Bottom-up and Top-down Tree Transformations— A Comparison

by

JOOST ENGELFRIET

Technical University Twente
Enschede, Netherlands

ABSTRACT

The top-down and bottom-up tree transducer are incomparable with respect to their transformation power. The difference between them is mainly caused by the different order in which they use the facilities of copying and nondeterminism. One can however define certain simple tree transformations, independent of the top-down/bottom-up distinction, such that each tree transformation, top-down or bottom-up, can be decomposed into a number of these simple transformations. This decomposition result is used to give simple proofs of composition results concerning bottom-up tree transformations.

A new tree transformation model is introduced which generalizes both the top-down and the bottom-up tree transducer.

Introduction. Finite state transformations (*fst*) of trees into trees were introduced and studied by Rounds [6] and Thatcher [8]. Finite state transformations are meant to be a model of the kind of tree transformations which are investigated, for instance, in the study of transformational grammars in linguistics and in the study of syntax-directed translations of context-free languages in compiler theory.

The *fst*, as they are discussed in [6, 7, 8, 9], process the input tree in a top-down mode, which is the way syntax-directed translation works. As a consequence, these *fst* fail to have some important properties. They are, for instance, unable to inspect a subtree in order to decide whether to delete it or not. Finite state transformations that process the input tree in a bottom-up mode (introduced in [10]) obviously have this ability, but on the other hand fail to have specific top-down properties.

In this paper we compare these two kinds of *fst*, and we define a generalized model which has all properties of both bottom-up and top-down *fst*. We show that each *fst* can be decomposed into "very simple" *fst* and we apply this decomposition to prove certain composition results concerning bottom-up *fst*.

In section 1 most of the necessary definitions are given.

In section 2 the incomparability of bottom-up and top-down *fst* is discussed.

In section 3 we define three classes of "simple" *fst* and show that each *fst* is a composition of at most four of such simple *fst*.

In section 4 we discuss composition of bottom-up *fst*.

In section 5 the "generalized *fst*" are introduced and investigated. We also introduce a new top-down model, "dual" with respect to the bottom-up model.

In section 6 we give a survey of decomposition results.

Section 7 contains a conclusion, acknowledgments and references.

1. Preliminaries. In this section we introduce some terms, definitions and facts which will be used in the rest of this paper. First we recall some terminology concerning sets, relations and strings. Then we discuss various notions concerning trees and tree transducers.

Sets, relations and strings. Inclusion of sets is denoted by $\subseteq$ and proper inclusion by $\subset$. The empty set is denoted by $\varnothing$. An ordered pair of objects a and b is denoted by $\langle a, b\rangle$. The cartesian product of sets A and B, denoted by $A \times B$, is the set $\{\langle a, b\rangle | a \in A \text{ and } b \in B\}$.

A relation R from A into B is any subset of $A \times B$. The inverse of R, denoted by R^{-1}, is $\{\langle b, a\rangle \in B \times A | \langle a, b\rangle \in R\}$. The domain of R, denoted by $\mathrm{dom}(R)$, is the set $\{a \in A | \langle a, b\rangle \in R \text{ for some } b \text{ in } B\}$. The range of R is the set $\{b \in B | \langle a, b\rangle \in R \text{ for some } a \text{ in } A\}$. For a in A, the image of a under R, denoted by $R(a)$, is the set $\{b \in B | \langle a, b\rangle \in R\}$, and, for $U \subseteq A$, the image of U under R, denoted by $R(U)$, is the set $\{b \in B | \langle a, b\rangle \in R \text{ for some } a \text{ in } U\}$.

If R_1 is a relation from A into B and R_2 a relation from B into C, then the composition of R_1 and R_2, denoted by $R_1 \circ R_2$, is the relation from A into C defined by

$$R_1 \circ R_2 = \{\langle a, c\rangle | \langle a, b\rangle \in R_1 \text{ and } \langle b, c\rangle \in R_2 \text{ for some } b \text{ in } B\}.$$

If F and G are classes of relations, then $F \circ G$ denotes the class of relations $\{R \circ S | R \in F \text{ and } S \in G\}$.

An alphabet Σ is any set (of "symbols"). Usually, an alphabet is assumed to be finite. The set of strings of symbols from Σ is denoted by Σ^*. A language over Σ is any subset of Σ^*. If A and B are languages over Σ, then their concatenation is denoted by AB. A language $\{w\}$ consisting of one word w only is often denoted by w. For instance, AwB denotes the concatenation of A, $\{w\}$ and B.

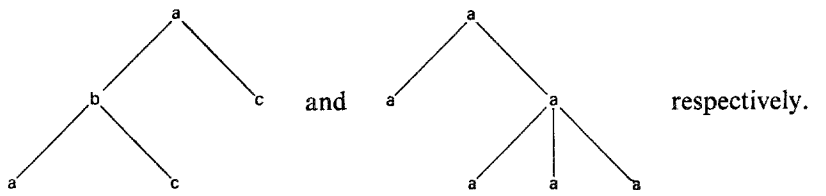
We shall often use the parentheses ")" and "(" as symbols in an alphabet, and simultaneously for their usual notational purposes. It is hoped that this will not lead to confusion. As an example, $A(B)$ might denote the image of the set B under the relation A, but it might also denote the concatenation of the languages A, $\{($, B and $)\}$.

Trees and tree transducers. An alphabet Σ is *ranked* if for each nonnegative integer k a subset Σ_k of Σ is specified, such that Σ_k is nonempty for a finite number of k's only. The elements of Σ_k are said to be of rank k. Note that the sets Σ_k need not be disjoint.

Given a ranked alphabet Σ, the set of all *trees over* Σ, denoted by T_Σ, is the language over the alphabet $\Sigma \cup \{(,)\}$ defined inductively as follows.

