and hence it follows that for $K' \in \{\text{del-tpa, ex-tpa}\}$, $\mathscr{L}_e(K') = \mathscr{L}(K')$. Using similar techniques as in the proof of Lemma 4.6 it can be shown that for an arbitrary ba-tpa **A** such that the stack alphabet Γ contains some symbols of rank greater than one there exist some tree stacks that **A** is not able to reduce to height zero. This suggests that possibly $\mathscr{L}_e(\text{ba-tpa}) \subset \mathscr{L}(\text{ba-tpa})$.

The deleting tree pushdown automaton is an intermediate version between the basic and the extended automata. It has the power to delete subtrees in the stack but not to duplicate them. Here we concentrate on the deterministic linear versions of these automata and show that in this case the deleting automata are strictly more powerful than the basic automata and the extended automata are strictly more powerful than the deleting ones. Next we define the deterministic versions of the above automata.

4.3. Definition. Let $\mathbf{A} = (\Sigma, \Gamma, \Gamma', \delta) \in H$-tpa, where $H \in \{\text{ba, del, ex, lin-ba, lin-del, lin-ex}\}$. Then **A** is said to be a *deterministic* H-tpa, H-dtpa, if

(a) For all $\sigma \in \Sigma_k, f_1, ..., f_k \in \Gamma$ $(k \geqslant 0)$,

$$^{\#}\delta(\sigma, f_1, ..., f_k) \leqslant 1.$$

(b) If $\delta(\sigma, f_1, ..., f_k) \neq \varnothing$, $\sigma \in \Sigma_k, f_1, ..., f_k \in \Gamma$, then for all $t \in F_\Gamma(X)$ such that $\text{lab}(t, \lambda) = f_i$, $i \in \{1, ..., k\}$: $\delta(t) = \varnothing$. Also $\delta(x) = \varnothing$ for all $x \in X$.

(c) If $t \in F_\Gamma(X)$ and $r \in \delta(t)$, then $\delta(t) = \{r\}$ and $\text{var}(r) \subseteq \text{var}(t)$.

(d) Suppose that t_1, $t_2 \in F_\Gamma(X)$, $t_1 \neq t_2$, and $t_1 \downarrow_u t_2$. Then $\delta(t_1) = \varnothing$ or $\delta(t_2) = \varnothing$.

The conditions of Definition 4.3 guarantee that at each stage there is at most one way for the automaton to continue the computation. Condition (a) guarantees that at most one shift-rule is applicable. Condition (b) prohibits shift-reduce conflicts and (c) and (d) together guarantee that there is at most one way to reduce a given tree stack. (If $\delta(x)$ would be nonempty for some $x \in X$, then for the computation of **A** to be "deterministic," δ could have shift-rules only for nullary input symbols and the single reduction rule determined by $\delta(x)$. The condition $\text{var}(r) \subseteq \text{var}(t)$ in (c) guarantees that there is no nondeterminism in the choice of Γ-trees corresponding to variables occurring in the right-hand side but not the left-hand side of a rule.)

4.4. Definition. Let $\mathbf{A} = (\Sigma, \Gamma, \Gamma', \delta)$ be an extended (or deleting or basic) tree pushdown automaton. The set of (Σ, Γ)-*configurations* $\text{con}(\Sigma, \Gamma)$ is defined to consist of all elements $t \in F_{\Sigma \cup \Gamma \cup \{\nabla\}}$ such that

$$\text{path}(t) \subseteq \Sigma^* \nabla \Gamma^+ \cup \Sigma^*.$$

In the following we define a forest L such that

$$L \in \mathscr{L}(\text{lin-del-dtpa}) - \mathscr{L}(\text{lin-ba-dtpa}).$$

Choose $\Sigma=\Sigma_2\cup\Sigma_1\cup\Sigma_0$, where $\Sigma_2=\{a\}$, $\Sigma_1=\{b(1), b(2), c(1), c(2)\}$ and $\Sigma_0=\{d(1), d(2)\}$. First we define a forest L_0 as: L_0 consists of all Σ-trees that are of the form

$$t_1(x_1\leftarrow t_2), \tag{1}$$

where

(i) The root of t_1 is labeled by $c(1)$ or $c(2)$, all other nonleaf nodes of t_1 are labeled by $b(1)$ or $b(2)$ and the single leaf of t_1 is labeled by x_1. Also the height of t_1 is at least two.

(ii) t_2 is an arbitrary $\Sigma_2\cup\Sigma_0$-tree.

Clearly the forest L_0 is regular. Let

$$t_1=c(i)\,b(j_k)\cdots b(j_1)(x_1),$$

$i, j_1, ..., j_k\in\{1, 2\}$, $k\geqslant 1$ (i.e., t_1 is a unary tree as in (i). We have left some of the parentheses out.) Then by $P(t_1)$ we denote the string

$$P(t_1)=j_1\cdots j_k\in\{1, 2\}^+.$$

Now we define the forest L to consist of all Σ-trees $t_1(x_1\leftarrow t_2)$ as in (1) such that if $\text{lab}(t_1, \lambda)=c(i)$, then $\text{lab}(t_2, P(t_1))=d(i)$, $i\in\{1, 2\}$.

An example of a tree of L is given in Fig. 1. Here

$$P(t_1)=21, \qquad t_2=a(d(2), a(d(1), d(2))), \qquad \text{lab}(t_2, 21)=d(1).$$

4.5. LEMMA. *Let Σ and L be as above. Then*

$$L\in\mathscr{L}(\text{lin-del-dtpa}).$$

Proof. We define a lin-del-dtpa $\mathbf{A}=(\Sigma, \Gamma, \Gamma', \delta)$ such that $L(\mathbf{A})=L$. Choose $\Gamma=\Gamma_2\cup\Gamma_1\cup\Gamma_0$, where $\Gamma_2=\{A\}$, $\Gamma_1=\{B(1), B(2), C(1), C(2), E\}$, $\Gamma_0=\{D(1), D(2), F\}$, and $\Gamma'=\{F\}$. The relation δ is defined by (i)–(vii). (The shift-rules are defined in (i)–(iv) and the reduction rules in (v)–(vii).)

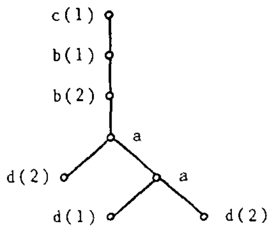

FIG. 1. An example of a tree of the forest L.

(i) $\delta(d(j)) = \{D(j)\}$, $j = 1, 2$.

(ii) $\delta(a, Z_1, Z_2) = \{A\}$ if $Z_1, Z_2 \in \{D(1), D(2), A\}$.

(iii) $\delta(b(j), Z) = \{B(j)\}$ if $Z \in \{A, E\}$, $j = 1, 2$.

(iv) $\delta(c(j), E) = \{C(j)\}$, $j = 1, 2$.

(v) $\delta[B(1)(A(x_1, x_2))] = \{E(x_1)\}$, $\delta[B(2)(A(x_2, x_1))] = \{E(x_1)\}$.

