MATHEMATICAL SYSTEMS THEORY **10**, 289–303 (1977)

Top-down Tree Transducers with Regular Look-ahead

by

JOOST ENGELFRIET*
Twente University of Technology
Enschede, Netherlands

ABSTRACT

Top-down tree transducers with regular look-ahead are introduced. It is shown how these can be decomposed and composed, and how this leads to closure properties of surface sets and tree transformation languages. Particular attention is paid to deterministic tree transducers.

Introduction. The top-down finite state tree transformations discussed in, for example, [3, 7, 13, 14] fail to have certain nice closure properties with respect to composition. It was argued in [7] that this is due to the fact that a top-down tree transducer cannot inspect a subtree before deleting it (a property possessed by bottom-up tree transducers). In this paper we add the facility of regular look-ahead to the usual type of top-down tree transducer. The transducer is now allowed to inspect the subtrees of a node before processing it (thus having an arbitrarily large look-ahead). However, the look-ahead is restricted in that the information which the transducer extracts from the subtrees should be finite and even regular (or "recognizable" i.e. computable by a finite tree automaton). We note that the idea of regular look-ahead also occurs in the theory of parsing of context-free languages [4]. It turns out that the class of tree transformations realized by this type of transducer has all the expected closure properties with respect to composition, for instance, both the classes of linear and deterministic top-down tree transformations are now closed under composition (we note here that composition results can also be proved for restricted types of top-down tree transducers, such as total or nondeleting ones, see [13, 14]). These composition results are proved, as in [7], by first decomposing the transformations into simpler parts and then showing composition properties of these simpler transformations. In fact, any top-down tree transducer with regular look-ahead can be realized in two phases. The first phase (which can be accomplished bottom-up and deterministically) computes all the look-ahead information and stores it in the labels at the nodes of the input tree. The second phase is an ordinary top-down tree transducer which uses this information to imitate the one with regular look-ahead. This decomposition result is also useful in obtaining results about top-down finite state tree transformations without regular look-ahead, in particular concerning their surface sets.

*The research reported here was carried out during a one-year visit of the author to the Dept. of Computer Science of Aarhus University, Aarhus, Denmark.

This paper is a sequel to [7] and the reader is assumed to be familiar with the methods and results of [7]. However, we do not assume any familiarity with section 5 of [7]. We note that the class *T'-FST*, defined in that section, is in fact equal to the class of top-down tree transformations with regular look-ahead (cf. the remarks following [7, Theorem 5.13]).

In section 1 we list some changes in terminology with respect to [7], some additional terminology, and some additional lemmas.

In section 2 we define the top-down tree transducer with regular look-ahead and show the above mentioned decomposition and composition results.

In section 3 we compare the deterministic bottom-up and top-down tree transformations. The deterministic bottom-up tree transformations are (properly) contained in the deterministic top-down tree transformations with regular look-ahead.

In section 4 we apply the results of the previous sections to top-down surface sets and yields of surface sets. It follows for instance from the result in section 3 that the deterministic top-down surface sets are closed under deterministic bottom-up tree transformations. We finally mention possible applications to Lindenmayer languages.

1. Preliminaries

The reader is referred to [7] for all unexplained terminology. That paper will from now on be referred to as [*BT*] rather than [7].

We recall that we often make no explicit distinction between a transducer $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ and the transformation from T_Σ to T_Δ that it computes. We also recall that by the relational composition $R_1 \circ R_2$ we mean "first R_1, then R_2". We finally restate the important properties (*B*1), (*B*2) and (*T*) of [*BT*].

(*B*1) Copying of an output tree after nondeterministic processing of the input tree.

(*B*2) Deciding whether to delete a tree or not after processing it.

(*T*) Copying of an input tree and processing the copies differently.

In the rest of this section we list some changes in and additions to the terminology in [*BT*]. Some additional facts, to be used in later sections, are also mentioned.

First we change our use of "deterministic top-down" and of "*DT*" so as to agree with [13]. A top-down *fst* $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ will be called *deterministic* if (1) Q_d is a singleton and (2) different rules in R have different lefthand sides. It is easy to see that every deterministic top-down *fst* is equivalent to one which processes the whole input tree (except eventually for its leaves) and then decides whether to accept it or not. The version in [*BT*] will be called a *total* deterministic *t-fst* (since it accepts every input tree). Determinism will be denoted as usual by a *D*, so that *DT-FST* denotes the class of deterministic top-down *fst* (and not the class of total deterministic top-down *fst*, as was the case in [*BT*]). The class of linear deterministic top-down *fst* will be denoted by *LDT-FST*. The definition of *HOM* and *LHOM* is not changed, i.e. homomorphisms are total.

Secondly, we shall write *REL* rather than *RELAB*. Thus *QREL* denotes the class of (bottom-up or top-down) finite state relabelings (cf. Definition 3.14 of

[*BT*]). The class of deterministic bottom-up finite state relabelings will be denoted by *DBQREL* (this class was denoted *DQRELAB* in [*BT*]), and the class of deterministic top-down finite state relabelings by *DTQREL*.

We shall use the following additional decomposition results, the detailed proof of which is left to the reader.

LEMMA 1.1.

(1) $$T\text{-}FST \subseteq HOM \circ LT\text{-}FST$$

$$DT\text{-}FST \subseteq HOM \circ LDT\text{-}FST,$$

(2) $$LDT\text{-}FST \subseteq DTQREL \circ LHOM.$$

Proof. (1) The first inclusion is shown in [*BT*, Lemma 3.6] and the second inclusion easily follows from the proof of that lemma.

(2) The proof of this inclusion is similar to that of [*BT*, Theorems 3.5 and 3.15]. Roughly, for T in *LDT-FST*, one can construct T_1 in *DTQREL* and T_2 in *LHOM* such that the i-th rule $q(\sigma(x_1 \ldots x_k)) \rightarrow t$ of T is split into two rules $q(\sigma(x_1 \ldots x_k)) \rightarrow i(q_1(x_1) \ldots q_k(x_k))$ of T_1 and $*(i(x_1 \ldots x_k)) \rightarrow t[*(x_1), \ldots, *(x_k)]$ of T_2, where $*$ is the only state of T_2, and where, for each $j (1 \leq j \leq k)$, either $q_j(x_j)$ occurs in t or $q_j = q_0$ (and q_0 is a new state which is, for instance, the identity on all trees). □

We shall also use the following result (cf. [12]).

LEMMA 1.2. *Let F be the composition of a finite number of bottom-up fst, i.e. $F = B_1 \circ B_2 \circ \ldots \circ B_n$ for some $n \geq 1$ and $B_i \in B\text{-}FST$. Then*
(1) *RECOG is closed under F^{-1} (i.e. if $L \in RECOG$, then $F^{-1}(L) \in RECOG$), and*
(2) *dom$(F) \in RECOG$.*

Proof. (1). It obviously suffices to assume that $F \in B\text{-}FST$. Let L be a recognizable tree language and let R be a finite tree automaton with domain L. Then $F^{-1}(L) = \text{dom}(F \circ R)$. By [*BT*, Lemma 4.2(1)] $F \circ R \in B\text{-}FST$ and hence its domain is recognizable by [*BT*, Corollary 3.12]. Statement (2) is immediate from (1) by the fact that $\text{dom}(F) = F^{-1}(T_\Delta)$, where Δ is the output alphabet of F. □

Note that, since *FTA*, *REL* and *HOM* are included in *B-FST*, the decomposition result for top-down *fst* [*BT*, Theorem 3.9] implies that Lemma 1.2 also holds with "bottom-up" replaced by "top-down" ([12]).