(1) If $\sigma \in \Sigma_0$, then σ is in T_Σ.
(2) If $k \geq 1$, $\sigma \in \Sigma_k$ and $t_1, \cdots, t_k \in T_\Sigma$, then $\sigma(t_1 \cdots t_k)$ is in T_Σ.

By this definition, trees over Σ are special kinds of strings over the alphabet $\Sigma \cup \{(,)\}$, but it is well known that the set of all such strings is in an effective one-to-one correspondence with the set of all finite rooted ordered directed trees, whose nodes are labeled with the elements of Σ, in such way that a node with k descendants is labeled by a symbol from Σ_k. For instance, if $\Sigma_0 = \{a, c\}$, $\Sigma_2 = \{a, b\}$ and $\Sigma_3 = \{a\}$, then $a(b(ac)c)$ and $a(aa(aaa))$ are in T_Σ and represent the trees



If t_1 and t_2 are trees over Σ such that $t_2 = \alpha t_1 \beta$ for certain $\alpha, \beta \in (\Sigma \cup \{(,)\})^*$, then that occurrence of t_1 is called a *subtree* of t_2.

A subset U of T_Σ is called a *tree language* over Σ, and a relation M from T_Σ into T_Δ (where Σ and Δ are ranked alphabets) is called a *tree transformation* from T_Σ into T_Δ.

Given some set S of symbols or strings, the set of all *trees over* Σ *indexed by* S, denoted by $T_\Sigma[S]$, is defined inductively as follows.

(1) $\Sigma_0 \cup S \subseteq T_\Sigma[S]$.
(2) If $k \geq 1$, $\sigma \in \Sigma_k$ and $t_1, \cdots, t_k \in T_\Sigma[S]$, then $\sigma(t_1 \cdots t_k) \in T_\Sigma[S]$.

Thus $T_\Sigma[S]$ is a language over the alphabet which is the union of $\Sigma \cup \{(,)\}$ and the alphabet of S. Note that if S is a set of symbols, then $T_\Sigma[S] = T_\Delta$, where Δ is the ranked alphabet such that $\Delta_0 = \Sigma_0 \cup S$ and, for $k \geq 1$, $\Delta_k = \Sigma_k$. Note also that $T_\Sigma[\varnothing] = T_\Sigma$, and that for instance $T_\Sigma[T_\Sigma] = T_\Sigma$.

The definition of a tree transducer will involve rules which employ variables ranging over trees. We now introduce the notion of a "semi-thue system with variables" of which the tree transducer will be a special case. Let us first recall the notion of semi-thue system.

A *semi-thue system* G is a pair $\langle A, R \rangle$ consisting of an alphabet A and a (not necessarily finite) subset R of $A^* \times A^*$, the elements of which are called *rules*.

A rule $\langle \alpha, \beta \rangle$ will be denoted by $\alpha \to \beta$. The binary relation $\underset{G}{\Rightarrow}$ in $A^* \times A^*$, is defined as follows: if $\alpha \to \beta$ is in R and φ, ψ are in A^*, then $\varphi\alpha\psi \underset{G}{\Rightarrow} \varphi\beta\psi$. The transitive-reflexive closure of $\underset{G}{\Rightarrow}$ is denoted as usual by $\overset{*}{\underset{G}{\Rightarrow}}$. If G is understood, then we write $\Rightarrow$ and $\overset{*}{\Rightarrow}$ rather than $\underset{G}{\Rightarrow}$ and $\overset{*}{\underset{G}{\Rightarrow}}$ respectively.

A *semi-thue system with variables* is a system $G = \langle A, X, D, R \rangle$ consisting of an alphabet A, a set X (of *variables*) disjoint with A, a mapping D from X into the power set of A^* and a finite set R (of *rules* or rule schemes) included in

$(A \cup X)^* \times (A \cup X)^*$. A rule $\langle \varphi, \psi \rangle$ will usually be denoted by $\varphi \to \psi$. Intuitively each variable x is meant to range over the language $D(x)$, and therefore each rule in R gives rise to a whole set of rules in $A^* \times A^*$ by replacing each variable x by a string from $D(x)$ throughout the rule. Formally, given a rule $\varphi \to \psi$ in R, we define $s(\varphi \to \psi)$ to be the set of all rules $\alpha \to \beta$ in $A^* \times A^*$, such that there exists a homomorphism h from $(A \cup X)^*$ into A^* with $h(x) \in D(x)$ for all x in X, $h(a) = a$ for all a in A, $h(\varphi) = \alpha$ and $h(\psi) = \beta$. For the set R of rules, let $s(R)$ denote $\bigcup_{r \in R} s(r)$. We now define the relations $\Rightarrow$ and $\overset{*}{\Rightarrow}$ of the semi-thue system G with variables to be those of the semi-thue system $H = \langle A, s(R) \rangle$. Thus, $\underset{G}{\Rightarrow} = \underset{H}{\Rightarrow}$ and $\overset{*}{\underset{G}{\Rightarrow}} = \overset{*}{\underset{H}{\Rightarrow}}$.

Example. Consider the following semi-thue system with variables: $G = \langle A, X, D, R \rangle$, where $A = \{1, (,), *\}$, $X = \{x, y\}$, $D(x) = D(y) = \{1\}^*$ and R consists of the rules

$$(x*y1) \to (x*y)x \quad \text{and} \quad (x*1) \to x.$$

Then $s(R)$ contains the following rules (among many others):

$$\begin{array}{lll} (11*111) \to (11*11)11 & & (h(x) = h(y) = 11 \text{ in the first rule}), \\ (11*11) \to (11*1)11 & & (h(x) = 11 \text{ and } h(y) = 1 \text{ in the first rule}), \\ (11*1) \to 11 & & (h(x) = 11 \text{ in the second rule}). \end{array}$$

Hence $(11*111) \underset{G}{\Rightarrow} (11*11)11 \underset{G}{\Rightarrow} (11*1)1111 \underset{G}{\Rightarrow} 111111$, and in general, for u, v and w in $\{1\}^*$, $(u*v) \overset{*}{\underset{G}{\Rightarrow}} w$ if and only if w is the product of u and v in tally notation.