(vi) $\delta[B(1)(E(A(x_1, x_2)))] = \{E(x_1)\}$, $\delta[B(2)(E(A(x_2, x_1)))] = \{E(x_1)\}$.

(vii) $\delta[C(j)(E(D(j)))] = \{F\}$, $j = 1, 2$.

It is easy to check that **A** accepts only trees of the form $t_1(x_1 \leftarrow t_2)$ as in (1). When reading a tree of the above form, **A** first stores t_2 to the tree stack using rules (i) and (ii) and just changes the names of the input symbols. The automaton reads the first $b(j)$ symbol of t_1 using the rule (iii) and then using rule (v) "pops" the jth subtree of the tree stack. Then **A** continues by reading the next symbol $b(j')$, popping the j'th subtree using the appropriate rule of (vi). (Note that after the first pop, the root of the stack was labeled by E.) After reading $c(j)$ at the root of t_1 with rule (iv), using reductions (vii), **A** is able to check whether the node $P(t_1)$ of t_2 was labeled by $d(j)$, $j \in \{1, 2\}$. (If, for instance, $P(t_1)$ does not belong to $\text{dom}(t_2)$, then the computation comes to a deadlock with a tree stack of the form $B(j_1)(E(D(j_2)))$ when **A** has read from t_1 the symbols that correspond to the longest prefix of $P(t_1)$ belonging to $\text{dom}(t_2)$.)

Clearly the reductions of δ are of the form given in (c′) of Definition 4.1 and it is also immediate that **A** is deterministic. ∎

4.6. Lemma. *Let Σ and L be as in the above lemma. Then L does not belong to the family* $\mathscr{L}$(lin-ba-dtpa).

Proof. Suppose that $L = L(\mathbf{A})$, where $\mathbf{A} = (\Sigma, \Gamma, \Gamma', \delta) \in$ lin-ba-dtpa. Let

$$c_1 = \max\{r(\gamma) \mid \gamma \in \Gamma\}$$

and

$$c_2 = \max\{\text{hg}(s) \mid s \in F_\Gamma(X), \delta(s) \neq \varnothing\}.$$

Let $t \in F_\Gamma$ and $u \in \text{dom}(t)$. We say that the node u of t is *dormantly active* if there exists a (Σ, Γ)-configuration C that has t as a tree stack (i.e., C has a subtree of the form $\nabla(t)$) and in the computation of **A** on C at some point the distance of the node that "corresponds to the node u of t" from the reading head ∇ is less than $c_2 + 1$. "Nodes that correspond to the node u of t" are defined as follows: Suppose that t is reduced using a rule $\nabla(s) \to \nabla(\sigma(x_1, ..., x_m))$, $s \in F_\Gamma(X_m)$, $t = \alpha(s)$ where α is a substitution. If the address of x_i in s is not a prefix of u for any $i \in \{1, ..., m\}$, then in the stack $\nabla(\alpha(\sigma(x_1, ..., x_m)))$ no node corresponds to u. (In this case the distance of u from ∇ is necessarily less than $c_2 + 1$.) If $\text{lab}(s, u_1) = x_i$, $i \in \{1, ..., m\}$, and $u = u_1 u_2$, then the node of $\nabla(\alpha(\sigma(x_1, ..., x_m)))$ that corresponds to u is iu_2. If **A** reads a node n of C using a shift rule and $\nabla(t)$ is the jth successor of n, then in the thus obtained tree stack the node corresponding to u in t is ju. Note that if u is not dormantly active, then for any configuration C with tree stack t the reading head cannot "come so close to u" that the subtree $t(u)$ could have any effect on the computation. (A is not able to check the equality of arbitrarily large subtrees of the stack because all the left-hand sides of reductions of A are linear.)

Choose

$$k = \exp(c_1, c_2).$$

(We denote $\exp(m_1, m_2) = (m_1)^{m_2}$.) Let $t \in F_\Gamma$ be arbitrary. Since A does not have rules that delete variables, in any computation step A cannot delete from a stack any subtrees of height greater than c_2. From this it follows that for any given (Σ, Γ)-configuration C having t as a tree stack, all the nodes of t that at some point of the computation of A on C come to a distance of at most c_2 from the reading head, belong to the domain of a k-passable supertree of t. Hence, all the dormantly active nodes of t belong to the supremum of these supertrees, $S(t, k)$. Here $S(t, k) \in F_\Omega$, where $\Omega = \Gamma \cup \bar{\Gamma}$ and the rank of elements of $\bar{\Gamma}$ is zero.

By Lemma 3.3 there exists a polynomial p_1 (of rank $(c_1 + 2)k$) such that $^{\#}S(t, k) \leqslant p_1(\mathrm{hg}(t))$. Denote $r = S(t, k)$ and let $w(r)$ be the word over the alphabet $\Gamma \cup \bar{\Gamma} \cup \{$"(," ")," ",", "$\}$ representing the tree r. Then

$$|w(r)| \leqslant (c_1 + 2)(^{\#}r),$$

and hence there exists a polynomial p such that

$$|w(r)| \leqslant p(\mathrm{hg}(t)). \tag{2}$$

Define K_n, $n \geqslant 0$, to consist of all balanced $\Sigma_2 \cup \Sigma_0$-trees of height n. Then

$$^{\#}K_n = \exp(2, \exp(2, n)).$$

When reading a tree $t_1 \in K_n$ the automaton A reaches the root of t_1 with a tree stack s of height at most n. (A deterministic automaton cannot have reduction rules with elements of X as left-hand sides.) By (2) $|w(S(s, k))| \leqslant p(n)$. (We may choose p to contain only positive coefficients.) The word $w(S(s, k))$ is over the alphabet $\Gamma \cup \bar{\Gamma} \cup \{$"(," ")," ",", "$\}$ and if we choose some symbol to correspond to the empty word, we may assume that $|w(S(s, k))| = p(n)$ for all trees s of height at most n. Denote by S_n the number of different words $w(S(s, k))$ corresponding to a tree stack s of height at most n. Then

$$S_n \leqslant \exp(2(^{\#}\Gamma) + 4, p(n)). \tag{3}$$

Let $n' \in N$ be such that

$$\exp(2, \exp(2, n')) > \exp(2(^{\#}\Gamma) + 4, p(n')).$$

Now by (3), $^{\#}K_{n'} > S_{n'}$. Hence there exist $r_1, r_2 \in K_{n'}$, $r_1 \neq r_2$, such that A reaches the root of r_i with tree stack s_i, $i=1,2$, and $w(S(s_1,k))=w(S(s_2,k))$. It follows that also

$$S(s_1,k)=S(s_2,k). \tag{4}$$

Without restriction we may assume that there exists $u \in \{1,2\}^*$, $|u|=n'$, such that $\mathrm{lab}(r_i,u)=d(i)$, $i=1,2$. Choose

$$t_1=c(1)\,b(j_{n'})\cdots b(j_1)(x_1),$$

where $j_1\cdots j_{n'}=u$. Now $t_1(x_1\leftarrow r_1)$ is in L and $t_1(x_1\leftarrow r_2)$ does not belong to L. This is a contradiction since by (4),

$$t_1(x_1\leftarrow r_1)\in L(\mathbf{A}) \qquad \text{iff} \quad t_1(x_1\leftarrow r_2)\in L(\mathbf{A}). \ \blacksquare$$