We finally introduce some more terminology concerning surface sets and tree transformation languages.

Let $\mathscr{L}$ be a class of tree languages and $\mathscr{F}$ a class of tree transformations. Then $\mathscr{F}(\mathscr{L})$ denotes the class of tree languages $\{F(L) \mid F \in \mathscr{F} \text{ and } L \in \mathscr{L}\}$, which we shall call *($\mathscr{F}$. $\mathscr{L}$) surface sets.* If $\mathscr{L} = RECOG$ then the ($\mathscr{F}$, $\mathscr{L}$) surface sets are the $\mathscr{F}$ surface sets.

Let e be a fixed symbol of rank 0 (which may or may not be an element of a ranked alphabet). The *yield* of a tree t, denoted by yield(t), is the string defined recursively as follows:

(1) for σ of rank 0, $\text{yield}(\sigma) = \begin{cases} \sigma & \text{if } \sigma \neq e \\ \lambda & \text{if } \sigma = e \end{cases}$ where λ is the empty string;

(2) for σ of rank $k \geq 1$ and trees $t_1, \ldots, t_k$,

$$\text{yield}(\sigma(t_1 \ldots t_k)) = \text{yield}(t_1) \ldots \text{yield}(t_k).$$

Furthermore we define, for a tree language L, $\text{yield}(L) = \{\text{yield}(t) | t \in L\}$ and, for a family $\mathscr{L}$ of tree languages, $\text{yield}(\mathscr{L}) = \{\text{yield}(L) | L \in \mathscr{L}\}$. Thus yield($L$) is a string language and yield($\mathscr{L}$) is a family of string languages.

For a class $\mathscr{L}$ of tree languages and a class $\mathscr{F}$ of tree transformations, the class of languages yield($\mathscr{F}(\mathscr{L})$) will be called the class of ($\mathscr{F}$, $\mathscr{L}$) *tree transformation languages*. Thus a tree transformation language is the yield of a surface set.

In the next lemma we show that in many cases we can do without the special symbol e to denote λ. This lemma is in fact a particular case of Theorem 3.2.10 in [3].

LEMMA 1.3. *Let $\mathscr{L}$ be a family of tree languages and $\mathscr{F}$ a family of tree transformations such that $\mathscr{F}(\mathscr{L})$ is closed under linear deterministic bottom-up fst. Let L be an ($\mathscr{F}$, $\mathscr{L}$) tree transformation language i.e. $L \in \text{yield}(\mathscr{F}(\mathscr{L}))$). Then $L - \{\lambda\} = \text{yield}(F(M))$ for some $M \in \mathscr{L}$ and some $F \in \mathscr{F}$, such that the output alphabet of $\mathscr{F}$ does not contain e.*

Proof. Let $L = \text{yield}(G(M_1))$ for some $M_1 \in \mathscr{L}$ and $G \in \mathscr{F}$, and let $G(M_1)$ be over the ranked alphabet Σ with $e \in \Sigma$ (otherwise there is nothing to prove). We now construct a linear deterministic bottom-up *fst* B which, for any tree t in T_Σ, deletes all subtrees t_1 of t with $\text{yield}(t_1) = \lambda$ (and does not accept t if $\text{yield}(t) = \lambda$). In fact, $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ where $Q = \{q_e, q_f\}$, $Q_d = \{q_f\}$, Δ is the ranked alphabet such that, for each k, $\Delta_k = \Sigma - \{e\}$, and R contains the following rules. First of all it contains rules $\sigma \to q_f(\sigma)$ for each $\sigma \in \Sigma_0 - \{e\}$ and one rule $e \to q_e(\sigma_0)$ where σ_0 is an arbitrary element of $\Sigma_0 - \{e\}$ (note that for $\Sigma_0 = \{e\}$ the proof is trivial). Furthermore, for each $k \geq 1$, $\sigma \in \Sigma_k$ and $q_1, \ldots, q_k \in Q$, R contains the rule $\sigma(q_1(x_1) \ldots q_k(x_k)) \to q(\sigma(x_{i_1} \ldots x_{i_n}))$ where $q_{i_1} = q_{i_2} = \ldots = q_{i_n} = q_f$ $(1 \leq i_1 < i_2 < \ldots < i_n \leq k)$ and all other q_j are equal to q_e, and $q = q_e$ if and only if $n = 0$ (i.e. $q_1 = q_2 = \ldots = q_k = q_e$; in this case the righthand side of the rule is $q_e(\sigma)$). It is easy to prove that, if $t_1 \overset{*}{\Rightarrow} q_e(t_2)$, then $\text{yield}(t_1) = \lambda$, and if $t_1 \overset{*}{\Rightarrow} q_f(t_2)$, then $\text{yield}(t_2) = \text{yield}(t_1) \neq \lambda$. Thus $L - \{\lambda\} = \text{yield}(B(G(M_1)))$ and, since $\mathscr{F}(\mathscr{L})$ is closed under B, $L - \{\lambda\} = \text{yield}(F(M))$ for some $F(M) \in \mathscr{F}(\mathscr{L})$, where F has output alphabet Δ. □

Note that, when a tree transducer, together with an input tree language, is viewed as a generating device of a tree transformation language, then Lemma 1.3 tells us that we can get rid of λ-rules.

2. Top-down tree transducers with regular look-ahead; decomposition and composition

In this section we add the facility of regular look-ahead to the top-down *fst*. Consequently the top-down *fst* will be able to inspect a subtree in order to decide whether to delete it or not (cf. property ($B2$)). Thus the difference between bottom-up *fst* and top-down *fst* with regular look-ahead can then be characterized by properties ($B1$) and (T).

In order to define the top-down *fst* with regular look-ahead we have to slightly generalize the notion of a semi-thue system with variables [*BT*, section 1]. We shall allow the range of the variables to be different for different rules. Formally we redefine a *semi-thue system with variables* to be a system $G = \langle A, X, R\rangle$, where A is an alphabet, $X = \{x_1, x_2, \ldots\}$ and R a finite set of rules of the form $\langle \phi \to \psi, D\rangle$ such that, for some $k \geq 0$, ϕ and ψ are in $(A \cup X_k)^*$ and D is a mapping from X_k into the powerset of A^*. For $1 \leq i \leq k$, $D(x_i)$ is called the *range* of x_i; ϕ is called the *lefthand side* and ψ the *righthand side* of the rule. Whenever D is understood (in particular when $k = 0$, D is always empty) or will be specified later, we shall write $\phi \to \psi$ rather than $\langle \phi \to \psi, D\rangle$. The relations $\underset{G}{\Rightarrow}$ and $\overset{*}{\underset{G}{\Rightarrow}}$ are defined as in [*BT*], the only difference being that the mapping D now depends upon the rule.