Let from now on X be a *fixed* denumerable *set of variables* $x_1, x_2, x_3, \cdots$:

$$X = \{x_1, x_2, x_3, \cdots\}.$$

Moreover, let, for $k \geq 1$, $X_k = \{x_1, x_2, \cdots, x_k\}$, and let $X_0 = \varnothing$. We shall use x, y and z to denote arbitrary elements of X. Also, in examples, we use x, y and z to denote x_1, x_2 and x_3 respectively.

We now define the notion of tree transducer. A (finite state) *tree transducer* is a 5-tuple $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$, where

- Σ is a ranked alphabet (of *input symbols*),
- Δ is a ranked alphabet (of *output symbols*),
- Q is a ranked alphabet (of *states*), each element of which has rank 1 (thus $Q_1 = Q$ and $Q_k = \varnothing$ for all $k \neq 1$), and $Q \cap (\Sigma \cup \Delta) = \varnothing$,
- Q_d is a subset of Q (of *designated states*), and
- R is a finite set (of *rules*) such that either $R \subseteq Q(T_\Sigma[X]) \times T_\Delta[Q(X)]$ or $R \subseteq T_\Sigma[Q(X)] \times Q(T_\Delta[X])$.[1]

  In the former case M is called a *top-down* tree transducer and in the latter case a *bottom-up* tree transducer.

---

[1] $Q(T_\Sigma[X])$ is the set of trees $\{q(t) | q \in Q, t \in T_\Sigma[X]\}$. $Q(X)$ is the set of trees $\{q(x) | q \in Q, x \in X\}$. Thus both $Q(T_\Sigma[X])$ and $T_\Sigma[Q(X)]$ are subsets of $T_{\Sigma \cup Q}[X]$.

A tree transducer $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ will be viewed as representing the semi-thue system with variables $\langle A, X, D, R \rangle$ with $A = \Sigma \cup \Delta \cup Q \cup \{(,)\}$ and with all variables ranging over T_Σ in the top-down case and over T_Δ in the bottom-up case (that is, $D(x) = T_\Sigma$ for all x in X, and $D(x) = T_\Delta$ for all x in X, respectively). The relations $\underset{M}{\Rightarrow}$ and $\overset{*}{\underset{M}{\Rightarrow}}$ (or simply $\Rightarrow$ and $\overset{*}{\Rightarrow}$) in $A^* \times A^*$ are defined to be those of that semi-thue system with variables.

Let $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a tree transducer. The *tree transformation defined by M*, also denoted by M, is the following relation from T_Σ into T_Δ: in the top-down case, $M = \{\langle t, s \rangle \in T_\Sigma \times T_\Delta \mid q(t) \overset{*}{\Rightarrow} s \text{ for some } q \text{ in } Q_d\}$, and in the bottom-up case, $M = \{\langle t, s \rangle \in T_\Sigma \times T_\Delta \mid t \overset{*}{\Rightarrow} q(s) \text{ for some } q \text{ in } Q_d\}$.
Note that we give the same name to both the transducer and the transformation defined by it. Note also that, in the derivations mentioned in the above definition of tree transformation defined by M, the intermediate results are in $T_\Delta[Q(T_\Sigma)]$ in the top-down case and $T_\Sigma[Q(T_\Delta)]$ in the bottom-up case.

The finite state tree transducer can be considered as the appropriate generalization to trees of the sequential transducer, defined in [3]. In this paper we shall restrict ourselves to tree transducers which correspond to the generalized sequential machine (see [9]). The definitions come about by restricting the left hand sides of rules to what amounts to the simplest possible form, and by restricting the right hand side of a rule to trees which contain only variables which are also contained in the left hand side of that rule.

A *finite state transformation* (abbreviated by *fst*) is either a top-down or a bottom-up finite state transformation.

A *top-down finite state transformation* (abbreviated by *t-fst*) is a top-down tree transducer $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ such that all rules in R have either the form $q(\sigma(x_1 \cdots x_k)) \to t$ with q in Q, $k \geq 1$, σ in Σ_k and $t \in T_\Delta[Q(X_k)]$, or the form $q(\sigma) \to t$ with q in Q, σ in Σ_0 and t in T_Δ.

A *bottom-up finite state transformation* (abbreviated by *b-fst*) is a bottom-up tree transducer $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ such that all rules in R have either the form $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ with $k \geq 1$, σ in Σ_k, $q_1, \cdots, q_k$, q in Q and t in $T_\Delta[X_k]$, or the form $\sigma \to q(t)$ with σ in Σ_0, q in Q and t in T_Δ.

Note that, for a *t-fst*, Q_d is the set of initial states, whereas the final states are those occurring in rules of the form $q(\sigma) \to t$. For a *b-fst*, the initial states are those occurring in rules of the form $\sigma \to q(t)$, whereas Q_d is the set of final states.

Terminology. The tree transformation defined by a top-down finite state transformation will also be called a top-down finite state transformation (and similarly for bottom-up finite state transformations). In fact, throughout this paper, we shall often make no distinction between an *fst* and the tree transformation defined by it. Hopefully this will not lead to confusion.

The class of all top-down *fst* will be denoted by *T-FST*, and the class of all bottom-up *fst* will be denoted by *B-FST*. □

Example. Consider the *t-fst* $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$, where $\Sigma_0 = \Sigma_1 = \{a\}$, $\Sigma_2 = \{\sigma\}$, $\Delta_0 = \Delta_1 = \{b, c\}$, $\Delta_3 = \{\tau\}$, $Q = \{q_0, q_b, q_c\}$, $Q_d = \{q_0\}$ and R consists of the following rules (numbered from 1 to 5):

1. $q_0(\sigma(x_1x_2)) \to b(\tau(q_b(x_2)q_c(x_1)q_b(x_1)))$,
2. $q_b(a(x_1)) \to b(q_b(x_1))$,
3. $q_b(a) \to b$,
4. $q_c(a(x_1)) \to c(q_c(x_1))$, and
5. $q_c(a) \to c$.