4.7. Theorem. $\mathscr{L}$(lin-ba-dtpa) $\subset \mathscr{L}$(lin-del-dtpa).

Proof. Clearly $\mathscr{L}$(lin-ba-dtpa) $\subseteq \mathscr{L}$(lin-del-dtpa), and hence the claim follows from Lemmas 4.5 and 4.6. ▮

4.8. Lemma. *Let $\Sigma=\Sigma_1\cup\Sigma_0$, where $\Sigma_1=\{a,b,c,d\}$ and $\Sigma_0=\{e\}$. Define the Σ-forest L by*

$$L=\{t\in F_\Sigma \mid \mathrm{path}(t)=\{ab^nc^nd^ne\} \text{ for some } n\geqslant 1\}.$$

Then $L\in\mathscr{L}$(lin-ex-dtpa).

Proof. The construction of a lin-ex-dtpa **A** recognizing L is very straightforward and we just describe it intuitively. Clearly we can assume that **A** is able to check that the input tree t is such that $\mathrm{path}(t)\in ab^+c^+d^+e$. When reading an input tree of this form the automaton **A** first reads the leaf e and symbols d from the input tree and stores them to a unary tree stack. After reading the first symbol c, **A** makes two copies of the "d-stack" using a reduction with a nonlinear right-hand side and, after this, pops one symbol d from the left copy. Always after reading a new symbol c, **A** pops the left d-stack. (This is done using a reduction rule of the form $\nabla(C(W(D(x_1),x_2)))\to\nabla(W(x_1,x_2))$. Here symbols C and D represent the input symbols c and d, and W is a stack symbol of rank two that is introduced when **A** duplicates the d-stack.) If the numbers of occurrences of c's and d's agree, **A** starts to read symbols b and, similarly, to pop the right d-stack. If when **A** reaches the symbol a marking the root of the input tree all symbols corresponding to d's in the right stack have been popped, then **A** reduces the stack to an accepting symbol. Thus **A** is able to check that also the numbers of occurrences of symbols b and d are equal. ▮

4.9. Lemma. *Let Σ and L be as in Lemma* 4.8. *Then L does not belong to the family $\mathscr{L}$*(lin-del-dtpa).

Proof. Suppose that $L = L(\mathbf{A})$, where $\mathbf{A} = (\Sigma, \Gamma, \Gamma', \delta) \in$ lin-del-dtpa. Let $t \in F_\Sigma$, $r \in \text{con}(\Sigma, \Gamma)$ be arbitrary and suppose that

$$t(\rightarrow_\Delta)^n\, r, \qquad n \geqslant 0.$$

The ranked alphabet Γ might contain symbols of rank greater than one but using induction on n it is easy to verify that the tree stack of r contains only symbols of rank at most one. Note that the right-hand sides of the reductions of a deleting automaton are linear in variables of X and since $\mathbf{A}$ is deterministic all the variables occurring in a right-hand side of a reduction occur also in the left-hand side. Hence the symbols of Γ of rank greater than one cannot be used in any computation of $\mathbf{A}$ on a Σ-tree. Thus we can view the input and the stack as strings and it follows that by simulating the computation of $\mathbf{A}$ a string pushdown automaton would be able to recognize the language $L_1 = \{ed^nc^nb^na \mid n \geqslant 1\}$. This is a contradiction since it is well known that L_1 is not a context-free language. ▮

Now from Lemmas 4.8 and 4.9 it follows that

4.10. Theorem. $\mathscr{L}(\text{lin-del-dtpa}) \subset \mathscr{L}(\text{lin-ex-dtpa})$.

Note that in the proof of Lemma 4.8 the automaton $\mathbf{A}$ does not need any deleting reductions, i.e., reductions where some variable in the left-hand side does not occur in the right-hand side. Hence it follows that also only the presence of duplicating rules would increase the power of the basic tree pushdown automata.

In the proof of Lemma 4.9 we did not need the assumption that $\mathbf{A}$ is linear, hence we have also:

4.11. Corollary. $\mathscr{L}(\text{del-dtpa}) \subset \mathscr{L}(\text{ex-dtpa})$.

We conjecture that Corollary 4.11 holds also for the nondeterministic case, i.e., that $\mathscr{L}(\text{del-tpa}) \subset \mathscr{L}(\text{ex-tpa})$. However, this cannot be proved as simply as Lemma 4.9 since a nondeterministic deleting automaton can reduce a unary tree stack to a nonunary one. Also we conjecture that $\mathscr{L}(\text{ba-tpa}) \subset \mathscr{L}(\text{del-tpa})$. In this case the difficulty in using arguments similar to those of the proof of Lemma 4.6 is that the height of the tree stack of a nondeterministic automaton corresponding to a subtree t of the input tree may be arbitrarily larger than $\text{hg}(t)$.

5. Monadic Tree Rewriting Systems

In this section we study congruence classes of monadic tree rewriting systems that were defined in [9] as a generalization of monadic Thue systems, cf. [5–7]. In [9] it was shown that finite unions of congruence classes of a terminating Church–Rosser monadic tree rewriting system can be recognized by a deterministic tree pushdown automaton that is able to check whether the final contents of the tree stack belong to a given regular forest. The automaton model used in [9] is equivalent to our ex-tpa. Here we show that this can be done by a lin-ex-dtpa without the additional acceptance condition. For right-linear monadic rewriting systems we have the stronger result that the union of congruence classes can be over an arbitrary regular forest and, furthermore, in this case we can use a deleting tree pushdown automaton instead of an extended one.

5.1. DEFINITION. Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$. The rewriting system S (or $\rightarrow_S$) is said to be *monadic* if S is left-linear and for every pair $(s, t) \in S$:

(i) $\operatorname{var}(t) \subseteq \operatorname{var}(s)$ and

(ii) $\operatorname{hg}(s) \geqslant 1$ and $1 \geqslant \operatorname{hg}(t)$.

The system S is *right-linear monadic* if, additionally, the right-hand sides of the rules of S are of the form $\sigma(z_1, ..., z_k)$ or x, $k \geqslant 0$, $\sigma \in \Sigma_k$, $z_1, ..., z_k$, $x \in X$, and $z_i \neq z_j$ when $i \neq j$.

Condition (ii) of the above definition does not allow rules of the form (σ_1, σ_2) or (x, σ_2), $\sigma_1, \sigma_2 \in \Sigma_0$, $x \in X$. However, it is easy to see that if S is a monadic system that may contain also rules of the forms (σ_1, σ_2) and (x, σ_2), then there exists a monadic system S' as in Definition 5.1 such that $\rightarrow_S$ and $\rightarrow_{S'}$ differ at most on a finite subset of $F_\Sigma \times F_\Sigma$. Also, in the following we are mostly concerned with terminating rewriting systems which in any case could not have rules of the form (x, σ_2).

