SURFACE TREE LANGUAGES AND PARALLEL DERIVATION TREES

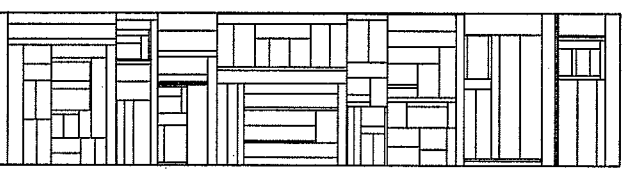
by

Joost Engelfriet

DAIMI PB-44

January 1975

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55



Abstract

The surface tree languages obtained by top-down finite state transformation of monadic trees are exactly the frontier-preserving homomorphic images of sets of derivation trees of ETOL systems. The corresponding class of tree transformation languages is therefore equal to the class of ETOL languages.

1. Introduction

The theory of formal languages, in particular regular languages, has been generalized to a large extent to a theory of tree languages (see for instance [7, 14, 15]). Regular tree languages (or, as they are called, recognizable tree languages) are of interest, among other reasons, because they are closely related to sets of derivation trees of context-free grammars. In fact, as shown in [12], each recognizable tree language is a projection (that is, node relabeling) of the set of derivation trees of some context-free grammar, and vice versa. It immediately follows that the class of frontiers of recognizable tree languages equals the class of context-free languages.

The notion of generalized sequential machine mapping has been generalized to that of finite state tree transformation [1, 3, 7, 13]. These tree transformations are studied because, when restricted to recognizable tree languages, they model the process of syntax-directed translation of context-free languages. The image of a recognizable tree language under a finite state tree transformation is called a surface tree language. Thus the class of frontiers of surface tree languages may be considered as the class of languages that can be obtained from the context-free languages by (generalized) syntax-directed translation.

In this paper we shall show that the notion of top-down finite state tree transformation is closely related to the notion of parallel rewriting in grammars, in particular the parallel rewriting which is the subject of the theory of Lindenmayer systems (see, for instance, [4, 9]).

The largest family of "context-free" Lindenmayer languages is that generated by so called ETOL systems. An ETOL system is like a context-free grammar except that, at each step in the derivation, each symbol in the intermediate string should be rewritten according to some production of the system (that is, all symbols in the string are rewritten in parallel). Moreover, the set of productions is divided into a number of (not necessarily disjoint) sets, called tables. At each step of the derivation only productions coming from one of these tables may be used. Finally one should note that for both terminals and nonterminals productions should be present in each table.

To show that tree transformation is related to parallel rewriting we shall consider the very special case that the trees given as input to the tree transducer are all monadic, i.e. they are "vertical strings". It will turn out that, in that case, the class of frontiers of surface languages is the class of ETOL languages. Moreover, as in the case of recognizable tree languages and context-free languages, there is a simple relationship between the surface tree languages and the sets of derivation trees of ETOL systems: each surface tree language is a homomorphic image of the set of derivation trees of an ETOL system and vice versa, where the tree homomorphism involved is restricted to a very simple type: a socalled frontier-preserving tree homomorphism.

In section 2 we recall some of the terminology used in connection with tree transformations and ETOL languages, and we introduce a few new concepts.

In section 3 we first derive our main result concerning surface tree languages and derivation tree languages of ETOL systems. Then we consider the deterministic case and the case that the "vertical input strings" to the tree transducer are over a one-letter alphabet. Finally we briefly consider the ETOL tree languages introduced in [2], and show that they are equal to the monadic surface tree languages.

Hopefully the results in this paper will throw some new light on both the field of tree transformations and that of parallel rewriting.

2. Terminology

In this section we recall some of the well known definitions concerning tree transformations and ETOL systems. The tree transformation model discussed is the usual top-down finite state tree transformation introduced in [7]. The definition of an ETOL derivation tree will be an obvious formalization of the informal notion occurring in [4].

Trees

2.1. Definition. An alphabet Σ is ranked if for each $k \geq 0$, a set $\Sigma_k \subseteq \Sigma$ is specified such that

(i) $\Sigma = \bigcup_{k \geq 0} \Sigma_k$

(ii) there is an integer N such that for $k \geq N$ $\Sigma_k = \emptyset$.

If $a \in \Sigma_k$, then k is a rank of a. Note that each symbol in Σ may have several ranks.

2.2. Terminology. We shall use the letter e to denote a fixed symbol of rank 0 and of no other rank. This symbol e will be used on nodes of trees to "stand for" the empty string λ.

2.3. Definition. Let Σ be a ranked alphabet. The set of trees over Σ, denoted by T_Σ, is the language over the alphabet $\Sigma \cup \{[\,, \,]\}$ defined recursively as follows.

(i) $\Sigma_0 \subseteq T_\Sigma$,

(ii) for each $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$, $a[t_1 \ldots t_k] \in T_\Sigma$.

2.4. Definition. Let Σ be a ranked alphabet and let S be a set of trees. The set of trees over Σ indexed by S, denoted by $T_\Sigma(S)$, is defined recursively as follows.

(i) $S \cup \Sigma_0 \subseteq T_\Sigma(S)$,

(ii) for each $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(S)$, $a[t_1 \ldots t_k] \in T_\Sigma(S)$.

2.5. Definition. Let Σ be a ranked alphabet. We define a mapping $fr: T_\Sigma \to \Sigma_0^*$ recursively as follows.

(i) For $a \in \Sigma_0$, $fr(a) = \begin{cases} a & \text{if } a \neq e \\ \lambda & \text{if } a = e \end{cases}$

(ii) For $k \geq 1$, $a \in \Sigma_k$ and $t_1, t_2, \ldots, t_k \in T_\Sigma$, $fr(a[t_1 \ldots t_k]) = fr(t_1) \cdot fr(t_2) \ldots fr(t_k)$.

For $t \in T_\Sigma$, $fr(t)$ is called the frontier of t.

Here we distinguish the symbol e from all other symbols: when taking the frontier of a tree, e is erased.

We now define "vertical strings".

2.6. Definition. Let Δ be an alphabet. The monadic ranked alphabet corresponding to Δ, denoted by $m(\Delta)$, is defined by $(m(\Delta))_0 = \{e\}$, $(m(\Delta))_1 = \Delta$ and $(m(\Delta))_k = \emptyset$ for $k \geq 2$.

The trees in $T_{m(\Delta)}$ will be called monadic trees.

For each string $w \in \Delta^*$ we define the corresponding monadic tree $\tilde{w} \in T_{m(\Delta)}$ recursively as follows.

(i) $\tilde{\lambda} = e$,

(ii) for each $a \in \Delta$ and $w \in \Delta^*$,
$\widetilde{aw} = a[\tilde{w}]$.

For example, if $w = abc$ then $\tilde{w} = a[b[c[e]]]$. Obviously the mapping $w \to \tilde{w}$ is a bijection between Δ^* and $T_{m(\Delta)}$. Note that if Σ is a ranked alphabet such that $\Sigma_0 = \{e\}$ and $\Sigma_k = \emptyset$ for all $k \geq 2$, then $\Sigma = m(\Sigma_1)$.