We now define a top-down *fst* with regular look-ahead to be a top-down *fst* in which the ranges of the variables in each rule are certain recognizable tree languages.

Definition 2.1. A *top-down finite state tree transformation with regular look-ahead* (abbreviated by t^r-*fst*) is a 5-tuple $T = \langle \Sigma, \Delta, Q, Q_d, R\rangle$, where Σ, Δ, Q and Q_d are as for a top-down *fst* and R is a finite set of rules of the form $\langle t_1 \to t_2, D\rangle$, where $t_1 \to t_2$ is an ordinary top-down *fst* rule and D is a mapping from X_k into the powerset of T_Σ (where X_k is the set of variables occurring in t_1), such that, for $1 \leq i \leq k$, $D(x_i) \in RECOG$. T is viewed as the semi-thue system with variables $\langle \Sigma \cup \Delta \cup Q \cup \{(,)\}, X, R\rangle$, and the tree transformation defined by T is as usual $\{\langle t, s\rangle \in T_\Sigma \times T_\Delta | q(t) \overset{*}{\Rightarrow} s$ for some q in $Q_d\}$. □

The class of all t^r-*fst* will be denoted by T^R-*FST*.

We note that it will always be assumed in a t^r-*fst* that the ranges of the variables are specified in some effective way, for instance as deterministic bottom-up finite tree automata. Throughout the paper all constructions will be effective in this sense.

Example 2.2. There is a t^r-*fst* that is not a *t*-*fst*. In fact, consider the bottom-up *fst* $B = \langle \Sigma, \Delta, Q, Q_d, R\rangle$ of [*BT*, Example 2.6] which is not a *t*-*fst*. Let U be the recognizable tree language T_Ω, where $\Omega_0 = \{b\}$, $\Omega_1 = \{a\}$ and $\Omega_2 = \{\sigma\}$. Consider now the t^r-*fst* $T = \langle \Sigma, \Delta, Q', Q'_d, R'\rangle$ where $Q' = Q'_d = \{*\}$ and R' consists of the rules

$$\langle *(\sigma(xy)) \to \sigma(*(x)), D_1\rangle \text{ with } D_1(x) = D_1(y) = U,$$
$$\langle *(a(x)) \to a(*(x)), D_2\rangle \text{ with } D_2(x) = T_\Sigma, \text{ and}$$
$$*(b) \to b.$$

Then, obviously, $T = B$. For instance, $*(\sigma(ba(b))) \Rightarrow \sigma(*(b)) \Rightarrow \sigma(b)$, since both b and $a(b)$ belong to U. But no rule is applicable to $*(\sigma(ba(a)))$. Note that $D_1(x)$ could as well be T_Σ since T can check later that the left subtree is of the required form. The essential use of the regular look-ahead is in the restriction of the right subtree to U.

An even more simple example was exhibited in [16]. Let a be of rank 0 and b of rank 2. Then the tree transformation $\{\langle b(aa), a\rangle\}$ is not a *t*-*fst*, but it is a t^r-*fst* and also a *b*-*fst*. □

We immediately obtain the following corollary.

COROLLARY 2.3. *T-$FST \subset T^R$-FST.*

Proof. Inclusion is trivial: each *t-fst* is changed into a *t^r-fst* by simply specifying all variables to range over the recognizable tree language T_Σ, where Σ is the input alphabet. Proper inclusion was shown in Example 2.2. □

We now obtain the following two facts.

COROLLARY 2.4.
(1) *The classes of tree transformations B-FST and T^R-FST are incomparable.*
(2) *T^R-FST is not closed under composition.*

Proof. For (1), it should be clear that the *b-fst* of [*BT*, Example 2.1] is not a *t^r-fst*, while Corollary 2.3 and [*BT*, Example 2.2] imply that there is a *t^r-fst* that is not a *b-fst*. For (2), note that [*BT*, Example 2.1] is a composition of two t^r-fst. □

We now define linearity and determinism.

Definition 2.5. Let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a *t^r-fst*. T is *linear* if all righthand sides of rules in R are linear. T is *deterministic* if the following holds:
(1) Q_d is a singleton;
(2) if $\langle s \to t_1, D_1 \rangle$ and $\langle s \to t_2, D_2 \rangle$ are different rules in R (with the same lefthand side), then $D_1(x_i) \cap D_2(x_i) = \varnothing$ for some $i \in \{1, 2, \ldots, k\}$, where k is the number of variables in s (for $k = 0$ this means that different rules should have different lefthand sides). □

Thus, in a deterministic *t^r-fst*, different rules may have the same lefthand side, but, in that case, the ranges of the variables are such that the two rules are never applicable in the same situation. Note that one can effectively determine whether a given *t^r-fst* is deterministic (*RECOG* is closed under intersection and has a solvable emptiness problem).

Linearity and determinism will be denoted as usual by L and D respectively. Note that, for a modifier $Z \in \{L, D, LD\}$, ZT-$FST \subset ZT$-FST (the *t^r-fst* of Example 2.2 is linear and deterministic).

Since for linear *b-fst* and linear *t^r-fst* all properties ($B1$), ($B2$) and (T) are now "eliminated", one would expect that LB-$FST = LT^R$-FST. Before proving this we show how to decompose the *t^r-fst*: the regular look-ahead can be computed in advance by a deterministic bottom-up finite state relabeling.

THEOREM 2.6. *T^R-$FST \subseteq DBQREL \circ T$-$FST$, and, for $Z \in \{L, D, LD\}$, ZT^R-$FST \subseteq DBQREL \circ ZT$-$FST$.*

Proof. Let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a *t^r-fst*. Consider all "recognizable properties" which T checks with its regular look-ahead. A finite state relabeling can be used to check, for a given input tree t, whether the subtrees of t have these properties or not, and to put this information at their father nodes. After this, an ordinary *t-fst* can be used to simulate T. Formally we proceed as follows.

Let $L_1, \ldots, L_n$ be all the recognizable tree languages occurring as ranges of variables in the rules of T. Let U denote the set $\{0, 1\}^n$, i.e. the set of all sequences of 0's and 1's of length n. For $u \in U$, the jth element $(1 \leq j \leq n)$ of u will be denoted by $u(j)$. Intuitively, an element u of U will be used to indicate membership of a tree in $L_1, \ldots, L_n$ ($u(j) = 1$ iff the tree belongs to L_j). Let Ω be the ranked alphabet such that $\Omega_0 = \Sigma_0$ and, for $k \geq 1$, $\Omega_k = \Sigma_k \times U^k$. Thus an element of Ω_k is of the form $\langle \sigma, \langle u_1, \ldots, u_k \rangle \rangle$ with $\sigma \in \Sigma_k$ and $u_1, \ldots, u_k \in U$. Intuitively, if a node is labeled by $\langle \sigma, \langle u_1, \ldots, u_k \rangle \rangle$, it means that u_i contains all the information about the ith subtree of the node. The mapping B: $T_\Sigma \to T_\Omega$ is now defined recursively as follows:

(1) for $\sigma \in \Sigma_0$, $B(\sigma) = \sigma$;

(2) for $k \geq 1, \sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma, B(\sigma(t_1 \ldots t_k)) = \tau(B(t_1) \ldots B(t_k))$, where $\tau = \langle \sigma, \langle u_1, \ldots, u_k \rangle \rangle$ and, for $1 \leq i \leq k$ and $1 \leq j \leq n$, $u_i(j) = 1$ iff $t_i \in L_j$.