Next we define the bottom-up normal rewriting relation essentially as in [9].

5.2. DEFINITION. Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be a finite rewriting system and let θ be a total order on S. The *$(S, \theta$-$)$ normal rewrite relation* $\rightarrow_{n,S,\theta}$ $(\subseteq \rightarrow_S)$ is defined as follows:

Let $t, t' \in F_\Sigma$. Then $t \rightarrow_{n,S,\theta} t'$ if we can write

$$t = p(u \leftarrow \alpha(s_1)) \qquad \text{and} \qquad t' = p(u \leftarrow \alpha(s_2)),$$

where $u \in \operatorname{dom}(p)$, $(s_1, s_2) \in S$, α is a substitution $\operatorname{var}(s_1) \cup \operatorname{var}(s_2) \rightarrow F_\Sigma$ and the following conditions hold:

(i) For all proper successors v of u such that $v \in \operatorname{dom}(t)$, $t(v)$ is not unifiable with any left-hand side of a rule of S.

(ii) (s_1, s_2) is in the ordering θ the greatest element of S for which $t(u)$ is unifiable with the left-hand side.

The relation $\rightarrow_{n,S,\theta}$ is denoted by $\rightarrow_n$ if S and θ are clear from the context.

The following lemma follows immediately from the above definition and Lemma 5.4 is proved easily similarly as the corresponding result in [9].

5.3. LEMMA. *Let S be a finite monadic tree rewriting system. Suppose that*

$$t_1 = p(u \leftarrow \alpha(s_1)) \rightarrow_n p(u \leftarrow \alpha(s_2)) = t_2, \qquad (s_1, s_2) \in S.$$

Then for all proper successors v_i of u such that $v_i \in \mathrm{dom}(t_i)$, $t_i(v_i) \in \mathrm{IRR}(S)$, $i \in \{1, 2\}$. (Note that since s_2 can have elements of Σ_0 as proper subtrees, here we need the assumption that S does not have rules with left-hand side of height zero.)

5.4. Lemma. *Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be a finite canonical rewriting system and θ a total order on S. Then for all $t \in F_\Sigma$ there exists a unique $t' \in \mathrm{IRR}(S)$ such that*

$$t \rightarrow^*_{n,S,\theta} t'.$$

For the proof of the main result of this section we still need the following definition.

5.5. Definition. Let Σ be a ranked alphabet and $p \in N \cup \{0\}$. We denote by $F_{p,\Sigma,X}$ the set of all ΣX-trees t such that

(i) $\mathrm{hg}(t) \leqslant p$,

(ii) if $u \in \mathrm{dom}(t)$, then $\mathrm{lab}(t, u) \in X$ iff $|u| = p$.

The trees of $F_{p,\Sigma,X}$ are called *p-truncated ΣX-trees*. The set of *ordered p-truncated ΣX-trees* $F'_{p,\Sigma,X}$ consists of all trees $t \in F_{p,\Sigma,X}$ such that the following condition holds:

Let $u_1, \ldots, u_r$, $r \geqslant 0$, be the nodes of $\mathrm{dom}(t)$ labeled by variables of X, $u_1 <_{lr} \cdots <_{lr} u_r$. Then $\mathrm{lab}(t, u_i) = x_i$, $i = 1, \ldots, r$.

Let $k \geqslant 0$, $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in F_{n,\Sigma,X}$, where $n = p-1$ or $n = p$, $p \geqslant 1$. We define the ordered p-truncated tree

$$\sigma[t_1, \ldots, t_k]_p \ (\in F'_{p,\Sigma,X})$$

as follows. Denote $s = \sigma[t_1, \ldots, t_k]_p$. Then

(i) $\mathrm{dom}(s) = \{\lambda\} \cup \{iu \mid u \in \mathrm{dom}(t_i), 1 \leqslant i \leqslant k, |u| < p\}$.

(ii) $\mathrm{lab}(s, \lambda) = \sigma$.

(iii) $\mathrm{lab}(s, iu) = \mathrm{lab}(t_i, u)$ if $u \in \mathrm{dom}(t_i)$, $|u| < p-1$.

(iv) Let $v_1, \ldots, v_r$, $r \geqslant 0$, be the nodes of s of length p, where $v_1 <_{lr} v_2 <_{lr} \cdots <_{lr} v_r$. Then $\mathrm{lab}(s, v_i) = x_i$, $i = 1, \ldots, r$.

5.6. Example. Let $\Sigma = \Sigma_2 \cup \Sigma_1 \cup \Sigma_0$, where $\Sigma_2 = \{a, b, c\}$, $\Sigma_1 = \{d\}$ and $\Sigma_0 = \{e\}$ and let $p = 3$. Let r and s be the p-truncated ΣX-trees of Fig. 2(a). Then the tree $a[r, s]_3$ is depicted in Fig. 2(b).

5.7. Lemma. *Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be a finite monadic rewriting system, $\theta \subseteq S \times S$ a total order and let $R \subseteq F_\Sigma$ be a regular forest. Define*

$$H = \{t \in F_\Sigma \mid (\exists t' \in \mathrm{IRR}(S) \cap R) \colon t \rightarrow^*_{n,S,\theta} t'\}.$$

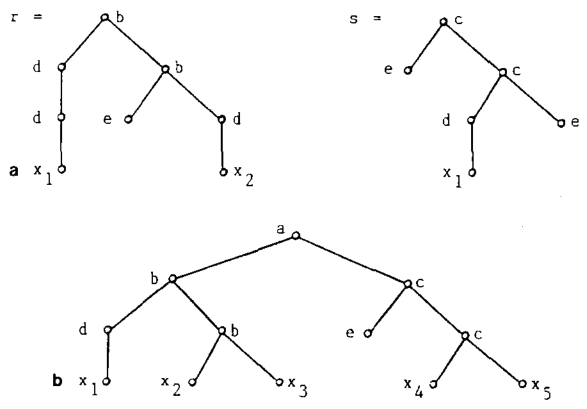

FIGURE 2

Then

$$H \in \mathscr{L}(\text{lin-ex-dtpa}).$$

Proof. Let $\mathbf{A} = (\Sigma, Q, Q_F, g)$ be a deterministic bottom-up tree recognizer such that $L(\mathbf{A}) = \mathrm{IRR}(S) \cap R$, the recognizer $\mathbf{A}$ exists since $\mathrm{IRR}(S)$ is regular, cf. [9, 4]. We define a lin-ex-dtpa

$$\mathbf{B} = (\Sigma, \Gamma, \Gamma', \delta)$$

such that $L(\mathbf{B}) = H$. The intuitive idea of the construction is roughly as follows. The elements of the ranked stack alphabet Γ consist of two components. After reading a node from the input tree using a shift-rule, $\mathbf{B}$ reduces the stack as in a normal reduction. This is possible by storing in the first component of the element of Γ labeling the root of the stack the information about a "large enough supertree" of the tree stack. The second component of the label of the root of a stack contains the states of the recognizer $\mathbf{A}$ that are reached at the nodes immediately below the root. Thus when $\mathbf{B}$ reaches the root of an input tree t, the stack contains a tree t' that is obtained from t in a normal reduction, and using the information in the second component of the root of t' $\mathbf{B}$ is able to check whether t' is in $\mathrm{IRR}(S) \cap R$. So the main difficulty lies in defining the rules of δ so that the information on labels of nodes in the stack is always correctly updated in the shift and reduce operations on the stack.