2.7. Definition. Let Σ and Δ be ranked alphabets. A tree language is a subset of T_Σ. A tree transformation is a subset of $T_\Sigma \times T_\Delta$. However, in this paper, the term "tree transformation" will refer exclusively to those tree transformations that can be realized by the tree transducer defined in the next definition.

Let $X = \{x_1, x_2, \ldots\}$ be a fixed infinite set of symbols and let $X_k = \{x_1, x_2, \ldots, x_k\}$. All symbols in X, when used as labels on trees, are of rank 0 (only). We shall often use x to denote x_1.

2.8. Definition. A tree transducer is a 5-tuple $M = (\Sigma, \Delta, Q, Q_{in}, R)$ where Σ is a ranked input alphabet, Δ is a ranked output alphabet, Q is a finite set of states (of rank 1), $Q_{in} \subseteq Q$ is a set of initial states and R is a finite set of rules of the form (i) or (ii):

(i) $q[a[x_1 x_2 \ldots x_k]] \to t$

with $q \in Q$, $k \geq 1$, $a \in \Sigma_k$ and $t \in T_\Delta(Q[X_k])$

(where $Q[X_k]$ is the set of trees $\{q[x_i] \mid q \in Q, 1 \leq i \leq k\}$)

or

(ii) $q[a] \to t$

with $q \in Q$, $a \in \Sigma_0$ and $t \in T_\Delta$.

The operation of M is modeled by a relation $\overset{\Rightarrow}{M}$ between trees in $T_\Delta(Q[T_\Sigma])$, where $Q[T_\Sigma]$ is the set of trees $\{q[t] \mid q \in Q, t \in T_\Sigma\}$. The relation $\overset{\Rightarrow}{M}$ is defined as follows.

For $u, v \in T_\Delta(Q[T_\Sigma])$, $u \overset{\Rightarrow}{M} v$ if and only if either there is a rule of the form (i) in R and there are strings α, β and trees $t_1, \ldots, t_k \in T_\Sigma$ such that $u = \alpha q[a[t_1 \ldots t_k]]\beta$ and $v = \alpha \bar{t} \beta$, where $\bar{t}$ is the result of substituting t_i for each occurrence of x_i in t (for all i, $1 \leq i \leq k$),

or there is a rule of the form (ii) in R and there are strings α and β such that $u = \alpha q[a]\beta$ and $v = \alpha t \beta$.

As usual $\overset{*}{\Rightarrow}_M$ denotes the transitive-reflexive closure of $\overset{\Rightarrow}{M}$. The tree transformation realized by M, denoted by $T(M)$, is

$$T(M) = \{(s, t) \in T_\Sigma \times T_\Delta \mid q[s] \overset{*}{\Rightarrow}_M t \text{ for some } q \text{ in } Q_{in}\}.$$

We shall use range(M) to denote the range of $T(M)$, thus range$(M) = \{t \in T_\Delta \mid (s, t) \in T(M) \text{ for some } s \in T_\Sigma\}$.

Some specific types of tree transducer are defined next.

2.9. Definition. Let $M = (\Sigma, \Delta, Q, Q_{in}, R)$ be a tree transducer.

M is called monadic if its input alphabet is monadic, that is, $\Sigma = m(\Omega)$ for some alphabet Ω. If, moreover, Ω is a singleton, then M is called unary.

M is called e-free if the symbol e is not in Δ.

M is called (partial) deterministic if (1) Q_{in} is a singleton, (2) no two different rules in R have the same left hand side.

M is called a finite tree automaton if all its rules are of the form $q[a[x_1 \ldots x_k]] \to a[q_1[x_1] \ldots q_k[x_k]]$ or $q[a] \to a$. In that case the domain (= range) of T(M) is called a recognizable tree language.

We now define the languages obtained by tree transformation of recognizable tree languages.

2.10. Definition. Let $M = (\Sigma, \Delta, Q, Q_{in}, R)$ be a tree transducer and let $U \subseteq T_\Sigma$ be a recognizable tree language. Then the set $M(U) = \{t \in T_\Delta \mid (s, t) \in T(M) \text{ for some } s \in U\}$ is called a surface tree language. (Note that, since T_Σ is recognizable, range(M) is a surface tree language.) Moreover, the set

$$fr(M(U)) = \{w \in \Delta_0^* \mid w = fr(t) \text{ for some } t \in M(U)\}$$

is called a tree transformation language.

2.11. Terminology. If M is an X tree transducer, where $X \in \{\text{monadic, unary, e-free, deterministic}\}^+$, then T(M) is called an X tree transformation and, for every recognizable tree language U, M(U) is called an X surface tree language and fr(M(U)) is called an X tree transformation language. (Note that, for instance, a monadic surface tree language is not a surface language consisting of monadic trees).

We finally need the notion of tree homomorphism.

2.12. Definition. Let Σ and Δ be ranked alphabets, and assume that each element of Σ has exactly one rank. Let $\hat{h}$ be a mapping from Σ into $T_\Delta(X)$ such that, for all $k \geq 1$ and $a \in \Sigma_k$, $\hat{h}(a) \in T_\Delta(X_k)$, and, for all $a \in \Sigma_0$, $\hat{h}(a) \in T_\Delta$. Then a mapping h, called a tree homomorphism, from T_Σ into T_Δ is determined from $\hat{h}$ as follows.

(i) For $a \in \Sigma_0$, $h(a) = \hat{h}(a)$.

(ii) For $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$, $h(a[t_1 \ldots t_k])$ is the result of substituting $h(t_i)$ for each occurrence of x_i in $\hat{h}(a)$ (for each i, $1 \leq i \leq k$).

The assumption that the elements of Σ have unique ranks is not essential but notationally more convenient (moreover we shall not need the general case).

The simplest possible generalization of the notion of string homomorphism to the tree case is the frontier-preserving tree homomorphism defined next.

2.13. Definition. A tree homomorphism h from T_Σ into T_Δ is called frontier-preserving if

(i) $\Sigma_0 \subseteq \Delta_0$ and, for each $a \in \Sigma_0$, $\hat{h}(a) = a$;

(ii) for each $k \geq 1$ and $a \in \Sigma_k$, $fr(\hat{h}(a)) = x_1 x_2 \ldots x_k$.

It is easy to see that, for any frontier-preserving tree homomorphism h from T_Σ into T_Δ and any tree $t \in T_\Sigma$, $fr(h(t)) = fr(t)$.

Note that a frontier-preserving homomorphism can erase symbols of rank 1 (only). This happens if, for some $a \in \Sigma_1$, $\hat{h}(a) = x_1$. Thus, monadic pieces of tree can be cut from a tree by a frontier-preserving homomorphism.

The following lemma, relating tree transformations and frontier-preserving homomorphisms, will be used in the sequel. Since its proof uses standard techniques, it is left to the reader.

2.14. Lemma. The class of tree transformations is closed under composition with frontier-preserving tree homomorphisms (formally: for each tree transducer M and each frontier-preserving tree homomorphism h, we can find a tree transducer N such that $T(N) = \{(s, h(t)) \mid (s, t) \in T(M)\}$). Moreover the same statement is true for the class of deterministic tree transformations. And, obviously, the same holds when the tree transformations are restricted to be e-free, monadic or unary.