M translates the tree $\sigma(a(a)a)$ into the tree $b(\tau(bc(c)b(b)))$ as follows (where the numbers at each step indicate the rule applied):

$$\begin{aligned} q_0(\sigma(a(a)a)) &\underset{1}{\Rightarrow} b(\tau(q_b(a)q_c(a(a))q_b(a(a)))) \\ &\underset{3}{\Rightarrow} b(\tau(bq_c(a(a))q_b(a(a)))) \underset{4}{\Rightarrow} b(\tau(bc(q_c(a))q_b(a(a)))) \\ &\underset{2}{\Rightarrow} b(\tau(bc(q_c(a))b(q_b(a)))) \underset{5}{\Rightarrow} b(\tau(bc(c)b(q_b(a)))) \\ &\underset{3}{\Rightarrow} b(\tau(bc(c)b(b))) \end{aligned}$$

or in pictures, see diagram on page 204. □

Next we define a number of natural restrictions on *fst*.

A *t-fst* $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ is (total) *deterministic* (abbreviated by *dt-fst*) if, for all $k \geq 0$, $\sigma \in \Sigma_k$ and $q \in Q$, there is exactly one rule with left hand side $q(\sigma(x_1 \cdots x_k))$ (or $q(\sigma)$ if $k = 0$), and Q_d is a singleton. The class of all *dt-fst* will be denoted by *DT-FST*.

A *b-fst* $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ is (total) *deterministic* (abbreviated by *db-fst*) if, for all $k \geq 0$, $\sigma \in \Sigma_k$ and $q_1, \cdots, q_k \in Q$, there is exactly one rule with left hand side $\sigma(q_1(x_1) \cdots q_k(x_k))$ (or σ if $k = 0$). The class of all *db-fst* will be denoted by *DB-FST*.

Note that all deterministic top-down *fst* are total functions, whereas all deterministic bottom-up *fst* are partial (not necessarily total) functions.

A tree in which variables from X occur is said to be *linear* if no variable occurs more than once in the tree. For $k \geq 0$, a tree in which variables from X_k occur is said to be *nondeleting with respect to* X_k if each variable from X_k occurs at least once in the tree.

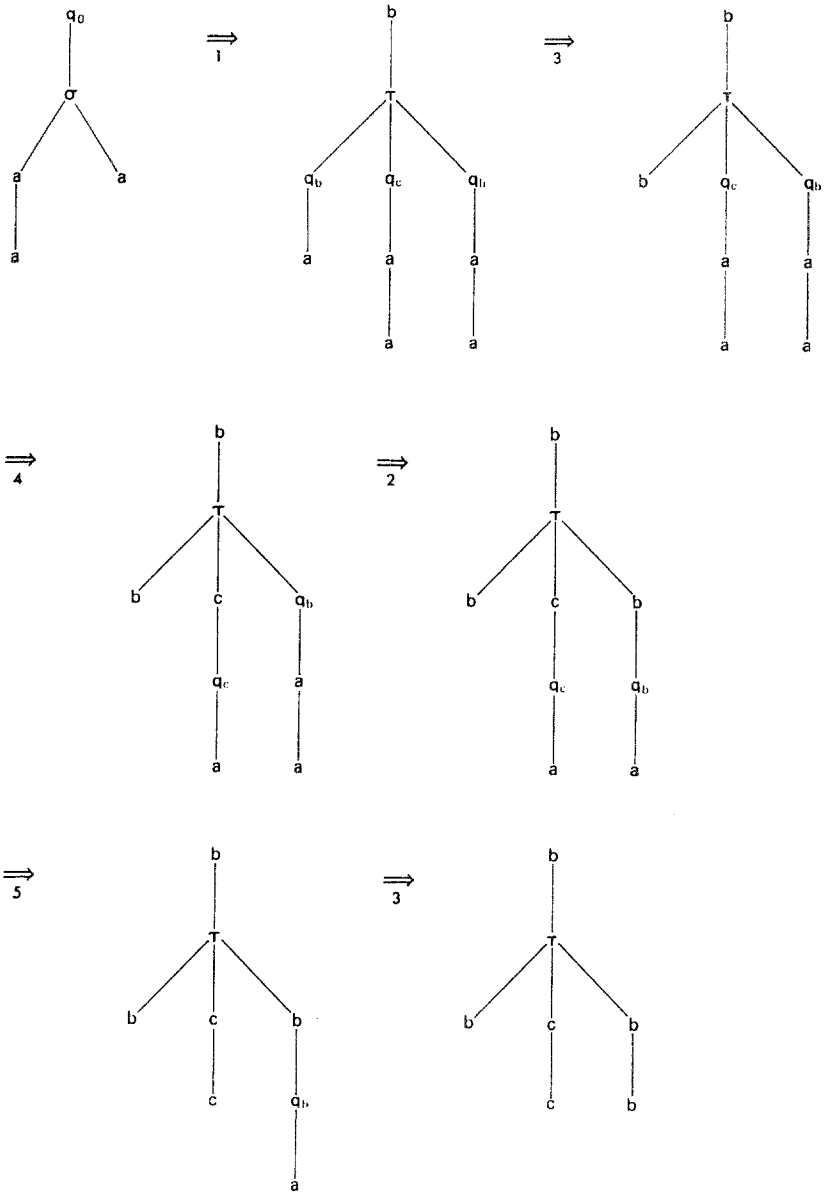
An *fst* is *linear* if all right hand sides of rules of the *fst* are linear (note that the left hand sides of rules of an *fst* are linear by definition). The phrases "linear top-down *fst*" and "linear bottom-up *fst*" will be abbreviated by "*lt-fst*" and "*lb-fst*" respectively. The classes of linear top-down and linear bottom-up *fst* will be denoted by *LT-FST* and *LB-FST* respectively.