It is left to the reader to show that B can be realized by a (total) deterministic bottom-up finite state relabeling (given the deterministic bottom-up finite tree automata recognizing $L_1, \ldots, L_n$). Next we define the top-down *fst* $T' = \langle \Omega, \Delta, Q, Q_d, R' \rangle$ such that

(1) if $q(\sigma) \to t$ is in R, then it is in R';

(2) if $\langle q(\sigma(x_1 \ldots x_k)) \to t, D \rangle$ is in R, then each rule of the form $q(\langle \sigma, \bar{u} \rangle (x_1 \ldots x_k)) \to t$ is in R', where $\bar{u} = \langle u_1, \ldots, u_k \rangle \in U^k$ and, for $1 \leq i \leq k$ and $1 \leq j \leq n$, if $D(x_i) = L_j$ then $u_i(j) = 1$.

This completes the construction. It should be clear that $T = B \circ T'$, and that, if T is linear, then so is T'. It should also be obvious that, in the above construction, U may be replaced by the smaller set $\{u \in U |$ for all j_1 and j_2, if $L_{j_1} \cap L_{j_2} = \emptyset$, then $u(j_1)$ and $u(j_2)$ are not both $1\}$ (other elements of U do not occur in trees $B(t)$). After this replacement (which influences T') one can easily see that if T is deterministic, then so is T'. □

An immediate consequence of this theorem and previous decomposition results (in [*BT*]) is that each element of T^R-*FST* is decomposable into elements of *REL*, *FTA* and *HOM*.

COROLLARY 2.7. *The domain of a t^r-fst is recognizable.*

Proof. Lemma 1.2(2). □

We now show that the classes of linear *b-fst* and linear t^r-*fst* coincide (cf. [*BT*, Theorem 2.8]).

THEOREM 2.8. *LT^R-FST = LB-FST.*

Proof. First

$$\begin{aligned} LT^R\text{-}FST &\subseteq DBQREL \circ LT\text{-}FST && \text{by Theorem 2.6} \\ &\subseteq DBQREL \circ LB\text{-}FST && \text{by [BT, Theorem 2.8]} \\ &\subseteq LB\text{-}FST && \text{by [BT, Theorem 4.5(2)]} \end{aligned}$$

(note that finite state relabelings are linear).

Secondly, we show that LB-*FST* $\subseteq LT^R$-*FST*. The construction is the same as that in the proof of [*BT*, Theorem 2.9], but now we can use look-ahead to handle deletion. Let $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be an arbitrary linear *b-fst*. Let, for each q in Q, $B(q)$ denote the *b-fst* $\langle \Sigma, \Delta, Q, \{q\}, R \rangle$. Then we construct the linear t^r-*fst* $T = \langle \Sigma, \Delta, Q, Q_d, R_T \rangle$, where R_T is defined by the following two requirements.

(1) If $\sigma \to q(t)$ is in R, then $q(\sigma) \to t$ is in R_T.

(2) If $\sigma(q_1(x_1) \ldots q_k(x_k)) \to q(t)$ is in R, then

the rule $\langle q(\sigma(x_1 \ldots x_k)) \to t[q_1(x_1), \ldots, q_k(x_k)], D \rangle$, is in R_T, where, for $1 \leq i \leq k$, $D(x_i) = \text{dom}(B(q_i))$. Note that $\text{dom}(B(q_i))$ is recognizable by [*BT*, Corollary 3.12]. Note also that it would suffice to have $D(x_i) = \text{dom}(B(q_i))$ for those x_i that do not occur in t, and $D(x_i) = T_\Sigma$ for the other x_i.

A formal proof that $T = B$ is left to the reader. Intuitively T simulates B in the top-down direction by translating each node in the same piece of tree as B. Whenever B deletes a subtree t after arriving at its top in state q, T checks whether $t \in \text{dom}(B(q))$ before deleting t ($t \in \text{dom}(B(q))$ means that there exists $s \in T_\Delta$ such that $t \overset{*}{\underset{B}{\Rightarrow}} q(s)$). □

In the rest of this section we discuss composition of t^r-*fst*. We shall show (cf. property ($B1$)) that, if either T_1 is deterministic or T_2 is linear, then $T_1 \circ T_2$ is in T^R-*FST*. Moreover, DT^R-*FST* and LT^R-*FST* are closed under composition. To prove these results we first consider some simple cases in the following two lemmas (concerning homomorphisms and finite state relabelings respectively).

LEMMA 2.9.

$$(1) \qquad T^R\text{-}FST \circ LHOM \subseteq T^R\text{-}FST,$$

$$(2) \qquad DT^R\text{-}FST \circ HOM \subseteq DT^R\text{-}FST.$$

Proof. Let $T = \langle \Sigma, \Delta, Q, Q_d, R\rangle$ be a t^r-*fst* and let H be a homomorphism from T_Δ into T_Ω. For both cases, (1) and (2), the construction of a t^r-*fst* T' defining $T \circ H$ is similar to that in [BT, Lemma 4.1]; look-ahead is used to handle deletion by H. Let H be extended to $T_\Delta[Q(X)]$ by defining, informally, $H(q(x)) = q(x)$ for all $q(x) \in Q(X)$. Let, for $p \in Q$, $T(p)$ denote the t^r-*fst* $\langle \Sigma, \Delta, Q, \{p\}, R\rangle$. Note that, by Corollary 2.7, $\text{dom}(T(p)) \in RECOG$. We now construct the t^r-*fst* $T' = \langle \Sigma, \Omega, Q, Q_d, R'\rangle$ such that

(1) if $q(\sigma) \to t$ is in R, then $q(\sigma) \to H(t)$ is in R';

(2) if $\langle q(\sigma(x_1 \ldots x_k)) \to t, D\rangle$ is in R,

then $\langle q(\sigma(x_1 \ldots x_k)) \to H(t), D'\rangle$ is in R', where, for $1 \le i \le k$, $D'(x_i)$ is the intersection of $D(x_i)$ and all tree languages $\text{dom}(T(p))$ such that $p(x_i)$ occurs in t but not in $H(t)$.