ETOL systems

The ETOL system to be used differs slightly from the usual one in that we take a set of initial symbols instead of one initial word. It is well known that this does not influence the class of languages generated. Moreover, the sets of derivation trees are obviously changed in a trivial way only.

2.15. Definition. An ETOL system is a 4-tuple $G = (\Sigma, \mathcal{P}, S, \Delta)$, where Σ is the alphabet, $\Delta \subseteq \Sigma$ is the target alphabet, $S \subseteq \Sigma$ is the set of axioms, and $\mathcal{P}$ is a finite set of tables, each table P in $\mathcal{P}$ being a finite subset of $\Sigma \times \Sigma^*$ such that for each $a \in \Sigma$ there is a $w \in \Sigma^*$ with $(a, w) \in P$. An element (a, w) of a table will be written as $a \to w$ and will be called a production.

For $u \in \Sigma^*$, $v \in \Sigma^*$ and $P \in \mathcal{P}$, we write $u \overset{P}{\Rightarrow} v$ if there exist $n \geq 0$, $u_1, \ldots, u_n \in \Sigma$ and $v_1, \ldots, v_n \in \Sigma^*$ such that $u = u_1 \ldots u_n$, $v = v_1 \ldots v_n$ and, for each $1 \leq i \leq n$, $u_i \to v_i$ is in P.

We write $u \Rightarrow v$ if $u \overset{P}{\Rightarrow} v$ for some P in $\mathcal{P}$. As usual $\overset{*}{\Rightarrow}$ denotes the transitive-reflixive closure of $\Rightarrow$. The language generated by G, denoted by $L(G)$, is defined to be $\{w \in \Delta^* \mid a \overset{*}{\Rightarrow} w$ for some a in $S\}$. $L(G)$ is called an ETOL language.

2.16. Terminology. Given an ETOL system $G = (\Sigma, \mathcal{P}, S, \Delta)$, we shall often use the set $\mathcal{P}$ as an alphabet (formally one should introduce a new set of symbols in one-to-one correspondence with the elements of $\mathcal{P}$). Thus a sequence of tables is an element of $\mathcal{P}^*$ and will be called a control word. It is often convenient to have a relation $\overset{\Pi}{\Rightarrow}$ for each control word π as follows: if $\pi = P_1 P_2 \ldots P_n$ $(n \geq 0)$, then $u \overset{\Pi}{\Rightarrow} v$ if and only if there are $u_1, u_2, \ldots, u_{n+1}$ such that $u_1 = u$, $u_{n+1} = v$ and $u_i \overset{P_i}{\Rightarrow} u_{i+1}$ for $1 \leq i \leq n$. Obviously, $u \overset{*}{\Rightarrow} v$ if and only if there is a control word π such that $u \overset{\Pi}{\Rightarrow} v$.

2.17. Definition. Let $G = (\Sigma, \mathcal{P}, S, \Delta)$ be an ETOL system.

G is called an EOL system if $\mathcal{P}$ is a singleton.

G is called deterministic if (1) S is a singleton, and (2) for every $P \in \mathcal{P}$ and $a \in \Sigma$ there is exactly one $w \in \Sigma^*$ such that $a \to w$

is in P. A deterministic E(T)OL system is also called an ED(T)OL system.

G is called propagating (or an EPTOL system) if for no $P \in \mathcal{P}$ and $a \in \Sigma$, $a \to \lambda$ is in P.

We now formalize the intuitive description of an ETOL derivation tree in [4].

2.18. Definition. Let $G = (\Sigma, \mathcal{P}, S, \Delta)$ be an ETOL system. Define Ω to be the ranked alphabet $\Sigma \cup \{e\}$ such that $\Omega_0 = \Delta \cup \{e\}$, $\Omega_1 = \{a \in \Sigma \mid$ there are $b \in \Sigma \cup \{\lambda\}$ and $P \in \mathcal{P}$ such that $a \to b$ is in $P\}$ and, for $k \geq 2$, $\Omega_k = \{a \in \Sigma \mid a \to w$ is in P for some $P \in \mathcal{P}$ and some $w \in \Sigma^+$ of length $k\}$.

For $a \in \Sigma$ and $\pi \in \mathcal{P}^*$, the set of derivation trees with top a and control word π, denoted by $D^a_\pi(G)$, is defined recursively as follows:

(i) for $a \in \Delta$, $a \in D^a_\lambda$;

(ii) for $a \in \Sigma$ and $P \in \mathcal{P}$,
if $a \to \lambda$ is in P, then $a[e]$ is in $D^a_{P\pi}(G)$
for all $\pi \in \mathcal{P}^*$;

(iii) for $n \geq 1$, $a, a_1, \ldots, a_n \in \Sigma$, $P \in \mathcal{P}$ and $t_1, \ldots, t_n \in T_\Omega$,
if $a \to a_1 \ldots a_n$ is in P and $t_i \in D^{a_i}_\pi(G)$ for $1 \leq i \leq n$,
then $a[t_1 \ldots t_n]$ is in $D^a_{P\pi}(G)$.

The set of derivation trees of G, denoted by $D(G)$, is defined by $D(G) = \bigcup_{\substack{a \in S \\ \pi \in \mathcal{P}^*}} D^a_\pi(G)$.

Note that, if G is propagating, then we do not need symbol e and (ii) above.

Note that $D(G)$ contains only derivation trees corresponding to "succesfull derivations" of G, i.e. derivations $a \overset{*}{\Rightarrow} w$ for $a \in S$ and $w \in \Delta^*$.

2.19. Lemma. For any ETOL system G, $fr(D(G)) = L(G)$.

Proof. It is left to the reader to show that, for $a \in \Sigma$, $w \in \Delta^*$ and $\pi \in \mathcal{P}^*$,

$a \overset{\pi}{\Rightarrow} w$ if and only if $w = fr(t)$ for some t in $D^a_\pi(G)$

(for $\overset{\pi}{\Rightarrow}$, see Terminology 2.16).

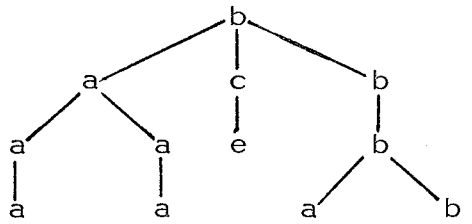
From this the lemma easily follows. ///

2.20. Example. Consider the ETOL system

$G = (\{a,b,c\}, \{P,Q\}, \{a\}, \{a,b\}$ where

$P = \{a \to aa\ ,\ b \to b\ ,\ b \to acb\ ,\ c \to \lambda\}$ and

$Q = \{a \to a\ ,\ b \to ab\ ,\ b \to acb\ ,\ c \to c\}$. Then the word $aaab$ can be derived from b in G as follows: $b \overset{P}{\Rightarrow} acb \overset{P}{\Rightarrow} aab \overset{Q}{\Rightarrow} aaab$. Thus the corresponding derivation tree $t_0 = b[a[a[a]a[a]]c[e]b[b[ab]]]$ is in D^b_{PPQ}. Note that t_0 is also in D^b_{QPQ}. A picture of t_0 is



///

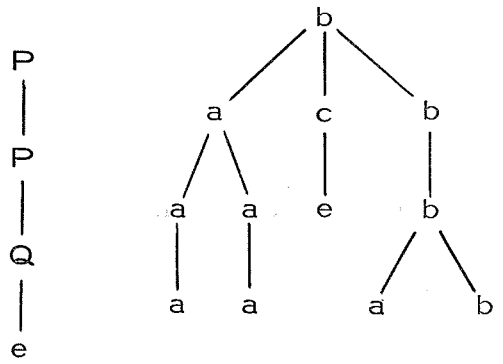
Note that, in the above example, the tree t_0 does not completely determine the derivation, since it may not be possible to determine the actual sequence of tables applied. Therefore, to obtain a complete

description of the derivation, we have to add the control word to the derivation tree.