An *fst* is *nondeleting* if for each rule of the *fst*, in the left hand side of which a symbol from Σ_k occurs ($k \geq 0$), the right hand side is nondeleting with respect to X_k.

A (nondeterministic) *finite tree automaton* (abbreviated by *fta*) is an *fst* $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ such that $\Delta = \Sigma$ (as ranked alphabets) and either (1) or (2) holds:

(1) all rules have either the form $q(\sigma(x_1 \cdots x_k)) \to \sigma(q_1(x_1) \cdots q_k(x_k))$ with $k \geq 1$, $\sigma \in \Sigma_k$ and $q_1, \cdots, q_k, q \in Q$, or the form $q(\sigma) \to \sigma$ with $q \in Q$ and $\sigma \in \Sigma_0$;

(2) all rules have either the form $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(\sigma(x_1 \cdots x_k))$ with $k \geq 1$, $\sigma \in \Sigma_k$ and $q_1, \cdots, q_k, q \in Q$, or the form $\sigma \to q(\sigma)$ with $q \in Q$ and $\sigma \in \Sigma_0$.



In case (1) the *fta* is called *top-down* and in case (2) *bottom-up*. The tree transformation defined by an *fta* is also called a finite tree automaton. The domain of a finite tree automaton is called a *recognizable* tree language. The class of all recognizable tree languages will be denoted by *RECOG*. As can easily be seen, a finite tree automaton defines a transformation which is the identity on a recognizable tree language (its domain) and is undefined elsewhere.

If U is a recognizable tree language and T is a tree transformation, then $T(U)$ is called a *surface set*. In particular, if T belongs to a class F of tree trans-

formations, then $T(U)$ is called an *F surface set* (thus we have *T-FST* surface sets, *DB-FST* surface sets, etc.).

Let us recall some well known facts concerning finite tree automata (see [2, 5, 11]). The class of tree languages recognizable by a bottom-up finite tree automaton is equal to the class of tree languages recognizable by a top-down finite tree automaton (both are equal to *RECOG*). Every recognizable tree language is the domain of a deterministic bottom-up *fta*. *RECOG* is closed under intersection and complementation.

We shall also need the notion of *substitution*. Let Σ be a ranked alphabet. For $k \geq 1$, t in $T_\Sigma[X_k]$ and $t_1, \cdots, t_k$ trees over some ranked alphabet, we define the result of substituting t_i for x_i in t, denoted by $t[t_1, \cdots, t_k]$, inductively as follows.

(1) For σ in Σ_0, $\sigma[t_1, \cdots, t_k] = \sigma$.
(2) For i in $\{1, 2, \cdots, k\}$, $x_i[t_1, \cdots, t_k] = t_i$.
(3) For $n \geq 1$, σ in Σ_n and $s_1, \cdots, s_n$ in $T_\Sigma[X_k]$, $\sigma(s_1 \cdots s_n)\ [t_1, \cdots, t_k] = \sigma(s_1[t_1, \cdots, t_k] \cdots s_n[t_1, \cdots, t_k])$.

Throughout this paper we shall only occasionally give formal proofs of theorems. Mostly the reader has to be content with an intuitive description, or a construction without proof of correctness. The reason behind this is essentially that most proofs are straightforward and very dull induction arguments (the induction being on the structure of the trees involved). Moreover, we have chosen a formalism which we hope to be intuitively clear, but which does not lend itself very well to detailed proofs (for an excellent formalism to give proofs in, see [9]). Nevertheless, we shall give (without proof!) in the following lemmas the main properties of the relation $\overset{*}{\Rightarrow}$ in bottom-up and top-down *fst*, needed in induction arguments.

Let us first indicate the role of substitution in the application of top-down and bottom-up *fst* rules. The application of a top-down rule of the form $q(\sigma(x_1 \cdots x_k)) \to t$ consists of the replacement of a subtree of the form $q(\sigma(t_1 \cdots t_k))$ by the tree $t[t_1, \cdots, t_k]$. Similarly, the application of a bottom-up rule of the form $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ consists of the replacement of a subtree of the form $\sigma(q_1(t_1) \cdots q_k(t_k))$ by the tree $q(t[t_1, \cdots, t_k])$.

We now state our lemmas.

LEMMA 1.1. *Let $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a bottom-up fst.*

(1) *For $\sigma \in \Sigma_0$, $q \in Q$ and $s \in T_\Delta$, if $\sigma \overset{*}{\Rightarrow} q(s)$, then the rule $\sigma \to q(s)$ is in R.*
(2) *For $k \geq 1$, $\sigma \in \Sigma_k$, $t_1, \cdots, t_k \in T_\Sigma$, $q \in Q$ and $s \in T_\Delta$, if $\sigma(t_1 \cdots t_k) \overset{*}{\Rightarrow} q(s)$, then there exist states $q_1, \cdots, q_k$ and trees $s_1, \cdots, s_k$ in T_Δ such that $\sigma(t_1 \cdots t_k) \overset{*}{\Rightarrow} \sigma(q_1(s_1) \cdots q_k(s_k)) \Rightarrow q(s)$ and for all i $(1 \leq i \leq k)$ $t_i \overset{*}{\Rightarrow} q_i(s_i)$.*
(3) *For $k \geq 1$, $t_1, \cdots, t_k \in T_\Sigma$, $q_1, \cdots, q_k \in Q$, $s_1, \cdots, s_k \in T_\Delta$ and $\sigma \in \Sigma_k$, if for all i $(1 \leq i \leq k)$ $t_i \overset{*}{\Rightarrow} q_i(s_i)$, then $\sigma(t_1 \cdots t_k) \overset{*}{\Rightarrow} \sigma(q_1(s_1) \cdots q_k(s_k))$.*