It is left to the reader to prove that $T \circ H \subseteq T'$ and that, if H is linear or T is deterministic, then $T' \subseteq T \circ H$ also. Note that, if T is deterministic, then so is T' (the $D'(x_i)$ are included in the $D(x_i)$). This proves the lemma. □

LEMMA 2.10.

$$(1) \qquad T^R\text{-}FST \circ QREL \subseteq T^R\text{-}FST,$$

$$(2) \qquad DT^R\text{-}FST \circ DTQREL \subseteq DT^R\text{-}FST,$$

$$(3) \qquad DT^R\text{-}FST \circ DBQREL \subseteq DT^R\text{-}FST.$$

Proof. We first prove (1) and (2). The proof is similar to that of [BT, Lemma 4.2]. Let $T = \langle \Sigma, \Delta, Q, Q_d, R_T\rangle$ be a t^r-*fst* and $L = \langle \Delta, \Omega, P, P_d, R_L\rangle$ a top-down finite state relabeling. We extend the input alphabet of L to $\Delta \cup X$ by adding X to Δ_0. We now define a t^r-*fst* K such that $K = T\ L$. Let $K = \langle \Sigma, \Omega, Q \times P, Q_d \times P_d, R_K\rangle$, where R_K is obtained by the following two requirements.

(i) If the rule $q(\sigma) \to t_1$ is in R_T and $p(t_1) \overset{*}{\underset{L}{\Rightarrow}} t_2$, then the rule $\langle q,p\rangle (\sigma) \to t_2$ is in R_K.

(ii) Let $\langle q(\sigma(x_1 \dots x_k)) \to t, D\rangle$ be in R_T. Obviously t can be written as $t = s_1[q_1(x_{i_1}), \dots, q_m(x_{i_m})]$, where $s_1 \in T_\Delta[X_m]$ is linear and nondeleting with respect to X_m. If $p(s_1) \overset{*}{\underset{L}{\Rightarrow}} s_2[p_1(x_1), \dots, p_m(x_m)]$, then the rule

$$\langle q, p\rangle(\sigma(x_1 \dots x_k)) \to s_2[\langle q_1, p_1\rangle(x_{i_1}), \dots, \langle q_m, p_m\rangle(x_{i_m})]$$

is in R_K with the same D.

Clearly, if T and L are deterministic, then so is K.

We now prove (3), which is the essential composition result. Let T be in DT^R-FST and B in $DBQREL$. We shall construct a transducer T' in DT^R-FST such that $T' = T \circ B$.

Intuitively, when T' arrives at a node of the input tree, it first computes the piece of output t that T would produce at this node, and then runs B on t. However, to be able to run B on t, T' should know the states in which B arrives at this piece of output. But, these states can be computed by regular look-ahead. The formal construction is as follows. Let $T = \langle \Sigma, \Delta, Q, Q_d, R\rangle$ with $Q_d = \{q_d\}$ and let $B = \langle \Delta, \Omega, Q_B, Q_{Bd}, R_B\rangle$. Let as usual, for $q \in Q$, $T(q)$ denote $\langle \Sigma, \Delta, Q, \{q\}, R\rangle$ and, for $q \in Q_B$, $B(q)$ denote $\langle \Delta, \Omega, Q_B, \{q\}, R_B\rangle$. We now construct the t^r-fst $T' = \langle \Sigma, \Omega, Q, Q_d, R'\rangle$, where R' is determined as follows.

(i) Let $q(\sigma) \to t$ be in R, where $q \in Q$, $\sigma \in \Sigma_0$ and $t \in T_\Delta$. Suppose that $t \overset{*}{\underset{B}{\Rightarrow}} p(t')$ for some $t' \in T_\Omega$ and some $p \in Q_B$ such that, if $q = q_d$ then $p \in Q_{Bd}$. Then the rule $q(\sigma) \to t'$ is in R'.

(ii) Let $\langle q_0(\sigma(x_1 \dots x_k)) \to t, D\rangle$ be in R, where $k \geq 1$, $\sigma \in \Sigma_k$, $t \in T_\Delta[Q(X_k)]$, $q_0 \in Q$ and, for $1 \leq i \leq k$, $D(x_i) \subseteq T_\Sigma$. Clearly t can be written as $t = s[q_1(x_{i_1}), \dots, q_m(x_{i_m})]$ for certain $m \geq 0$, $s \in T_\Delta[X_m]$, $q_1, \dots, q_m \in Q$ and $x_{i_1}, \dots, x_{i_m} \in X_k$, such that $x_1, \dots, x_m$ all occur in s. Let $p_1, \dots, p_m$ be a sequence of m states from Q_B and suppose that $s[p_1(x_1), \dots, p_m(x_m)] \overset{*}{\underset{B}{\Rightarrow}} p_0(s')$ for some $s' \in T_\Omega[X_m]$ and some $p_0 \in Q_B$ such that, if $q_0 = q_d$ then $p_0 \in Q_{Bd}$ (B is of course extended to trees with variables in the usual way). Then the rule $q_0(\sigma(x_1 \dots x_k)) \to s'[q_1(x_{i_1}), \dots, q_m(x_{i_m})]$ is in R', where the ranges of $x_1, \dots, x_k$ are specified by D' as follows. For $1 \leq u \leq k$, $D'(x_u)$ is the intersection of $D(x_u)$ and all tree languages $\mathrm{dom}(T(q_j) \circ B(p_j))$ such that $x_{i_j} = x_u$. Note that these tree languages are recognizable by Lemma 1.2 and the fact that each t^r-fst can be decomposed into b-fst (Theorem 2.6).

This ends the construction of T'. It is left to the reader to check that T' is deterministic (using the determinism of T and B) and to prove that $T' = T \circ B$. □

We can now prove the composition results for t^r-fst.

THEOREM 2.11.

(1) $$T^R\text{-}FST \circ LT^R\text{-}FST \subseteq T^R\text{-}FST, \text{ and}$$
$$LT^R\text{-}FST \circ LT^R\text{-}FST \subseteq LT^R\text{-}FST.$$

(2) $$DT^R\text{-}FST \circ T^R\text{-}FST \subseteq T^R\text{-}FST, \text{ and}$$
$$DT^R\text{-}FST \circ DT^R\text{-}FST \subseteq DT^R\text{-}FST.$$

Proof.

(1) The second inclusion is immediate by Theorem 2.8 and [*BT*, Theorem 4.5(2)]. The first inclusion can be shown as follows.

$$\begin{aligned}
&T^R\text{-}FST \circ LT^R\text{-}FST \\
&\quad = T^R\text{-}FST \circ LB\text{-}FST && \text{by Theorem 2.8} \\
&\quad \subseteq T^R\text{-}FST \circ QREL \circ LHOM && \text{by [}BT\text{, Theorem 3.15(2)]} \\
&\quad \subseteq T^R\text{-}FST \circ LHOM && \text{by Lemma 2.10(1)} \\
&\quad \subseteq T^R\text{-}FST && \text{by Lemma 2.9(1).}
\end{aligned}$$

(2) For both inclusions we have that

$$\begin{aligned}
&DT^R\text{-}FST \circ (D)T^R\text{-}FST \\
&\quad \subseteq DT^R\text{-}FST \circ DBQREL \circ (D)T\text{-}FST && \text{by Theorem 2.6} \\
&\quad \subseteq DT^R\text{-}FST \circ (D)T\text{-}FST && \text{by Lemma 2.10(3)} \\
&\quad \subseteq DT^R\text{-}FST \circ HOM \circ L(D)T\text{-}FST && \text{by Lemma 1.1(1)} \\
&\quad \subseteq DT^R\text{-}FST \circ L(D)T\text{-}FST && \text{by Lemma 2.9(2).}
\end{aligned}$$

Now $DT^R\text{-}FST \circ LT\text{-}FST \subseteq T^R\text{-}FST$ by (1) of this theorem, and

$$\begin{aligned}
&DT^R\text{-}FST \circ LDT\text{-}FST \\
&\quad \subseteq DT^R\text{-}FST \circ DTQREL \circ LHOM && \text{by Lemma 1.1(2)} \\
&\quad \subseteq DT^R\text{-}FST \circ LHOM && \text{by Lemma 2.10(2)} \\
&\quad \subseteq DT^R\text{-}FST && \text{by Lemma 2.9(2).}
\end{aligned}$$

This proves the theorem. □

It is left to the reader to show that $LDT^R\text{-}FST$ is closed under composition.