2.21. Definition. Let $G = (\Sigma, \mathfrak{P}, S, \Delta)$ be an ETOL system. The set of complete derivation trees of G, denoted by $D_{cpl}(G)$, is defined by $D_{cpl}(G) = \{(\tilde{\pi}, t) \mid t \in D_{\pi}^{a} \text{ for some } a \in S\}$.

Thus a complete derivation tree actually consists of two trees: the first is a "vertical control word" and the second an ordinary derivation tree with that control word.

2.22. Example. The complete derivation tree corresponding to the derivation in Example 2.20 is $(P[P[Q[e]]], t_0)$ or in a picture



///

3. Surface trees and parallel derivation trees

A connection between sets of derivation trees of ETOL systems on the one hand and surface tree languages on the other hand will be established in this section. It will be shown that derivation trees of ETOL systems are quite closely related to trees obtained by finite state transformation of monadic trees.

We start by showing that any ETOL system may be viewed as a top-down tree transducer.

3.1. Theorem. For each ETOL system G there is a monadic tree transducer M such that $T(M) = D_{cpl}(G)$.

Proof. The idea of the proof is that the tables of G may be viewed as input symbols to the monadic tree transducer M. Thus a control word of G is viewed as a monadic input tree to M. The symbols of the alphabet of G will be used as the states of the tree transducer. The rules of M are constructed from the productions of G in such a way that, whenever G produces a string according to some control word, then M transforms this control word into the corresponding derivation tree. Formally we proceed as follows.

Let $G = (\Sigma, \mathcal{P}, S, \Delta)$ be the ETOL system. We now construct a tree transducer M such that $T(M) = D_{cpl}(G)$.
Let $M = (m(\mathcal{P}), \Omega, \overline{\Sigma}, \overline{S}, R)$ where $\Omega = \Sigma \cup \{e\}$, the ranking of Ω being defined as in Definition 2.18, $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$, $\overline{S} = \{\overline{a} \mid a \in S\}$ and R is obtained as follows:

(1) for every $k \geq 1$, $a, a_1, \ldots, a_k \in \Sigma$ and $P \in \mathcal{P}$, if $a \to a_1 \ldots a_k$ is a production in P, then $\bar{a}[P[x]] \to a[\bar{a}_1[x] \ldots \bar{a}_k[x]]$ is in R;

(2) for every $a \in \Sigma$ and $P \in \mathcal{P}$, if $a \to \lambda$ is a production in P, then $\bar{a}[P[x]] \to a[e]$ is in R;

(3) for every $a \in \Delta$, $\bar{a}[e] \to a$ is in R.

It is left to the reader to show that, for every $a \in \Sigma$, $\pi \in \mathcal{P}^*$ and $t \in T_\Omega$,

(*) $\bar{a}[\tilde{\pi}] \overset{*}{\underset{M}{\Rightarrow}} t$ if and only if $t \in D^a_\pi(G)$.

The proof of (*) is by straightforward induction on the length of π. From (*) it follows that

$$\begin{aligned} T(M) &= \{(\tilde{\pi}, t) \mid \bar{a}[\tilde{\pi}] \overset{*}{\Rightarrow} t \text{ for some } a \in S\} \\ &= \{(\tilde{\pi}, t) \mid t \in D^a_\pi(G) \text{ for some } a \in S\} \\ &= D_{cpl}(G), \end{aligned}$$

and this proves the theorem. ///

It immediately follows from this theorem that the set of derivation trees of an ETOL system is a monadic surface tree language, and that the language generated by an ETOL system is a monadic tree transformation language. These facts are stated in the following corollaries.

3.2. Corollary. For any ETOL system G, $D(G)$ is a monadic surface tree language.

3.3. Corollary. Every ETOL language is a monadic tree transformation language.

Proof (of both corollaries). Let M be the tree transducer such that $T(M) = D_{cpl}(G)$. Then obviously $D(G) = \text{range}(M)$ and so $D(G)$ is a monadic surface tree language. Furthermore, since $L(G) = fr(D(G))$, it follows that $L(G) = fr(\text{range}(M))$. Hence $L(G)$ is a monadic tree transformation language. ///

Next we consider some special cases of Theorem 3.1.

3.4. Remark. It is easy to see that, in Theorem 3.1,

(i) if G is propagating, then M is e-free;

(ii) if G is deterministic, then M is deterministic;

(iii) if G has one table only, then M is unary.

We shall now aim at a converse of Corollary 3.2.

When trying to relate monadic surface tree languages to sets of derivation trees of ETOL systems in a way similar to Theorem 3.1, we should be aware of the following facts:

(1) in a derivation tree of an ETOL system all paths through the tree (not terminating in e) are of equal length;

(2) the surface language is obtained by transforming a recognizable set of input trees, whereas in an ETOL system there are no restrictions on the set of control words;

(3) in a tree of a surface language the symbol e may appear anywhere on its frontier.

The first fact suggests that, in order to obtain trees with paths of different length, we might consider tree homomorphic images of ETOL derivation tree languages. That this can be done, when disregarding the other two facts, is shown next.

3.5. Lemma. For each monadic e-free tree transducer M there is a propagating ETOL system G and a frontier-preserving tree homomorphism h such that range(M) = h(D(G)) .

Proof. We shall use the well known technique of decomposition. M will be decomposed into two parts: very roughly speaking, the first part imitates M , but, at each step, instead of outputting the right hand side of the applied rule, it outputs one node, labeled by the rule itself; the second part is a tree homomorphism that transforms each node into the right hand side of the rule labeling that node. Since the first part emits one node at each step it may be described as an ETOL system generating a derivation tree. Note that, if one wants the homomorphism to be frontier-preserving, then, whenever the tree transducer M outputs symbols of rank 0 , the ETOL system is forced to "prolong" the involved paths until they have the same length as all other paths; however, these paths may be cut off at the right length again by the "monadic erasing" capability of the frontier-preserving tree homomorphism. We now proceed to the formal construction.

Let $M = (m(\mathfrak{P}), \Sigma, Q, Q_{in}, R)$ be an arbitrary monadic e-free tree transducer, where $\mathfrak{P}$ is some alphabet. We first construct an

ETOL system G for the "first part" of M (see above). In G, the elements of R will be used as symbols and the elements of $\mathcal{P}$ will be used as (names of) tables.