We now proceed with the construction of $\mathbf{B}$. Let

$$p = \max\{\mathrm{hg}(s) \mid (\exists t \in F_\Sigma(X)) \colon (s, t) \in S\} + 2$$

and let $Y = \{y_1, y_2, \ldots\}$ be a denumerable set of variables disjoint from X. The set S defines in the natural way also a rewriting relation on $F_\Sigma(Y)$ (one just views the elements of Y as constants), and in the following we assume that $\rightarrow_S \subseteq F_\Sigma(Y) \times F_\Sigma(Y)$. Let

$$W = \{t \in F_\Sigma(Y) \mid t(u) \in \mathrm{IRR}(S) \text{ for all } u \in \mathrm{dom}(t) \text{ such that } |u| > 0\},$$
$$M = \max\{r(\sigma) \mid \sigma \in \Sigma\}.$$

The ranked alphabet Γ consists of all pairs $(t, \bar{q})$ such that $t \in F'_{p,\Sigma,Y} \cap W$ and $\bar{q} \in Q^i$, where $i = r(\mathrm{lab}(t, \lambda))$. (Note that Γ is finite since in trees of $F'_{p,\Sigma,Y}$ the elements of Y occur in order $y_1, y_2, \ldots$ with respect to the relation $<_{lr}$.) The ranks of the elements of Γ are determined by

$$r((t, \bar{q})) = r(\mathrm{lab}(t, \lambda))$$

for all $t \in F'_{p,\Sigma,Y} \cap W$ and $\bar{q} \in Q^i$, $0 \leqslant i \leqslant M$. Further, let

$$\Gamma' = \{(t, \bar{q}) \in \Gamma \mid [\mathrm{lab}(t, \lambda)]_g(\bar{q}) \in Q_F\}.$$

Note that if $(t, \bar{q}) \in \Gamma$, then t is reducible mod S iff $t = \alpha(s)$, where s is a left-hand side of a rule of S and α is a substitution. Hence, it follows that the rule that reduces t in a normal rewrite step is unique (if it exists). This observation will be used many times in the following. The relation δ is defined in (i), (ii), (iiia), and (iiib):

(i) For every $\sigma \in \Sigma_0$,

$$\delta(\sigma) = \{(\sigma, \phi)\}.$$

(Here ϕ stands for the "0-tuple" of elements of Q.)

(ii) Suppose that $k \geqslant 1$, $\sigma \in \Sigma_k$, $e_i = (f_i, \bar{q}_i) \in \Gamma$, $i = 1, \ldots, k$. If $f_i \in \mathrm{IRR}(S)$, $i = 1, \ldots, k$, then

$$\delta(\sigma, e_1, \ldots, e_k) = \{(\sigma[f_1, \ldots, f_k]_p, [\mathrm{lab}(f_1, \lambda)]_g(\bar{q}_1), \ldots, [\mathrm{lab}(f_k, \lambda)]_g(\bar{q}_k))\}.$$

We define the tree homomorphism $h: F_\Gamma(X) \rightarrow F_\Sigma(X)$ by

$$h_m((t, \bar{q})) = (\mathrm{lab}(t, \lambda))(\xi_1, \ldots, \xi_m)$$

for all $m \geqslant 0$, $(t, \bar{q}) \in \Gamma_m$. (Here $\xi_1, \ldots, \xi_m$ are as in Definition 2.2.)

Let Π_1 and Π_2 denote respectively the projection functions $\Gamma \rightarrow (F'_{p,\Sigma,Y} \cap W)$ and $\Gamma \rightarrow \cup(Q^i \mid 0 \leqslant i \leqslant M)$. Further let $\Pi(i)$ be the ith projection $Q^i \rightarrow Q$. Now we are ready to define the reductions of δ.

(iiia) Let $r \in F_\Gamma(X_m)$, $s \in F_\Sigma(X_m)$, $h(r) = s$, and $(s, t) \in S$, where $t = \sigma(z_1, \ldots, z_k)$, $k \geqslant 0$, $\sigma \in \Sigma_k$, $z_1, \ldots, z_k \in X_m \cup \Sigma_0$. Suppose that (s, t) is the unique rule to be applied to $\Pi_1(\mathrm{lab}(r, \lambda))$ in a normal rewrite step. Then

$$(\sigma[X^{1,1}, \ldots, X^{1,k}]_p, X^{2,1}, \ldots, X^{2,k})(\bar{z}_1, \ldots, \bar{z}_k) \in \delta(r).$$

Here the symbols $X^{i,j}$ and $\bar{z}_j$ are defined as follows:

Without loss of generality we may assume that s, and hence also r, contains all the variables $x_1, ..., x_m$ exactly once; let their addresses in s and r be $u_1, ..., u_m$. (Note that the addresses are same in s and r since h is only a renaming of function symbols.) Let

$$u_i = u_i' v_i, \qquad v_i \in N,$$

$i = 1, ..., m$. For $j = 1, ..., k$ define

$$X^{1,j} = \begin{cases} [\Pi_1(\text{lab}(r, u'_{(i,j)}))](v_{(i,j)}) & \text{if } z_j = x_{(i,j)},\ 1 \leqslant (i,j) \leqslant m. \\ \tau & \text{if } z_j = \tau \in \Sigma_0. \end{cases}$$

(Note that $X^{1,j} \in F_{p-1}, \Sigma, Y \cap \text{IRR}(S)$ but $X^{1,j}$ does not necessarily belong to $F'_{p-1,\Sigma,Y}$.)

$$X^{2,j} = \begin{cases} \Pi(v_{(i,j)})[\Pi_2(\text{lab}(r, u'_{(i,j)}))] & \text{if } z_j = x_{(i,j)},\ 1 \leqslant (i,j) \leqslant m. \\ \tau_g & \text{if } z_j = \tau \in \Sigma_0. \end{cases}$$

$$\bar{z}_j = \begin{cases} z_j & \text{if } z_j \in X. \\ (\tau, \phi) & \text{if } z_j = \tau \in \Sigma_0. \end{cases}$$