Note that it follows from Theorem 2.11 that the inclusion signs in Theorem 2.6 may be replaced by equality signs. Thus $T^R\text{-}FST = DBQREL \circ T\text{-}FST$. We finally mention a result similar to [*BT*, Theorem 3.7] (see also [*BT*, Theorem 5.15]).

THEOREM 2.12. $T^R\text{-}FST = HOM \circ LT^R\text{-}FST$.

Proof. The inclusion $HOM \circ LT^R\text{-}FST \subseteq T^R\text{-}FST$ is immediate from Theorem 2.11. The inclusion $T^R\text{-}FST \subseteq HOM \circ LT^R\text{-}FST$ can be shown in much the same way as in the proof of $T\text{-}FST \subseteq HOM \circ LT\text{-}FST$ [*BT*, Lemma 3.6]. The only additional problem is the regular look-ahead: the image of a recognizable tree language under a homomorphism need not be recognizable. The solution is to consider a homomorphism H from $T_{\Sigma'}$ to T_Σ (see the proof of [*BT*, Lemma 3.6] for notation) such that, for all t in T_Σ, $H(T_1(t)) = t$. The easy definition of H is left to the reader. Now, if in a rule of the t^r-*fst* T, the recognizable tree language U occurs as look-ahead, then we can use $H^{-1}(U)$ as look-ahead in the corresponding rule of T_2. Note that $H^{-1}(U) \in RECOG$ (cf. Lemma 1.2(1)). The details of the proof are left to the reader. □

COROLLARY 2.13. $T^R\text{-}FST = T\text{-}FST \circ LHOM$.

Proof. By Theorems 2.12 and 2.8, $T^R\text{-}FST = HOM \circ LB\text{-}FST$. From the proof of (7) in [*BT*, section 6] it follows that $HOM \circ LB\text{-}FST = T\text{-}FST \circ LHOM$. □

3. Comparison of deterministic fst

The classes of tree transformations $DB\text{-}FST$ and $DT\text{-}FST$ are incomparable. In fact there are several reasons for the incomparability of these classes. We now consider some typical *db-fst* and *dt-fst* capabilities respectively. We start by considering advantages of *DB-FST* over *DT-FST*.

Firstly we note that property ($B1$) is eliminated, but property ($B2$) is not. Thus *DB-FST* contains elements not even in *T-FST* (obviously, the *b-fst* B in [*BT*, Example 2.6] is in *DB-FST*).

Secondly, a *db-fst* can recognize the "lowest" occurrence of some symbol in a tree (since it is the first occurrence), but this cannot be done by a *dt-fst* (since it is the last occurrence for him).

Thirdly, it is well known (see for instance [15]) that there are recognizable tree languages which cannot be recognized by a deterministic top-down finite tree automaton. The next theorem shows that such languages cannot be the domain of any deterministic *t-fst* (cf. [11]).

THEOREM 3.1. *A tree language is the domain of a deterministic t-fst if and only if it is the domain of a deterministic top-down fta.*

Proof. The if-direction is trivial. To prove the only-if direction, let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a *dt-fst*. We may assume that for all $k \geq 1$, $\sigma \in \Sigma_k$ and $q \in Q$ there is a rule with lefthand side $q(\sigma(x_1 \ldots x_k))$ in R. We construct the deterministic top-down *fta* $F = \langle \Sigma, \Sigma, Q', Q'_d, R' \rangle$ such that Q' is the powerset of Q, $Q'_d = \{Q_d\}$ and R' is defined as follows.

(1) For $k \geq 1$, $A \subseteq Q$ and $\sigma \in \Sigma_k$, the rule $A(\sigma(x_1 \ldots x_k)) \rightarrow \sigma(A_1(x_1) \ldots A_n(x_n))$ is in R', where $A_i = \{p \in Q |$ there is a rule $q(\sigma(x_1 \ldots x_k)) \rightarrow t$ in R such that $q \in A$ and $p(x_i)$ occurs in $t\}$.

(2) For $A \subseteq Q$ and $\sigma \in \Sigma_0$, $A(\sigma) \rightarrow \sigma$ is in R' if and only if for all $q \in A$ there is a rule with lefthand side $q(\sigma)$ in R (note that in particular $\varnothing(\sigma) \rightarrow \sigma$ is in R').

It is left to the reader to show that $\mathrm{dom}(F) = \mathrm{dom}(T)$. Intuitively, the state of F at some node contains all states in which T arrives at copies of this node (made by T when processing higher nodes). At the leaves, F checks whether all these states are final states of T. □

Next we consider advantages of *DT-FST* over *DB-FST*. First we note that property (T) is not eliminated: a *dt-fst* has the ability to copy an input subtree and to continue translation of these copies in different states. Thus the *dt-fst* which translates every tree $\sigma(b(b(\ldots b(a) \ldots)))$ into $\tau(b((\ldots b(a_1) \ldots))b(b(\ldots b(a_2) \ldots)))$ is not in *B-FST*.

Secondly, a *dt-fst* can recognize the "highest" occurrence of some symbol in a tree, but this cannot be done by a *db-fst*.

Thirdly, a *dt-fst* can distinguish between left and right, but a *db-fst* is not able to see this difference, because it starts at the bottom.

This concludes our comparison of *DB-FST* and *DT-FST*. The reader might have noticed that the mentioned advantages of *DB-FST* over *DT-FST* can all be handled by the use of regular look-ahead. Also, those of *DT-FST* over *DB-FST* can be eliminated by restricting the number of states of the *dt-fst* to one. We now show that this holds in general. Let ODT^R-*FST* denote the class of dt^r-*fst* $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ such that $Q = Q_d$, i.e. the class of one-state deterministic t^r-*fst*.

THEOREM 3.2. ODT^R-*FST* = *DB-FST* $\subset$ DT^R-*FST*.