For any $r \in R$ let state(r) denote the (unique) state occurring in the left hand side of r.

Let G be the ETOL system $(R \cup \Sigma_0 \cup \overline{\Sigma}_0 \cup \{\#\}, \mathcal{P} \cup \{f\}, S, \Sigma_0)$, where $\#$ and f are "new" symbols, $\overline{\Sigma}_0 = \{\overline{a} \mid a \in \Sigma_0\}$, S is the set of all r in R such shat $\text{state}(r) \in Q_{in}$, and the tables of G are constructed as follows.

(i) Let $r \in R$ be of the form $q[P[x]] \to t$ for some $q \in Q$, $P \in \mathcal{P}$ and $t \in T_\Sigma(Q[x])$. (Note that $Q[x] = \{q[x] \mid q \in Q\}$. Recall that $x = x_1$.) Then each production $r \to w$ is in table P, where w is in the set flat(t) defined next.

For any $t \in T_\Sigma(Q[x])$ we define the finite set of strings flat(t) recursively as follows:

(a) for each $q' \in Q$, $\text{flat}(q'[x])$ is the set of all $r \in R$ such that $\text{state}(r) = q'$;

(b) for each $a \in \Sigma_0$, $\text{flat}(a) = \{\overline{a}\}$;

(c) for every $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(Q[x])$, $\text{flat}(a[t_1 \ldots t_k]) = \text{flat}(t_1) \ldots \text{flat}(t_k)$.

Thus an element of flat(t) is obtained from t by taking its "frontier" (considering elements of $Q[x]$ as symbols), putting bars on elements of Σ_0 and replacing each $q'[x]$ by some rule involving q' in its left hand side.

(ii) Let $r \in R$ be of the form $q[e] \to t$ for some $q \in Q$ and $t \in T_\Sigma$. Then $r \to fr(t)$ is a production in table f.

(iii) For every $a \in \Sigma_0$, the production $\overline{a} \to \overline{a}$ is in table P for each $P \in \mathcal{P}$, and the production $\overline{a} \to a$ is in table f.

(iv) For every $a \in R \cup \overline{\Sigma}_0 \cup \Sigma_0 \cup \{\#\}$ and every $P \in \mathcal{P} \cup \{f\}$, if it does not follow from (i) - (iii) that there is a production with left hand side a in P, then $a \to \#$ is a production in table P.

This ends the construction of G. Note that G is propagating. Note also that for each rule r in R there is exactly one table in $\mathcal{P} \cup \{f\}$ in which there are rules $r \to w$ with $w \neq \#$. Note finally that in all tables the only production for an element a of $\Sigma_0 \cup \{\#\}$ is $a \to \#$.

We now construct the tree homomorphism h for the "second part" of M. Let Ω be the ranked alphabet $R \cup \overline{\Sigma}_0 \cup \Sigma_0$, where the ranks are defined such that all elements of $D(G)$ are trees over Ω (cf. Definition 2.18). Note that $\#$ does not appear in derivation trees of G and that we do not need e.

The homomorphism h is defined as follows.

(1) For each $a \in \Omega_0$ (that is, $a \in \Sigma_0$), $\hat{h}(a) = a$.

(2) For each $a \in \Sigma_0$, $\hat{h}(\overline{a}) = x_1$.

(3) Let $r \in R$, $r \in \Omega_k$ for some $k \geq 1$. Suppose that the right hand side of rule r is $t \in T_\Sigma(Q[X])$. Let $A = Q[X] \cup \Sigma_0$. Then, obviously, k is the number of occurrences of elements of A in t, and these occurrences are ordered from left to right in t. (Obviously, for $a \in \Sigma_0$, we only count those occurrences of a that are not followed by the symbol $[$). Let $\overline{t}$ be the tree obtained from t by replacing the i^{th} occurrence of an element of A in t by x_i, for each i, $1 \leq i \leq k$. Then we define $\hat{h}(r) = \overline{t}$.

This ends the construction of h. Note that h is frontier-preserving (in (3), $fr(\bar{t}) = x_1 \ldots x_k$). Note that (2) provides the cutting off of "prolonged paths" to the correct length.

Now the following statement holds.

(*) For every $q \in Q$, $\pi \in \mathcal{P}^*$ and $t \in T_\Sigma$, $q[\tilde{\pi}] \overset{*}{\underset{M}{\Rightarrow}} t$ if and only if there is a tree s (in T_Ω) such that $t = h(s)$ and $s \in D^r_{\pi f}(G)$ for some $r \in R$ such that $q = state(r)$.

The proof of (*) is by the usual straightforward but awkward induction argument (on the length of π). In fact, in order to provide a complete formal proof of (*) we would have to define a lot of additional technical machinery. To avoid this we leave it to the reader to see that (*) is true. It follows easily from (*) that range(M) = h(D(G)); note that, by the construction of G, in a derivation in G of a string in Σ_0^* the last table applied is always f and all other applied tables are in $\mathcal{P}$. ///

3.6. Example. Consider the tree transducer $M = (m(\mathcal{P}), \Sigma, Q, Q_{in}, R)$ where $\mathcal{P} = \{P, T\}$, $\Sigma_0 = \{a, b\}$, $\Sigma_2 = \{b\}$, $Q = \{q_1, q_2\}$, $Q_{in} = \{q_1\}$ and R consists of the following rules:

r_{1P} : $q_1[P[x]] \to b[ab[q_1[x]q_2[x]]]$

r_{1T} : $q_1[T[x]] \to b$

r_{2T} : $q_2[T[x]] \to q_2[x]$

r_{2e} : $q_2[e] \to b$

The ETOL system G, constructed as in the proof of the preceding lemma is

$G = (\{r_{1P}, r_{1T}, r_{2T}, r_{2e}, a, b, \bar{a}, \bar{b}, \#\}\ ,\ \{P, T, f\}\ ,\ S, \{a, b\})$

where $S = \{r_{1P}, r_{1T}\}$ and

$P = \{r_{1P} \to \bar{a}\, r_{1P}\, r_{2T}\ ,\ r_{1P} \to \bar{a}\, r_{1P}\, r_{2e}\ ,\ r_{1P} \to \bar{a}\, r_{1T}\, r_{2T}\ ,$

$r_{1P} \to \bar{a}\, r_{1T}\, r_{2e}\ ,\ r_{1T} \to \#\ ,\ r_{2T} \to \#\ ,\ r_{2e} \to \#\ ,$

$a \to \#\ ,\ b \to \#\ ,\ \# \to \#\ ,\ \bar{a} \to \bar{a}\ ,\ \bar{b} \to \bar{b}\}\ ,$

$T = \{r_{1T} \to \bar{b}\ ,\ r_{2T} \to r_{2T}\ ,\ r_{2T} \to r_{2e}\ ,$

$r_{1P} \to \#\ ,\ r_{2e} \to \#\ ,\ a \to \#\ ,\ b \to \#\ ,\ \# \to \#\ ,$

$\bar{a} \to \bar{a}\ ,\ \bar{b} \to \bar{b}\}\ ,$

$f = \{r_{2e} \to b\ ,\ \bar{a} \to a\ ,\ \bar{b} \to b\ ,\ r_{1P} \to \#\ ,\ r_{1T} \to \#\ ,$

$r_{2T} \to \#\ ,\ a \to \#\ ,\ b \to \#\ ,\ \# \to \#\}\ .$

The frontier-preserving homomorphism h is given by

$\hat{h}(a) = a$, $\hat{h}(b) = b$, $\hat{h}(\bar{a}) = x_1$, $\hat{h}(\bar{b}) = x_1$,

$\hat{h}(r_{1P}) = b[x_1 b[x_2 x_3]]$, $\hat{h}(r_{1T}) = x_1$, $\hat{h}(r_{2T}) = x_1$ and

$\hat{h}(r_{2e}) = x_1$.