(iiib) Let $(s, t) \in S$, where $s \in F_\Sigma(X_m)$ and $t = x_i$, $1 \leqslant i \leqslant m$. Let $k \geqslant 0$ and $\sigma \in \Sigma_k$ be arbitrary and let $r \in F_\Gamma(X_{m+k})$ be such that

$$h(r) = s(x_i \leftarrow \sigma(x_{m+1}, ..., x_{m+k})).$$

Suppose that (s, t) is the unique rule to be applied to $\Pi_1(\text{lab}(r, \lambda))$ in a normal rewrite step and that u_i is the address of x_i in s. Then

$$[\text{lab}(r, u_i)](x_{m+1}, ..., x_{m+k}) \in \delta(r).$$

This completes the definition of **B**, and **B** is clearly left-linear. It is easy to see that **B** is deterministic: Condition (a) of Definition 4.3 follows from the definition of shift-rules (i) and (ii). Condition (b) of 4.3 holds since a shift-rule (ii) can be applied only if the first components of the labels of the roots of the tree stacks are irreducible mod S and in this case no reduction is possible. Also, clearly $\delta(x) = \varnothing$ for all $x \in X$. Condition (c) follows immediately from (iiia) and (iiib). For condition (d), suppose that $r_1, r_2 \in F_\Gamma(X)$, $r_1 \downarrow_u r_2$, and $\delta(r_i) \neq \varnothing$, $i = 1, 2$. Hence $h(r_i)$ is a left-hand side of a rule of S that reduces $\Pi_1(\text{lab}(r_i, \lambda))$ in a normal reduction or $h(r_i)$ is as in (iiib), $i = 1, 2$. Since r_1 and r_2 are unifiable, it follows that $\text{lab}(r_1, \lambda) = \text{lab}(r_2, \lambda)$ and $h(r_1) = h(r_2)$. Furthermore, since h maps every variable of X onto itself, it follows that $r_1 = r_2$ and thus (d) holds.

We say that $r \in F_\Gamma$ is *well behaved* if for every subtree r_1 of r.

(1) $h(r_1) \downarrow_u \Pi_1(\text{lab}(r_1, \lambda))$ and

(2) $[\text{lab}(\Pi_1(\text{lab}(r_1, \lambda)), \lambda)]_g (\Pi_2(\text{lab}(r_1, \lambda))) = (h(r_1))_g$.

(Note that since r_1 does not contain variables, (1) is equivalent to the condition that there exists a substitution $\alpha\colon Y \to F_\Gamma$ such that $h(r_1) = \alpha(\Pi_1(\mathrm{lab}(r_1, \lambda)))$, i.e., $\Pi_1(\mathrm{lab}(r_1, \lambda))$ is a "supertree" of $h(r_1)$.) Also we say that a (Σ, Γ)-configuration t is well behaved if every subtree t_1 of t such that $t_1 \in F_\Gamma$ is well behaved.

We extend the tree homomorphism h to

$$\bar{h}\colon F_{\Sigma \cup \Gamma \cup \{\nabla\}}(X) \to F_\Sigma(X)$$

by defining

(i) $\bar{h}_m(\sigma) = \sigma(\xi_1, ..., \xi_m)$ for all $m \geqslant 0$, $\sigma \in \Sigma_m$;
(ii) $\bar{h}_1(\nabla) = \xi_1$.

We denote also the extension $\bar{h}$ of h simply by h. We now prove a number of claims.

Claim 1. *Let $t \in F_\Sigma$ and suppose that*

$$t \to_\Delta^* r\ (\in \mathrm{con}(\Sigma, \Gamma)).$$

Then r is well behaved.

Proof of Claim 1. This is seen easily by induction on the length of the derivation. (In the rules of **B** the first component of the label of the root of the tree stack is always updated to contain the "supertree" of height p of the tree stack. Similarly the last components of a symbol of Γ labeling a node u are updated to contain the states of **A** that are reached at the immediate successors of u.)

Claim 2. *Let $t \in F_\Sigma$ and suppose that $t(\to_\Delta)^m\, r$, $m \geqslant 0$. Let $r_1 \in F_\Gamma$ be a tree stack of r (i.e., $r_1 \in \mathrm{sub}(r)$ and the predecessor of the root of r_1 is labeled by ∇). Then every proper subtree of $h(r_1)$ is irreducible* mod S.

Proof of Claim 2. Suppose that some proper subtree of $h(r_1)$ is reducible mod S. Hence we can choose a proper subtree s of $h(r_1)$ such that s is unifiable with a left-hand side of a rule of S. There exists a proper subtree r_2 of r_1 such that $h(r_2) = s$. Suppose that $r_2 = r_1(uv)$, $u \in N^*$, $v \in N$. Now by Claim 1,

$$h(r_1(u)) \downarrow_u \Pi_1(\mathrm{lab}(r_1(u), \lambda))$$

and because $[h(r_1(u))](v) = h([r_1(u)](v)) = h(r_2)$ we have that $[\Pi_1(\mathrm{lab}(r_1(u), \lambda))](v)$ is reducible mod S. (Note that here we need the information that p is the maximum height of the left-hand sides of S plus two.) This is a contradiction since $\Pi_1(\mathrm{lab}(r_1(u), \lambda)) \in W$.

Claim 3. *Let $r \in F_\Sigma$ and suppose that*

$$r(\to_\Delta)^m\, r', \qquad r' \in \mathrm{con}(\Sigma, \Gamma), m \geqslant 0.$$

Then

$$r = h(r) \to_n^* h(r').$$

Proof of Claim 3. If $m = 0$ there is nothing to prove. Suppose that the claim holds for $m = k$ and

$$r(\to_\Delta)^k s \to_\Delta r'.$$

By the induction assumption $r \to_n^* h(s)$. If r' is obtained from s using a shift-rule (i) or (ii), we are done since then $h(s) = h(r')$. Suppose then that

$$s = s'(u \leftarrow \nabla(t_1)) \to_\Delta s'(u \leftarrow \nabla(t_2)) = r'$$

using a reduction rule. By Claim 2 all proper subtrees of $h(t_1)$ are irreducible mod S. Hence from Claim 1 and the definition of reduction rules of **B** it follows immediately that $h(t_1) \to_n h(t_2)$ and hence also $h(s) \to_n h(r')$.

CLAIM 4. *Let* $r, r' \in F_\Sigma$ *and suppose that*

$$r(\to_n)^m r', \qquad m \geqslant 0.$$

Then there exists $s \in \text{con}(\Sigma, \Gamma)$ *such that* $h(s) = r'$ *and*

$$r \to_\Delta^* s.$$

Proof of Claim 4. In the case $m = 0$ we can choose $s = r$. Suppose that the claim holds for $m = k$ and that

$$r(\to_n)^k r_1 \to_n r'.$$

Hence there exists $s_1 \in \text{con}(\Sigma, \Gamma)$ such that $h(s_1) = r_1$ and $r \to_\Delta^* s_1$. Suppose that we can write

$$r_1 = t(u \leftarrow \alpha(t_1)) \to_n t(u \leftarrow \alpha(t_2)) = r', \qquad (t_1, t_2) \in S.$$

Since $\alpha(t_1)$ is reducible mod S it follows by Claim 2 that either in s_1 no node v such that v is a predecessor of u is labeled by ∇ or $\text{lab}(s_1, u) = \nabla$. Since by Lemma 5.3 all proper subtrees of $\alpha(t_1)$ are irreducible mod S, it follows from Claim 1 that in the first case $s_1 \to_\Delta^* s_1'$, using only the shift-rules of **B** where $\text{lab}(s_1', u) = \nabla$ (i.e., the automaton just advances the reading heads to the root of the subtree of s_1 that corresponds to $\alpha(t_1)$ in r_1). In the latter case we choose $s_1' = s_1$. By Claim 1, $s_1'(u1)$ is well behaved so using a rule of δ that is obtained from (t_1, t_2) by (iiia) or (iiib) we get $s_1' \to_\Delta s$, where $h(s) = r'$.