Proof. Inclusion of *DB-FST* in DT^R-*FST* is proved as follows.

$$\begin{array}{ll}
\textit{DB-FST} & \\
\quad \subseteq \textit{DBQREL} \circ \textit{HOM} & \text{by [}\textit{BT}\text{, Theorem 3.15(3)]} \\
\quad \subseteq \textit{DT}^R\textit{-FST} \circ \textit{DBQREL} \circ \textit{HOM} & \text{(since the identity is in } \textit{DT}^R\textit{-FST}\text{)} \\
\quad \subseteq \textit{DT}^R\textit{-FST} \circ \textit{HOM} & \text{by Lemma 2.10(3)} \\
\quad \subseteq \textit{DT}^R\textit{-FST} & \text{by Lemma 2.9(2).}
\end{array}$$

Since the identity can be realized by a one-state *dt*^{r}*-fst* and since the constructions in Lemmas 2.9 and 2.10(3) preserve the number of states, *DB-FST* is included in *ODT*^{R}*-FST*. The properness of the inclusion of *DB-FST* in *DT*^{R}*-FST* follows from the discussion preceding this theorem. Inclusion of *ODT*^{R}*-FST* in *DB-FST* can be proved as follows. By Theorem 2.6, *DT*^{R}*-FST* $\subseteq$ *DBQREL* $\circ$ *DT-FST*. Moreover, from the construction in the proof of that theorem it follows that every one-state *dt*^{r}*-fst* is the composition of an element of *DBQREL* and a one-state *dt-fst*. It is left to the reader to show that each one-state *dt-fst* is in *DB-FST*. The required inclusion now follows from the closure of *DB-FST* under composition ([*BT*, Theorem 4.6(2)]). □

Thus the addition of regular look-ahead to *T-FST* has made the deterministic bottom-up *fst* into a proper subclass of the deterministic top-down *fst* (with regular look-ahead).

4. Surface sets and tree transformation languages

In this section we show how the results of the previous sections can be used to prove properties of surface sets and tree transformation languages, in particular closure properties.

Notation 4.1. Throughout this section, $\mathscr{L}$ denotes a fixed family of tree languages closed under deterministic bottom-up finite state relabelings (i.e. elements of *DBQREL*). □

Note that *DBQREL* is included in both *LB-FST* and *DB-FST*. Note also that for instance *RECOG* is closed under *DBQREL*.

We first show that regular look-ahead has no influence on surface sets: the classes of (*T-FST*, $\mathscr{L}$) and (*T*^{R}*-FST*, $\mathscr{L}$) surface sets are equal.

THEOREM 4.2.

$$(1) \qquad T^R\textit{-FST}(\mathscr{L}) = T\textit{-FST}(\mathscr{L}),$$
$$(2) \qquad DT^R\textit{-FST}(\mathscr{L}) = DT\textit{-FST}(\mathscr{L}),$$
$$(3) \qquad LT^R\textit{-FST}(\mathscr{L}) = LT\textit{-FST}(\mathscr{L}) = LB\textit{-FST}(\mathscr{L}).$$

Proof. Follows immediately from the decomposition result of Theorem 2.6 (and, for (3), Theorem 2.8). □

Obviously a similar result for tree transformation languages is obtained by applying yield to the above equations.

From this theorem and the composition results in Theorem 2.11 we obtain a number of closure properties of surface sets, some of which are expressed in the next theorem.

THEOREM 4.3.
(1) *T-$FST(\mathscr{L})$ is closed under linear fst.*
(2) *DT-$FST(\mathscr{L})$ is closed under deterministic bottom-up and top-down fst.*

Proof. Immediate from Theorem 4.2, Theorem 2.11 and Theorem 3.3. □

Theorem 4.3(1) was proved by Baker [3, Theorem 1.2.5] by generalizing Rounds' proof [13] for the special case $\mathscr{L} = RECOG$. Closure of DT-$FST(RECOG)$ under *dt-fst* was proved by Rounds [13].

These theorems can easily be extended to surface sets which are obtained by repeated application of top-down *fst*. In fact, the next theorem shows that the regular look-ahead can be "taken out of" any sequence of t^r-*fst*. Let, for any class $\mathscr{F}$ of tree transformations, $\mathscr{F}^k$ be defined by $\mathscr{F}^1 = \mathscr{F}$ and $\mathscr{F}^{k+1} = \mathscr{F}^k \circ \mathscr{F}$.

THEOREM 4.4. *For each $k \geq 1$,*
(1) $(T^R\text{-}FST)^k = DBQREL \circ (T\text{-}FST)^k$,
(2) $(T^R\text{-}FST)^k \circ DT^R\text{-}FST = DBQREL \circ (T\text{-}FST)^k \circ DT\text{-}FST$.

Proof. (1) We first show that $T^R\text{-}FST \circ T^R\text{-}FST = T^R\text{-}FST \circ T\text{-}FST$. One inclusion is trivial. The other inclusion is proved as follows:

$$\begin{array}{lll} T^R\text{-}FST \circ T^R\text{-}FST & & \\ \quad \subseteq T^R\text{-}FST \circ DBQREL \circ T\text{-}FST & & \text{by Theorem 2.6} \\ \quad \subseteq T^R\text{-}FST \circ T\text{-}FST & & \text{by Lemma 2.10(1).} \end{array}$$

From this, and the fact that $T^R\text{-}FST = DBQREL \circ T\text{-}FST$ (see section 2), (1) easily follows. The proof of (2) is similar. □

From this theorem it follows for instance that $(T\text{-}FST)^k(\mathscr{L}) = (T^R\text{-}FST)^k(\mathscr{L})$, and hence $(T\text{-}FST)^k(\mathscr{L})$ is closed under linear *fst* ([3, Corollary 1.2.7]). Similarly, $DT\text{-}FST((T\text{-}FST)^k(\mathscr{L}))$ is closed under deterministic bottom-up and top-down *fst*.

Let us now turn to tree transformation languages. Recall that we have introduced a symbol e such that yield$(e) = \lambda$. We note first that it follows from Theorem 4.3 that Lemma 1.3 holds for both yield$(T\text{-}FST(\mathscr{L}))$ and yield$(DT\text{-}FST(\mathscr{L}))$. We express this informally in the following corollary.

COROLLARY 4.5 *Both $(T\text{-}FST, \mathscr{L})$ and $(DT\text{-}FST, \mathscr{L})$ tree transformation languages can be "generated without λ-rules" (modulo λ).* □

It should be clear that from Theorem 4.3 other closure properties for these tree transformation languages can be inferred. Since the closure properties of yield$(T\text{-}FST(\mathscr{L}))$ have been discussed thoroughly by Baker [3], we restrict ourselves to the following closure property of deterministic tree transformation languages.

THEOREM 4.6. *The class of tree transformation languages yield$(DT\text{-}FST(\mathscr{L}))$ is closed under deterministic gsm mappings.*

Proof. Let Σ and Δ be ranked alphabets with $e \in \Sigma_0$ and $e \in \Delta_0$. Let $S = \langle K, \Sigma_0 - \{e\}, \Delta_0 - \{e\}, \delta, q_0, F\rangle$ be a deterministic *gsm* (for notation, see [10, sections 9.3 and 12.3]). We shall show that there exists a deterministic top-down *fst* T with regular look-ahead such that, for every $t \in T_\Sigma$, if yield(t) is not accepted by S, then t

is not accepted by T, and if yield(t) is accepted by S, then so is t by T and yield($T(t)$) $= S$(yield(t)). Consequently, for any tree language $L \subseteq T_\Sigma$, yield($T(L)$) $= S$(yield(L)). The theorem then easily follows from the closure of DT^R-FST under composition (Theorem 2.11(2)).