Consider, as an example, M working on the input

$P[T[e]]$:

$$q_1[P[T[e]]] \underset{M}{\Rightarrow} b[ab[q_1[T[e]]q_2[T[e]]]]$$

$$\underset{M}{\Rightarrow} b[ab[bq_2[T[e]]]]$$

$$\Rightarrow b[ab[bq_2[e]]]$$

$$\Rightarrow b[ab[bb]]\ .$$

The "corresponding" derivation of the frontier abb of this tree in G is:

$$r_{1P} \overset{P}{\Rightarrow} \bar{a}\, r_{1T}\, r_{2T} \overset{T}{\Rightarrow} \overline{ab}\, r_{2e} \overset{f}{\Rightarrow} abb$$

with derivation tree

$r_{1P}[\bar{a}[\bar{a}[a]]\ r_{1T}[\bar{b}[b]]\ r_{2T}[r_{2e}[b]]]$. It is easy to see that the image of this derivation tree under h is $b[ab[bb]]$. ///

Lemma 3.5. is true not only for ranges of monadic tree transformations but also for arbitrary monadic surface tree languages. To show this we prove the following lemma.

3.7. Lemma. Every monadic e-free surface tree language is the range of some monadic e-free tree transformation.

Proof. Let $M = (m(\mathcal{P}), \Sigma, Q, Q_{in}, R)$ be a monadic e-free tree transducer. We first note that we may assume M to be nondeleting, which means that, for each rule $q[P[x]] \to t$, t contains at least one x (if t is in T_Σ, then take some symbol a occurring in $fr(t)$, replace it by $\bar{a}[x]$, where $\bar{a}$ is a new state, and add rules $\bar{a}[T[x]] \to \bar{a}[x]$, for all $T \in \mathcal{P}$, and $\bar{a}[e] \to a$). Now let U be a recognizable subset of $T_{m(\mathcal{P})}$. It is clear that $U = \{\tilde{\pi} \mid \pi \in V\}$, for a regular set of strings $V \subseteq \mathcal{P}^*$. Let $N = (Q_N, \mathcal{P}, \delta, q_0, F)$ be a deterministic finite automaton recognizing V. Thus δ is a mapping $Q_N \times \mathcal{P} \to Q_N$ and $U = \{\tilde{\pi} \mid \hat{\delta}(q_0, \pi) \in F\}$, where $\hat{\delta}$ is the usual extension of δ to $Q_N \times \mathcal{P}^*$. We now construct a monadic e-free tree transducer $\bar{M}$ such that $range(\bar{M}) = M(U)$.

Let $\bar{M} = (m(\mathcal{P}), \Sigma, Q \times Q_N, Q_{in} \times \{q_0\}, \bar{R})$ where $\bar{R}$ consists of the following rules:

(1) For every $q \in Q$, $P \in \mathcal{P}$, $t \in T_\Sigma(Q[x])$ and $q_1, q_2 \in Q_N$, if $q[P[x]] \to t$ is in R and $\delta(q_1, P) = q_2$, then $(q, q_1)[P[x]] \to \bar{t}$ is in $\bar{R}$, where $\bar{t}$ is the result of replacing each state q' in t by the pair of states (q', q_2);

(2) for every $q \in Q$, $t \in T_\Sigma$ and $q_1 \in Q_N$, if $q[e] \to t$ is in R and $q_1 \in F$, then $(q, q_1)[e] \to t$ is in $\bar{R}$.

It is easy to see that this $\overline{M}$ satisfies the requirement. In fact, since M was nondeleting, $\overline{M}$ is able to check whether the input tree belongs to U while simulating M. ///

At the ETOL side this lemma corresponds to the fact that ETOL is "closed under regular control", that is, if one restricts the set of control words of an ETOL system to some regular language, then the resulting language is still an ETOL language [5].

We can now formulate our main theorem, relating monadic surface tree languages to ETOL derivation tree languages.

3.8. Theorem. A tree language is a monadic e-free surface tree language if and only if it is a frontier-preserving homomorphic image of the set of derivation trees of some propagating ETOL system.

Proof. The only-if part follows directly from Lemma's 3.5 and 3.7. To prove the if-part, note that, by Corollary 3.2 and Remark 3.4 (i), the set of derivation trees of any propagating ETOL system is a monadic e-free surface tree language. Since the class of tree transformations is closed under composition with a frontier-preserving tree homomorphism (see Lemma 2.14), the result follows. ///

The string languages involved are related as follows.

3.9. Theorem.

(i) A language is a monadic e-free tree transformation language if and only if it is an EPTOL language.

(ii) A language is a monadic tree transformation language if and only if it is an ETOL language.

Proof.

(i) (If). See Corollary 3.3 and Remark 3.4 (i).

(Only if). Let U be a monadic e-free surface tree language. We have to show that $fr(U)$ is an EPTOL language. By Theorem 3.8, $U = h(D(G))$ for some EPTOL system G and frontier-preserving homomorphism h. Hence $fr(U) = fr(h(D(G))) = fr(D(G)) = L(G)$.

(ii) (If). See Corollary 3.3.

(Only if). Let U be a monadic surface tree language, eventually involving e. We have to show that $fr(U)$ is an ETOL language. Change everywhere in U the symbol e into the new symbol $\bar{e}$. Let $\bar{U}$ be the resulting tree language. Obviously $\bar{U}$ is a monadic e-free surface tree language and $fr(U) = p(fr(\bar{U}))$, where p is the string homomorphism such that $p(\bar{e}) = \lambda$ and $p(a) = a$ for all other symbols involved. By (i) of this theorem $fr(\bar{U})$ is an EPTOL language. Now, since the class of ETOL languages is closed under homomorphisms [8], if follows that $p(fr(\bar{U})) = fr(U)$ is an ETOL language.

///

It is known that if L is an ETOL language, then $L - \{\lambda\}$ is an EPTOL language [8]. Hence, by Theorem 3.9, the use of the symbol e in monadic tree transducers gives us no extra power (except for λ).

Note that in Theorem 3.9 (ii) and in the preceding remark we used results about ETOL (we did not do so far). One could also prove that, for instance, EPTOL = ETOL (modulo λ) by the use of tree transformation arguments (using composition results concerning top-down and bottom-up tree transducers). Proofs like this would perhaps provide more insight in known results, just as properties of recogni-

zable tree languages give us more insight into certain results about context-free languages (see [14]).

We now consider some restricted cases of Theorems 3.8 and 3.9. We start by investigating determinism.