From Claims 3 and 4 it follows that for all $r, r' \in F_\Sigma$:

(3) $r \to_n^* r'$ iff there exists $s \in \text{con}(\Sigma, \Gamma)$ such that $h(s) = r'$ and $r \to_\Delta^* s$.

Furthermore, if $r' \in \mathrm{IRR}(S)$, then s can be chosen to be such that $\mathrm{lab}(s, \lambda) = \nabla$. (Using the shift-rules of **B** we can move the reading heads to the root of s.) From Claim 1 and equation (2) it follows that for $s = \nabla(s_1)$ as above

(4) $\mathrm{lab}(s_1, \lambda) \in \Gamma'$ iff $r' \in R \cap \mathrm{IRR}(S)$.

Now (3) and (4) imply that $L(\mathbf{B}) = H$. ∎

5.8. Theorem. *Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be a finite canonical monadic rewriting system. Then for every regular forest $R \subseteq F_\Sigma$,*

$$K = [R \cap \mathrm{IRR}(S)] \in \mathscr{L}(\text{lin-ex-dtpa}).$$

Proof. Let $\theta \subseteq S \times S$ be a total order. Since S is terminating and Church–Rosser, $t \in K$ iff $(\exists t' \in \mathrm{IRR}(S) \cap R)$: $t \to^*_{n,S,\theta} t'$ by Lemma 5.4. Hence $K \in \mathscr{L}(\text{lin-ex-dtpa})$ by Lemma 5.7. ∎

Now as in [5, 9] it follows that

5.9. Corollary. *Let S be a finite canonical monadic tree rewriting system. Then every finite union of congruence classes of S belongs to the family $\mathscr{L}$(lin-ex-dtpa).*

As a conclusion we consider monadic systems that are also right-linear. If S and H are as in Lemma 5.7 and S is right-linear, then from the proof of Lemma 5.7 it follows (by renaming variables) that $H \in \mathscr{L}$(lin-del-dtpa). Hence, we have

5.10. Corollary. *Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be a finite right-linear rewriting system that is monadic and canonical. Then for every regular forest $R \subseteq F_\Sigma$,*

$$[R \cap \mathrm{IRR}(S)] \in \mathscr{L}(\text{lin-del-dtpa}).$$

5.11. Lemma. *Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be a right-linear monadic set of rules (not necessarily finite). Then for every regular forest $L \subseteq F_\Sigma$, $S^*(L)$ is regular.*

Proof. By renaming variables we may assume that the rules of S are of the form

$$r \to \sigma(x_1, \ldots, x_n) \qquad \text{or} \qquad r \to x_1,$$

where $m \geqslant n \geqslant 0$, $\sigma \in \Sigma_n$, $r \in F_\Sigma(X_m)$, and every variable of X_m occurs exactly once in r. If $r \to x_1 \in S$, then $m \geqslant 1$.

Let $\mathbf{A} = (\Sigma, Q, Q_F, g)$ be a nondeterministic connected bottom-up tree recognizer such that $L(\mathbf{A}) = L$. We define the functions (g, i), $i \geqslant 0$, that associate with each element $\sigma \in \Sigma_n$ $(n \geqslant 0)$ a mapping

$$\sigma_{(g,i)}: Q^n \to \mathscr{P}(Q)$$

as follows:

(i) $(g, 0) = g$.

(ii) Let $i > 0$, $n \geqslant 0$, $\sigma \in \Sigma_n$, and $a_1, \ldots, a_n \in Q$. Denote

$N(i, \sigma, a_1, \ldots, a_n)$

$= \cup ([r(x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n, x_{n+1} \leftarrow b_{n+1}, \ldots, x_m \leftarrow b_m)]_{(g,i-1)} \mid m \geqslant n, r \in F_\Sigma(X_m),$

$r \rightarrow \sigma(x_1, \ldots, x_n) \in S, b_{n+1}, \ldots, b_m \in Q)$

$\cup ([r(x_1 \leftarrow c_1, x_2 \leftarrow b_2, \ldots, x_m \leftarrow b_m)]_{(g,i-1)} \mid m \geqslant 1, r \in F_\Sigma(X_m), (r \rightarrow x_1) \in S,$

$c_1 \in \sigma_{(g,i-1)}(a_1, \ldots, a_n), b_2, \ldots, b_m \in Q).$

Then we define

$$\sigma_{(g,i)}(a_1, \ldots, a_n) = \sigma_{(g,i-1)}(a_1, \ldots, a_n) \cup N(i, \sigma, a_1, \ldots, a_n).$$

Now for all $n \geqslant 0$, $\sigma \in \Sigma_n$, $a_1, \ldots, a_n \in Q$,

$$\sigma_{(g,i-1)}(a_1, \ldots, a_n) \subseteq \sigma_{(g,i)}(a_1, \ldots, a_n),$$

and since Q and Σ are finite there exists a natural number k_0 such that

$$(g, k) = (g, k+1) \text{ for all } k \geqslant k_0. \qquad (1)$$

We choose $\mathbf{B} = (\Sigma, Q, Q_F, (g, k_0))$ and show that

$$L(\mathbf{B}) = S^*(L). \qquad (2)$$

(Intuitively **B** is obtained from **A** by iteratively adding rules that on right-hand sides of rules of S simulate the computation on the corresponding left-hand sides.)

Let $t \in F_\Sigma$, $u \in \mathrm{dom}(t)$ and $\mathrm{lab}(t, u) = \sigma \in \Sigma_n$. Let C be a computation of **B** on t. We say that the node u is computed in C using a *proper* (g, i)-*rule*, $0 \leqslant i \leqslant k_0$, if in the computation C the rule applied at u is $\sigma(q_1, \ldots, q_n) \rightarrow q$ $(q_1, \ldots, q_n, q \in Q)$, where

$$q \in \sigma_{(g,i)}(q_1, \ldots, q_n) - \sigma_{(g,i-1)}(q_1, \ldots, q_n).$$

(We interpret that $\sigma_{(g,-1)}(q_1, \ldots, q_n) = \varnothing$.) If $i \in \{0, \ldots, k_0\}$ is the maximal number such that proper (g, i)-rules are used in the computation C and if $w > 0$ is the number of applications of proper (g, i)-rules in C, then we say that C is a (g, i, w)-*computation*. We say that t has an accepting (g, i)-computation if for some $w > 0$ there is an accepting (g, i, w)-computation on t.

Now we prove (2). First let $t \in L(\mathbf{B})$ and let C_1 be an accepting (g, i, w)-computation of **B** on t, $0 \leqslant i \leqslant k_0$, $w > 0$.

(i) If $i = 0$, then $t \in L$.