T is constructed as follows (the construction being a variation on a known theme). Let, for $q_1, q_2 \in K$, $R(q_1, q_2)$ denote the recognizable tree language consisting of all trees $t \in T_\Sigma$ such that $\delta(q_1, \text{yield}(t)) = \langle q_2, w\rangle$ for some output string $w \in (\Delta_0 - \{e\})^*$ (thus, when started in state q_1, S arrives in state q_2 after processing yield(t)). Recognizability of $R(q_1, q_2)$ follows from a straight forward extension, to handle e, of [13, section 3, Lemma 2]. Let now $T = \langle \Sigma, \Delta, Q, Q_d, R\rangle$, where $Q = (K \times K) \cup \{q_s\}$ (with q_s new), $Q_d = \{q_s\}$ and the rules of R are defined as follows.

(1) For $k \geq 1, \sigma \in \Sigma_k$ and $q_1, q_2, \ldots, q_{k+1} \in K$, the rule $\langle q_1, q_{k+1}\rangle (\sigma(x_1 \ldots x_k)) \to \sigma(\langle q_1, q_2\rangle (x_1) \langle q_2, q_3\rangle (x_2) \ldots \langle q_k, q_{k+1}\rangle (x_k))$ is in R, where the range of variable x_i is $D(x_i) = R(q_i, q_{i+1})$.

(2) For $\sigma \in \Sigma_0 - \{e\}$ and $\langle q_1, q_2\rangle \in K \times K$, if $\delta(q_1, \sigma) = \langle q_2, w\rangle$ for some $w \in (\Delta_0 - \{e\})^*$, then the rule $\langle q_1, q_2\rangle (\sigma) \to t$ is in R, where t is some tree in T_Δ such that yield(t) $= w$ (note that, if $w = \lambda$, one can take $t = e$).

(3) For $q \in K$, the rule $\langle q, q\rangle (e) \to e$ is in R.

(4) For $k \geq 1, \sigma \in \Sigma_k$ and $q_1, \ldots, q_{k+1} \in K$, if $q_1 = q_0$ and $q_{k+1} \in F$, then the rule $q_s(\sigma(x_1 \ldots x_k)) \to \sigma(\langle q_1, q_2\rangle (x_1) \ldots \langle q_k, q_{k+1}\rangle (x_k))$ is in R, where the range of x_i is $R(q_i, q_{i+1})$.

(5) For $\sigma \in \Sigma_0 - \{e\}$, if $\delta(q_0, \sigma) = \langle q_f, w\rangle$ for some $q_f \in F$ and $w \in (\Delta_0 - \{e\})^*$, then the rule $q_s(\sigma) \to t$ is in R, where t is a tree such that yield(t) $= w$.

(6) If $q_0 \in F$, then $q_s(e) \to e$ is in R.

This ends the construction of T. It should be clear that T is deterministic and that T satisfies the requirements. □

Note that it follows from this theorem that yield(DT-$FST(\mathscr{L})$) is closed under string homomorphisms and intersection with a regular language.

We finally mention that these results can directly be applied to certain classes of Lindenmayer languages (see also [1]).

Let MON be the class of monadic recognizable tree languages (a tree language is monadic if all symbols appearing in its trees are of rank 0 or 1; in [8] the number of symbols of rank 0 is restricted to one, but this is not essential for what follows).

It was shown in [1, 5, 8] that $ETOL =$ yield(T-$FST(MON)$) and $EDTOL$ $=$ yield(DT-$FST(MON)$), where $ETOL$ and $EDTOL$ are classes of Lindenmayer languages defined in for instance [9]. Thus, since MON is obviously closed under $DBQREL$, Corollary 4.5 implies the well known fact that (modulo λ) $ETOL$ and $EDTOL$ languages can be generated without λ-rules. From Theorem 4.6 we directly obtain the following useful result (cf. [6]).

COROLLARY 4.7. *$EDTOL$ is closed under deterministic gsm mappings.* □

For any $\mathscr{L} \subseteq MON$ (with certain closure properties) yield(T-$FST(\mathscr{L})$) and yield(DT-$FST(\mathscr{L})$) are equal to the $\mathscr{L}$-controlled $ETOL$ languages and the $\mathscr{L}$-controlled $EDTOL$ languages respectively (see [2]; for $L \in \mathscr{L}$, only those sequences of tables which are in L may be used in the generation of the $ETOL$

language). It follows that the above results are also applicable to controlled *ETOL* and *EDTOL* languages.

REFERENCES

[1] A. Arnold and M. Dauchet, Transductions de forets regulieres monadiques; forets coregulieres, *RAIRO* **10** (1976), 5–28.

[2] P. R. J. Asveld, Controlled iteration grammars and full hyper-AFL's, *Memoradum 114,* Technical University Twente, Holland, 1976.

[3] B. S. Baker, Tree transductions and families of tree languages, Ph.D. Thesis, Harvard University, *Report TR-9-73,* 1973 (also: *5th Theory of Computing,* 200–206).

[4] K. Čulik II and R. Cohen, LR-regular grammars—an extension of LR(k) grammars, *JCSS* 7 (1973), 66–96.

[5] P. J. Downey, Tree transducers and *ETOL* tree systems (abstract), *Conference on Formal Languages, Automata and Development,* Noordwijkerhout, Holland, 1975.

[6] A. Ehrenfeucht and G. Rozenberg, On inverse homomorphic images of deterministic *ETOL* languages, *LOCOS 13,* Utrecht University, Holland, 1974.

[7] J. Engelfriet, Bottom-up and top-down tree transformations—a comparison, *Math. Syst. Theory* **9** (1975), 198–231. This paper is also referred to as BT.

[8] J. Engelfriet, Surface tree languages and parallel derivation trees, *DAIMI Report PB-44,* Aarhus University, Denmark, 1975 (to appear in Theoretical Computer Science).

[9] G. T. Herman and G. Rozenberg, *Developmental systems and languages,* North-Holland Publ. Co., Amsterdam, 1975.

[10] J. E. Hopcroft and J. D. Ullman, *Formal languages and their relation to automata,* Addison-Wesley Publ. Co., Reading, Mass., 1969.

[11] L. S. Levy and A. K. Joshi, Some results in tree automata, *Math. Syst. Theory* **6** (1973), 334–342.

[12] W. F. Ogden and W. C. Rounds, Composition of *n* transducers, *4th Symp. on Theory of Computing,* 1972, pp. 198–206.

[13] W. C. Rounds, Mappings and grammars on trees, *Math. Syst. Theory* **4** ((1970), 257–287.

[14] J. W. Thatcher, Generalized^2 sequential machine maps, *JCSS* **4** (1970), 339–367.

[15] J. W. Thatcher, Tree automata: an informal survey, in: *Currents in the Theory of Computing* (ed. A. V. Aho), Prentice-Hall, 1973, pp. 143–172.

[16] M. Dauchet, *Transductions inversibles de forets,* These, Univ. de Lille, France, 1975.

Received September 18, 1975 and in revised form December 2, 1976.