3.10. Theorem. A tree language is a deterministic monadic e-free surface tree language if and only if it is a frontier-preserving homomorphic image of the set of derivation trees of some deterministic propagating ETOL system.

Proof.

(If). As in the proof of Theorem 3.8 (see Remark 3.4 (ii)).

(Only if). Since the deterministic version of Lemma 3.7 clearly holds, we only have to prove the result corresponding to Lemma 3.5.

Let $M = (m(\mathfrak{P}), \Sigma, Q, Q_{in}, R)$ be an arbitrary deterministic monadic e-free tree transducer. Let $G = (\Delta, \mathfrak{P} \cup \{f\}, S, \Sigma_0)$, where $\Delta = R \cup \Sigma_0 \cup \overline{\Sigma}_0 \cup \{\#\}$, be the ETOL system as constructed in the proof of Lemma 3.5. In general, the tables in $\mathfrak{P}$ do not satisfy the determinism requirement (but f does). Construct the deterministic ETOL system $G' = (\Delta, \mathfrak{P}' \cup \{f\}, S, \Sigma_0)$, where $\mathfrak{P}' = \{P' \subseteq \Delta \times \Delta^* \mid P' \subseteq P$ for some $P \in \mathfrak{P}$, and for each $a \in \Delta$ there is exactly one $w \in \Delta^*$ such that $(a, w) \in P'\}$. In other words, $\mathfrak{P}'$ is the set of all "deterministic sub-tables" of tables in $\mathfrak{P}$. It can easily be shown from the determinism of M that $D(G') = D(G)$. Hence, by the proof of Lemma 3.5, $range(M) = h(D(G)) = h(D(G'))$, where h is defined in that proof.

This proves the theorem. ///

For the corresponding languages we have

3.11. Theorem. A language is a deterministic monadic tree transformation language if and only if it is an EDTOL language.

Proof.

(If). See Corollary 3.3 and Remark 3.4 (ii).

(Only if). Similar to the only-if proof of Theorem 3.9 (ii), using the known fact that the class of EDTOL languages is closed under homomorphisms. ///

Let us now consider the unary case; that is, the case that the tree transducer is unary and the ETOL system is an EOL system (has one table only). We state the analogue of Theorem 3.8.

3.12. Theorem. A tree language is a unary e-free surface tree language if and only if it is a frontier-preserving homomorphic image of the set of derivation trees of some propagating EOL system.

Proof.

(If). As in the proof of Theorem 3.8 (cf. Remark 3.4 (iii)).

(Only if). Since it is easy to see that the unary analogue of Lemma 3.7 is valid, we only have to check whether the unary analogue of Lemma 3.5 is true. Note that, in the proof of the latter lemma, starting from a $\mathfrak{P}$ with one element P one does not obtain an EOL system G, since one "final table" f is added. However it is clear from the construction of G that the EOL system G' with table $P' = P \cup f$ has the same set of derivation trees as G.

///

The analogue of Theorem 3.9 is

3.13. Theorem. A language is a unary tree transformation language if and only if it is an EOL language.

Proof. Similar to the proof of Theorem 3.9 (ii), using the fact that the class of EOL languages is closed under homomorphisms [10].

///

Finally we consider the deterministic unary case. Let an HEDOL language be one which is obtained by applying a (string) homomorphism to an EDOL language. We now show that in the deterministic unary case we obtain the HEDOL languages as tree transformation languages.

3.14. Theorem. A language is a deterministic unary tree transformation language if and only if it is a HEDOL language.

Proof.

(If). Given an EDOL system G and a homomorphism h, it is easy to construct an appropriate tree transducer such that the frontier of its range is $h(L(G))$. In fact, one can use the construction of Theorem 3.1, where each rule $\overline{a}[e] \to a$ should be replaced by the rule $\overline{a}[e] \to h(a)$ if $h(a) \neq \lambda$, and by $\overline{a}[e] \to e$ if $h(a) = \lambda$.

(Only if). Since the needed version of Lemma 3.7 is obviously true, it suffices to show that the frontier of the range of a given deterministic unary tree transducer $M = (m(\mathfrak{P}), \Sigma, Q, Q_{in}, R)$ is an HEDOL language. $\mathfrak{P}$ is a singleton, say $\mathfrak{P} = \{P\}$. Construct the EDOL system $G = (Q \cup \Sigma_0 \cup \{\#\}, P, Q_{in}, Q \cup \Sigma_0)$, where the productions of P are defined as follows.

(1) If $q[P[x]] \to t$ is in R, then $q \to f(t)$ is in R, where, for $t \in T_\Sigma(Q[x])$, $f(t)$ is defined by

(i) $f(q'[x]) = q'$ for $q' \in Q$,

(ii) $f(a) = a$ for $a \in \Sigma_0$, and

(iii) $f(a[t_1 \ldots t_k]) = f(t_1) \ldots f(t_k)$.

(2) For all $a \in \Sigma_0 \cup \{\#\}$, $a \to a$ is in P.

(3) If there is no rule in R with left hand side $q[P[x]]$, then $q \to \#$ is in P.

This ends the construction of G. Now let h be the string homomorphism from $(Q \cup \Sigma_0)^*$ into Σ_0^* such that $h(a) = a$ for all $a \in \Sigma_0$ and, if $q[e] \to t$ is in R, then $h(q) = fr(t)$. It is left to the reader to show that $fr(\text{range}(M)) = h(L(G))$. ///

We note that it can be proved in the same way that in the e-free case one obtains the class of NEDOL languages (those languages which are obtained by applying a nonerasing homomorphism to an EDOL language), which is a proper sub-class of the class of HEDOL languages (see [6]; actually HEDOL = HDOL and NEDOL = NDOL, where the disappearance of the E means that, in the EDOL system $G = (\Sigma, \mathcal{P}, S, \Delta)$ involved, $\Delta = \Sigma$). Thus the use of e is significant in the case of deterministic unary tree transformations.

To conclude this section we consider a generalization of the ETOL system to trees: the so called ETOLT system [2]. In an ETOLT system productions are applied, in parallel, at the frontiers of trees. We slightly change the definition of an ETOLT system, given in [2], without influencing the class of languages generated.

3.15. Definition. An ETOLT system is a 4-tuple $G = (\Sigma, \mathcal{P}, S, \Delta)$, where Σ is the ranked alphabet, $\Delta \subseteq \Sigma_0$ is the target alphabet, $S \subseteq \Sigma_0$ is the set of axioms, and $\mathcal{P}$ is a finite set of tables, each table P in $\mathcal{P}$ being a finite subset of $\Sigma_0 \times T_\Sigma$ such that for each $a \in \Sigma_0$ there is a $t \in T_\Sigma$ with $(a,t) \in P$. An element (a,t) of a table will be written as $a \to t$ and called a production.

For $P \in \mathcal{P}$ the relation $\overset{P}{\Rightarrow}$ on T_Σ is defined recursively as follows:

(i) for each $a \in \Sigma_0$ and $t \in T_\Sigma$,
if $a \to t$ is in P, then $a \overset{P}{\Rightarrow} t$;

(ii) for every $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k, s_1, \ldots, s_k \in T_\Sigma$,
if $t_i \overset{P}{\Rightarrow} s_i$ for each i, $1 \leq i \leq k$, then
$a[t_1 \ldots t_k] \overset{P}{\Rightarrow} a[s_1 \ldots s_k]$.