(ii) Let $i > 0$ and $w \geqslant 2$. From the definition of (g, i) it follows that there exists a Σ-tree t' such that $t' \rightarrow_S t$ and there exists an accepting $(g, i, w-1)$-computation of **B** on t'. Note that here we need the information that S is right-linear and that **A** is connected.

(iii) Let $i > 0$ and $w = 1$. Similarly as in (ii) it is seen that there exists t' such that $t' \rightarrow_S t$ and t' has an accepting $(g, i-1, w')$-computation, $w' \in N$.

Applying (ii) and (iii) iteratively we see that there exists a Σ-tree t_1 and $w_1 \geqslant 1$ such that $t_1 \rightarrow_S^* t$ and there is an accepting $(g, 0, w_1)$-computation of $\mathbf{B}$ on t_1. (Each application of (ii) and (iii) can increase the size of the tree by an arbitrary amount. However, the process terminates always as is seen by the following: If $i = 0$, there is nothing to prove. Suppose then that $i = i_0 + 1$ and that the process terminates for all Σ-trees with accepting (g, i_0, w_0)-computations, $w_0 > 0$. After $w - 1$ applications of (ii) and one application of (iii) we have $t'' \in F_\Sigma$ such that $t''(\rightarrow_S)^w\, t$ and t'' has an accepting (g, i_0, w'')-computation, $w'' > 0$.) Hence $t_1 \in L$ and $t \in S^*(L)$.

For the converse inclusion $S^*(L) \subseteq L(\mathbf{B})$ it is easy to see that if $t \rightarrow_S t'$ and there exists an accepting (g, i)-computation on t, then there exists an accepting (g, i')-computation on t', where $i' \leqslant i + 1$. Here we need the fact that monadic rules are also left-linear. This is because 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 (g, i)-computation of t. (Note that it may be also that $i' < i$ since it is possible that all the nodes in t that are computed using proper (g, i)-rules are deleted.) Now the relation $S^*(L) \subseteq L(\mathbf{B})$ follows by (1) and the fact that $L \subseteq L(\mathbf{B})$. ▌

If strings are seen as unary trees, then every monadic Thue system (cf. [5, 7]) is a right-linear monadic tree rewriting system. In [7] it is shown that for context-free monadic Thue systems the set of descendants of a regular set is always regular. Now by Lemma 5.11 we have the stronger result:

5.12. Corollary. *For an arbitrary (possibly infinite) monadic Thue system the set of descendants of a regular set is regular.*

For right-linear monadic systems we can strenghten the result of Corollary 5.9.

5.13. Theorem. *Let $S \subseteq F_\Sigma(X) \times F_\Sigma(X)$ be finite, right-linear, canonical, and monadic. Then for every regular forest $L \subseteq F_\Sigma$,*

$$[L] \in \mathscr{L}(\text{lin-del-dtpa}).$$

Proof. Since S is terminating we have

$$[L] = [S^*(L) \cap \mathrm{IRR}(S)].$$

By Lemma 5.11, $S^*(L)$ is regular and hence $[L] \in \mathscr{L}(\text{lin-del-dtpa})$ by Corollary 5.10. ▌

It should be noted that if S is infinite, then in the proof of Lemma 5.11 the construction of the recognizer for $S^*(L)$ might not be effective. However, from the proofs of Lemmas 5.7 and 5.11 it follows that if S and L are as in Theorem 5.13 (i.e., S is finite), then we can effectively construct a lin-del-dtpa $\mathbf{A}$ such that $[L]_S = L(\mathbf{A})$. It is not difficult to modify the proof of Lemma 5.11 so that one does not need the assumption that the rules of S are left-linear. Lemma 5.11 does not hold for monadic systems that are not right-linear. This is seen by the following example.

5.14. Example. Let $\Sigma = \Sigma_2 \cup \Sigma_1 \cup \Sigma_0$, where $\Sigma_2 = \{\omega\}$, $\Sigma_1 = \{\tau, \sigma\}$ and $\Sigma_0 = \{\gamma\}$. Let $S = \{(\tau(x_1), \omega(x_1, x_1))\}$ and $L = \{\tau(\sigma^n(\gamma)) \mid n \geqslant 0\} \in \text{REG}$. Then $S^*(L) = L \cup \{\omega(\sigma^n(\gamma), \sigma^n(\gamma)) \mid n \geqslant 0\}$, which is not regular.

Acknowledgment

I thank Professor Ronald V. Book for suggesting the topic and for encouragement and support at all stages of the work.

References

1. A. V. Aho, Indexed grammars—An extension of context-free grammars, *J. Assoc. Comput. Mach.* **15** (1968), 647–671.
2. A. Arnold and M. Dauchet, Une théorème de duplication pour les forêts algébriques, *J. Comput. System Sci.* **13** (1976), 223–244.
3. A. Arnold and M. Dauchet, Forêts algébriques et homomorphismes inverses, *Inform. and Control* **37** (1978), 182–196.
4. J. Berstel, Congruences plus que parfaites et langages algébriques, *in* "Seminaire d'Informatique Théorique," Institut de Programmation, 1976–77, pp. 123–147.
5. R. V. Book, Confluent and other types of Thue systems, *J. Assoc. Comput. Mach.* **29** (1982), 171–182.
6. R. V. Book, Thue systems as rewriting systems, *J. Symbolic Comput.* **3** (1987), 39–68.
7. R. V. Book, M. Jantzen, and C. Wrathall, Monadic Thue systems, *Theoret. Comput. Sci.* **19** (1982), 231–251.
8. J. Engelfriet and E. M. Schmidt, IO and OI, Parts 1, 2, *J. Comput. System Sci.* **15** (1977), 328–353; **16** (1978), 67–99.
9. J. H. Gallier and R. V. Book, Reductions in tree replacement systems, *Theoret. Comput. Sci.* **37** (1985), 123–150.
10. F. Gécseg and M. Steinby, "Tree Automata," Akadémiai Kiadó, Budapest, 1984.
11. I. Guessarian, Pushdown tree automata, *Math. Systems Theory* **16** (1983), 237–263.
12. G. Huet, Confluent reductions: Abstract properties and applications to term rewriting systems, *J. Assoc. Comput. Mach.* **27** (1980), 797–821.
13. G. Huet and D. C. Oppen, Equations and rewrite rules, *in* "Formal Language Theory, Perspectives and Open Problems" (R. V. Book, Ed.), pp. 349–393, Academic Press, New York/London, 1980.
14. B. K. Rosen, "Subtree Replacement Systems," Tech. Rep. 2-71, Ctr. for Res. in Computing Technology, Harvard University, 1971.
15. B. K. Rosen, Tree-manipulating systems and Church–Rosser theorems, *J. Assoc. Comput. Mach.* **20** (1973), 160–187.
16. W. C. Rounds, Mappings and grammars on trees, *Math. Systems Theory* **4** (1970), 257–287.
17. K. M. Schimpf, "A Parsing Method for Context-free Tree Languages," Ph. D. dissertation, University of Pennsylvania, 1982.
18. K. M. Schimpf and J. H. Gallier, Tree pushdown automata, *J. Comput. System Sci.* **30** (1985), 25–40.