LEMMA 1.2. *Let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a top-down fst.*

(1) *For $q \in Q$, $\sigma \in \Sigma_0$ and $t \in T_\Delta$, if $q(\sigma) \overset{*}{\Rightarrow} s$, then the rule $q(\sigma) \to s$ is in R.*

(2) *Let $k \geq 1$ and let $t \in T_\Delta[X_k]$ be linear and nondeleting with respect to X_k. Let $q_1, \cdots, q_k$ be in Q, $t_1, \cdots, t_k$ in T_Σ and s in T_Δ. Then the following holds. If $t[q_1(t_1), \cdots, q_k(t_k)] \overset{*}{\Rightarrow} s$, then there exist trees $s_1, \cdots, s_k$ in T_Δ such that $s = t[s_1, \cdots, s_k]$ and for all i $(1 \leq i \leq k)$ $q_i(t_i) \overset{*}{\Rightarrow} s_i$. Furthermore, if T is deterministic, then the same is true without the requirement of linearity of t.*

(3) *For $k \geq 1$, $q_1, \cdots, q_k \in Q$, $t_1, \cdots, t_k \in T_\Sigma$, $s_1, \cdots, s_k \in T_\Delta$ and $t \in T_\Delta[X_k]$, if for all i $(1 \leq i \leq k)$ $q_i(t_i) \overset{*}{\Rightarrow} s_i$, then $t[q_1(t_1), \cdots, q_k(t_k)] \overset{*}{\Rightarrow} t[s_1, \cdots, s_k]$.*

With regard to Lemma 1.2(2) we note the following. If t is a tree in $T_\Delta[Q(X_k)]$ for some $k \geq 1$ (in other words, t is a possible right hand side of a top-down *fst* rule), then there exist an $n \geq 0$ (namely, n is the number of occurrences of elements from $Q(X_k)$ in t) and a tree t' in $T_\Delta[X_n]$, which is linear and nondeleting with respect to X_n, such that $t = t'[q_1(x_{i_1}), \cdots, q_n(x_{i_n})]$ for certain states $q_1, \cdots, q_n$ and certain indices $i_1, \cdots, i_n$ in $\{1, 2, \cdots, k\}$. In fact, $q_1(x_{i_1}), \cdots, q_n(x_{i_n})$ are all occurrences of elements of $Q(X_k)$ in t, and t' can be obtained from t by replacing these occurrences by $x_1, \cdots, x_n$ respectively.

2. Comparison of bottom-up and top-down fst. In this section we will be concerned with convincing the reader of the incomparability of the classes of bottom-up and top-down *fst* (Theorem 2.3). Roughly speaking, this incomparability is caused by the different order in which *b-fst* and *t-fst* construct and process copies of subtrees.

A bottom-up *fst* has the ability of making a copy of an output tree after nondeterministic processing of the input tree. Such behavior cannot be displayed by any top-down *fst*. It can however be simulated by a composition of two top-down *fst*, one for processing the input tree nondeterministically and the other for copying the resulting output tree. This implies the well known result that the class of top-down *fst* is not closed under composition (Theorem 2.4).

On the other hand a top-down *fst* has the ability of copying an input tree and treating the resulting copies differently. No single bottom-up *fst* has this ability. It can however be simulated by a composition of two bottom-up *fst*, one for copying the input tree and the other for processing the copies. This implies that the class of bottom-up *fst* is not closed under composition (Theorem 2.5).

Obviously, in the case of linear *fst* the above differences between bottom-up and top-down disappear. Nevertheless it will be shown that the class of linear top-down *fst* is properly included in the class of linear bottom-up *fst* (Theorem 2.8). Intuitively this is caused by the fact that a (linear) bottom-up *fst* has the ability to process an input tree and decide, by the state it is in, whether to delete that tree or not, whereas a (linear) top-down *fst* can only delete an input tree without inspecting it at all. Since a composition of two (linear) top-down *fst* can first process the input tree and then decide whether to delete it or not, we obtain the well known result that the class of linear top-down *fst* is not closed under composition (Theorem 2.7). It will be shown in Section 4 that the class of linear bottom-up *fst is* closed under composition.

Finally we show that in the case of linear nondeleting *fst* all differences between bottom-up and top-down disappear (Theorem 2.9).

To shorten later considerations we now give names to the above mentioned typical bottom-up and top-down capabilities.

(B1) Copying of an output tree after nondeterministic processing of the input tree.
(B2) Deciding whether to delete a tree or not after processing it.
(T) Copying of an input tree and processing the copies differently.

We first provide an example of a bottom-up *fst*, which is not a top-down *fst* because it has property (B1). (This example is a simplified version of the one in Lemma 8.7 of [9]).

Example 2.1. Consider the *b-fst* $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$, where $\Sigma_0 = \{a\}$, $\Sigma_1 = \{a, \sigma\}$, $\Delta_0 = \Delta_1 = \{a, b\}$, $\Delta_2 = \{\sigma\}$, $Q = Q_d = \{*\}$ and R consists of the rules

$$a \to *(a), \qquad a \to *(b),$$
$$a(*(x)) \to *(a(x)), \qquad a(*(x)) \to *(b(x)), \qquad \text{and}$$
$$\sigma(*(x)) \to *(\sigma(xx)).$$

Then, for instance, B may transform $\sigma(a(a(a)))$ into $\sigma(a(b(b))a(b(b)))$ as follows

$$\sigma(a(a(a))) \Rightarrow \sigma(a(a(*(b)))) \Rightarrow \sigma(a(*(b(b))))$$
$$\Rightarrow \sigma(*(a(b(b)))) \Rightarrow *(\sigma(a(b(b))a(b(b)))).$$

Intuitively it is clear that no top-down *fst* T can define the same tree transformation as B. To see this, consider in T_Σ a tree with top symbol σ and a tail of n symbols a, where $n \geq 1$, that is, a tree of the form $\sigma(a(a(\cdots a(a)\cdots)))$. Such a tree is translated by B into all trees in T_Δ of the form $\sigma(tt)$, where t is any tail of length n, labeled by a's and b's, that is, t is of the form $u_1(u_2(\cdots(u_n)\cdots))$, where $u_i \in \{a, b\}$. If T would define the same transformation, then it would have to start with copying the tail of a's and continue with relabeling the resulting tails in an arbitrary but identical way. But, since the translation of one tail is independent of that of the other, the top-down *fst* T cannot accomplish both tails to be labeled in the same way (for n sufficiently large). For a formal proof of such a statement, see Lemma 8.7 of [9].