We write $t \Rightarrow s$ if $t \overset{P}{\Rightarrow} s$ for some P in $\mathcal{P}$. As usual $\overset{*}{\Rightarrow}$ denotes the transitive-reflexive closure of $\Rightarrow$. The tree language generated by G, denoted by $L(G)$, is defined to be $\{t \in T_\Sigma \mid fr(t) \in \Delta^*$ and $a \overset{*}{\Rightarrow} t$ for some a in $S\}$. $L(G)$ is called an ETOLT language.

ETOLT languages are related to surface tree languages in an obvious way.

3.16. Theorem. A tree language is an ETOLT language if and only if it is a monadic e-free surface tree language.

Proof. We only provide the constructions and leave the proof of correctness of these constructions to the reader. The constructions are in fact similar to those of Theorem 3.1 and Lemma 3.5.

(Only if). Let $G = (\Sigma, \mathcal{P}, S, \Delta)$ be an ETOLT system, We construct a monadic e-free tree transducer $M = (m(\mathcal{P}), \Sigma, \overline{\Sigma}, \overline{S}, R)$ where

$\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$, $\overline{S} = \{\overline{a} \mid a \in S\}$ and R is obtained as follows:

(1) for every $a \in \Sigma_0$, $t \in T_\Sigma$ and $P \in \mathcal{P}$,
if $a \to t$ is a production in P , then
$\overline{a}[P[x]] \to x(t)$ is in R , where $x(t)$ is
defined recursively by
(i) for $a \in \Sigma_0$, $x(a) = \overline{a}[x]$;
(ii) for $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$,
$x(a[t_1 \ldots t_k]) = a[x(t_1) \ldots x(t_k)]$;

(2) for every $a \in \Delta$, $\overline{a}[e] \to a$ is in R .

Then range(M) = L(G) , and so L(G) is a monadic e-free surface tree language.

(If). By Lemma 3.7 we may restrict attention to ranges. Let $M = (m(\mathcal{P}), \Sigma, Q, Q_{in}, R)$ be an arbitrary monadic e-free tree transducer. We construct the ETOLT system $G = (\Omega, \mathcal{P} \cup \{f\}, Q_{in}, \Sigma_0)$, where $\Omega = \Sigma \cup Q \cup \overline{\Sigma}_0 \cup \{\#\}$, $\Omega_0 = \Sigma_0 \cup Q \cup \overline{\Sigma}_0 \cup \{\#\}$ and for $k \geq 1$ $\Omega_k = \Sigma_k$, and the productions of G are obtained as follows.

(1) For every $q \in Q$, $P \in \mathcal{P}$ and $t \in T_\Sigma(Q[x])$, if $q[P[x]] \to t$ is in R , then $q \to g(t)$ is in table P , where $g(t)$ is defined recursively as follows:

(i) for $a \in \Sigma_0$, $g(a) = \overline{a}$
(ii) for $q' \in Q$, $g(q'[x]) = q'$
(iii) for $k \geq 1$, $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(Q[x])$,
$g(a[t_1 \ldots t_k]) = a[g(t_1) \ldots g(t_k)]$.

(2) For every $q \in Q$ and $t \in T_\Sigma$,
if $q[e] \to t$ is in R , then $q \to t$ is in table f .

(3) For every $a \in \Sigma_0$,

the production $\bar{a} \to \bar{a}$ is in table P for each $P \in \mathcal{P}$, and the production $\bar{a} \to a$ is in table f.

(4) For every $a \in \Omega_0$ and every $P \in \mathcal{P} \cup \{f\}$, if it does not follow from (1) - (3) that there is a production with left hand side a in P, then $a \to \#$ is in P.

Then $L(G) = \text{range}(M)$, and so range(M) is an ETOLT language.

///

Of course, it follows from this theorem and Theorem 3.8 that a tree language is an ETOLT language if and only if it is a frontier-preserving homomorphic image of the set of derivation trees of some propagating ETOL system. It is left to the reader to check the deterministic and unary cases.

Conclusion.

We have shown the relationship between ETOL systems and top-down tree transducers working on monadic trees. On the one hand this relationship can be used to give "tree -oriented proofs" of, for instance, closure properties of ETOL langauges. On the other hand one may expect that the results and techniques used in ETOL language theory can be generalized to deal with surface tree languages obtained from arbitrary recognizable tree languages.

It remains to be seen whether the results of this paper can be extended to other classes of parallel rewriting systems and tree transducers. We can say, roughly speaking, that parallel rewriting

corresponds to copying in tree transducers. For instance, in [10], a distinction is made between top-down parallelism and bottom-up parallelism, the former referring to the parallelism in systems like ETOL and the latter referring to, for instance, the level grammars introduced in [11]. This distinction seems to correspond to the different kinds of copying of top-down transducers and bottom-up transducers as investigated in [1,3].

Acknowledgment.

I would like to thank Mogens Nielsen, Arto Salomaa, Erik M. Schmidt and Sven Skyum for their comments on the first version of this paper.

References

1. B. S. Baker, Tree transductions and families of tree languages,
Ph. D. Thesis, Harvard University, 1973.

2. P. J. Downey, Formal languages and recursion schemes,
Ph. D. Thesis, Harvard University, 1974.

3. J. Engelfriet, Bottom-up and top-down tree transformations - a comparison,
Memorandum nr. 19, Technical University Twente, Netherlands, 1971. (To appear in Math. Syst. Th.).

4. G. T. Herman and G. Rozenberg, Developmental systems and languages,
North-Holland Publishing Company, to appear.

5. M. Nielsen, EOL and ETOL systems with control devices,
Daimi Report PB-37, University of Aarhus, Denmark, 1974.

6. M. Nielsen, G. Rozenberg, A. Salomaa and S. Skyum,
Nonterminals, homomorphisms and codings in various OL systems,
ACTA Informatica, Vol. 4, Fasc. 1, 1974

7. W. C. Rounds, Mappings and grammars on trees,
Math. Syst. Th. 4(1970), 257-287.

8. G. Rozenberg, Extension of tabled OL-systems and langauges,
International Journal of Computer and Information Sciences 2 (1973), 311-334.

9. G. Rozenberg and A. Salomaa (eds), L Systems, Lecture Notes in Computer Science 15, Springer-Verlag, 1974.

10. A. Salomaa, Parallelism in rewriting systems, in: Automata, Languages and Programming (ed. J. Loeckx), Lecture Notes in Computer Science 14, Springer-Verlag, 1974, pp. 523-533.

11. S. Skyum, On Extensions of Algol-like languages, Information and Control 26, No. 1, Sept. 1974.

12. J. W. Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, JCSS 1 (1967), 317-322.

13. J. W. Thatcher, Generalized^2 sequential machine maps, JCSS 4 (1970), 339-367.

14. 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.

15. J. W. Thatcher and J. B. Wright, Generalized finite automata theory with an application to a decisionproblem of second-order logic, Math. Syst. Th. 2 (1968), 57-81.