However, the transformation B can be simulated by the composition of two top-down *fst* T_1 and T_2, where T_1 relabels the tail of a's and T_2 does the copying. In fact, let $T_1 = \langle \Sigma, \Omega, \{*\}, \{*\}, R_1 \rangle$, where $\Omega_0 = \{a, b\}$, $\Omega_1 = \{a, b, \sigma\}$ and R_1 consists of the rules

$$*(\sigma(x)) \to \sigma(*(x)),$$
$$*(a(x)) \to a(*(x)), \qquad *(a(x)) \to b(*(x)),$$
$$*(a) \to a \quad \text{and} \quad *(a) \to b,$$

and let $T_2 = \langle \Omega, \Delta, \{*\}, \{*\}, R_2 \rangle$, where R_2 consists of the rules

$$*(\sigma(x)) \to \sigma(*(x)*(x)),$$
$$*(a(x)) \to a(*(x)), \qquad *(b(x)) \to b(*(x)),$$
$$*(a) \to a \quad \text{and} \quad *(b) \to b.$$

To prove that $B = T_1 \circ T_2$ it obviously suffices to show that for all t in T_Σ and s in T_Δ,

($) $t \overset{*}{\underset{B}{\Rightarrow}} *(s)$ if and only if $*(t) \overset{*}{\underset{T_1}{\Rightarrow}} u$ and $*(u) \overset{*}{\underset{T_2}{\Rightarrow}} s$ for some u in T_Ω.

Using Lemmas 1.1 and 1.2, an easy and straightforward proof of ($) can be given by induction on t (in both the if and the only-if direction). The proof is left to the reader. □

Next we provide an example of a top-down *fst*, which is not a bottom-up *fst* because it has property (*T*).

Example 2.2. Consider the *t-fst* $T = \langle \Sigma, \Delta, \{*\}, \{*\}, R \rangle$, where Σ and Δ are the same as in Example 2.1 and R consists of the rules

$$*(\sigma(x)) \to \sigma(*(x)*(x)),$$
$$*(a(x)) \to a(*(x)), \qquad *(a(x)) \to b(*(x)),$$
$$*(a) \to a \quad \text{and} \quad *(a) \to b.$$

Then, for instance, T can transform $\sigma(a(a))$ into $\sigma(a(b)b(a))$ as follows

$$*(\sigma(a(a))) \Rightarrow \sigma(*(a(a))*(a(a))) \Rightarrow$$
$$\sigma(a(*(a))*(a(a))) \Rightarrow \sigma(a(b)*(a(a))) \Rightarrow$$
$$\sigma(a(b)b(*(a))) \Rightarrow \sigma(a(b)b(a)).$$

Intuitively it is clear that no bottom-up *fst* B can define the same tree transformation as T. To see this, consider, as in Example 2.1, a tree with top symbol σ and a tail of n symbols a, where $n \geq 1$. This tree is translated by T into all trees of the form $\sigma(t_1 t_2)$, where t_1 and t_2 are tails of length n, arbitrary labeled by a's and b's (and not necessarily equal). Obviously, with the same tree as input, B cannot produce all the output trees, since (for sufficiently large n) B can only double the tail if it has arrived at the top, and then the resulting tails have to be (partly) the same. A formal proof that T is not a *b-fst* is left to the reader.

However, the transformation T can easily be defined by the composition of two bottom-up *fst* B_1 and B_2, where B_1 does the copying and B_2 the relabeling. In fact, let $B_1 = \langle \Sigma, \Omega, \{*\}, \{*\}, R_1 \rangle$, where $\Omega_0 = \Omega_1 = \{a\}$, $\Omega_2 = \{\sigma\}$ and R_1 consists of the rules

$$a \to *(a), \quad a(*(x)) \to *(a(x)) \quad \text{and} \quad \sigma(*(x)) \to *(\sigma(xx)),$$

and let $B_2 = \langle \Omega, \Delta, \{*\}, \{*\}, R_2 \rangle$ where R_2 consists of the rules

$$a \to *(a), \qquad a \to *(b),$$
$$a(*(x)) \to *(a(x)), \qquad a(*(x)) \to *(b(x)) \quad \text{and}$$
$$\sigma(*(x)*(y)) \to *(\sigma(xy)).$$

It is left to the reader to show that $T = B_1 \circ B_2$. □

The following three theorems are immediate consequences of Examples 2.1 and 2.2.

THEOREM 2.3. *The classes of tree transformations B-FST and T-FST are incomparable.*

Proof. The bottom-up *fst B* of Example 2.1 does not belong to *T-FST*, and the top-down *fst T* of Example 2.2 does not belong to *B-FST*. □

THEOREM 2.4. ([8, 7]). *T-FST is not closed under composition.*

Proof. In Example 2.1, $T_1 \circ T_2$ is not in *T-FST*. □

THEOREM 2.5. *B-FST is not closed under composition.*

Proof. In Example 2.2, $B_1 \circ B_2$ is not in *B-FST*. □

Note the strong "duality" between top-down and bottom-up in the above examples and theorems.

So far we have seen that the difference between top-down and bottom-up *fst* came about by their different ways of copying. We now consider the case that copying is not allowed, that is, the case of linear *fst*.

First we give an example of a linear bottom-up *fst* which is not a top-down *fst* (linear or nonlinear), because it has property (*B*2). The example is adapted from one of W. Ogden in §4 of [7].

Example 2.6. Consider the linear bottom *fst* $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$, where $\Sigma_0 = \Sigma_1 = \{a, b\}$, $\Sigma_2 = \{\sigma\}$, $\Delta_0 = \{a, b\}$, $\Delta_1 = \{a, b, \sigma\}$, $Q = Q_d = \{*\}$ and R consists of the rules

$$b \to *(b), \quad a(*(x)) \to *(a(x)), \quad \text{and} \quad \sigma(*(x)*(y)) \to *(\sigma(x)).$$

Intuitively it is clear that no top-down *fst T*, whether it is linear or not, can define the same tree transformation as *B*. To see this, consider an arbitrary tree of the form $\sigma(t_1 t_2)$, where t_1 and t_2 do not contain σ. Obviously, this tree is in the domain of *B* if and only if t_1 and t_2 both are tails of symbols a with a symbol b at the end, and if that is the case, *B* translates $\sigma(t_1 t_2)$ into $\sigma(t_1)$. Clearly, *T* cannot both check that t_2 has the required property and delete it. A formal proof is left to the reader.

Notice that, in *B*, the property (*B*2) is present as a rather pathological case of the general property (*B*2) described before: *B* decides whether to delete t_2 or be undefined for $\sigma(t_1 t_2)$ by respectively being defined or undefined for t_1 and t_2. It is easy to give less pathological examples of bottom-up *fst* which are not top-down *fst* because of property (*B*2), however we wished to show by the present example that even one-state *b-fst* may have this property.

Similar to the situation in the previous example, the translation of *B* can be simulated by two linear top-down *fst* T_1 and T_2, where T_1 checks that *b*'s only occur at the bottom of the tree and T_2 does the deletion. In fact, let $T_1 = \langle \Sigma, \Sigma, \{*\}, \{*\}, R_1 \rangle$, where R_1 consists of the rules

$$*(\sigma(xy)) \to \sigma(*(x)*(y)), \quad *(a(x)) \to a(*(x)), \quad \text{and} \quad *(b) \to b,$$

and let $T_2 = \langle \Sigma, \Delta, \{*\}, \{*\}, R_2 \rangle$, where R_2 consists of the rules

$$*(\sigma(xy)) \to \sigma(*(x)),$$
$$*(a(x)) \to a(*(x)), \qquad *(b(x)) \to b(*(x)),$$
$$*(a) \to a \quad \text{and} \quad *(b) \to b.$$

(Obviously two of the rules of T_2 are superfluous. They were added so as to make T_2 a *dt-fst*.) Again, it is left to the reader to show that $B = T_1 \circ T_2$. □

Since, in the above example, $T_1 \circ T_2$ is not a *t-fst*, we obtain the following well known theorem.

THEOREM 2.7 ([7, 9]). *LT-FST is not closed under composition.*

In the next theorem we prove that the class of linear top-down *fst* is included in the class of linear bottom-up *fst*.

THEOREM 2.8. *LT-FST* $\subset$ *LB-FST.*

Proof. If containment is proved, then proper containment follows from Example 2.6. The proof of containment now follows.

Let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be an arbitrary linear *t-fst*. We have to show the existence of a linear *b-fst* B defining the same transformation as T. Let $B = \langle \Sigma, \Delta, Q \cup \{e\}, Q_d, R_B \rangle$, where e is a new symbol (standing for "erase"), and R_B is constructed according to the following three requirements. (1) Let δ_0 be an arbitrary fixed element of Δ_0. For each $k \geq 1$ and σ in Σ_k the rule $\sigma(e(x_1)e(x_2)\cdots e(x_k)) \to e(\delta_0)$ is in R_B, and for each σ in Σ_0 the rule $\sigma \to e(\delta_0)$ is in R_B. (2) For each q in Q, σ in Σ_0 and r in T_Δ, if $q(\sigma) \to r$ is in R, then $\sigma \to q(r)$ is in R_B. (3) Let $q(\sigma(x_1 \cdots x_k)) \to r$ be in R, where $k \geq 1$, $q \in Q$, $\sigma \in \Sigma_k$ and r is a linear tree in $T_\Delta[Q(X_k)]$. Let, for each i in $\{1, \cdots, k\}$, the state q_i in $Q \cup \{e\}$ be defined by

$$q_i = \begin{cases} p & \text{if } p(x_i) \text{ occurs in } r \\ e & \text{otherwise (that is, } x_i \text{ does not occur in } r). \end{cases}$$

(Note that this is a proper definition: since r is linear, no x_i occurs more than once in r). Then the rule $\sigma(q_1(x_1)\cdots q_k(x_k)) \to q(r')$ is in R_B, where r' is the result of replacing any string of the form $p(x_i)$ by x_i in r (thus r' is linear and $r = r'[q_1(x_1), \cdots, q_k(x_k)]$).

This ends the construction of B. Intuitively, B treats the input in the same way as T but in the reverse order. Moreover, whenever T deletes a subtree, B should make a translation of that subtree before deletion. (It does not matter which translation is made of the subtree. We have defined B in such a way that it can translate it into δ_0). To show formally that $B = T$ it obviously suffices to prove that for every q in Q, t in T_Σ and s in T_Δ,

($) $$q(t) \overset{*}{\underset{T}{\Rightarrow}} s \quad \text{if and only if} \quad t \overset{*}{\underset{B}{\Rightarrow}} q(s).$$

We shall prove this by induction on t. Firstly, let us assume that $t = \sigma$, where $\sigma \in \Sigma_0$. Then the truth of ($) is clear from Lemmas 1.1(1) and 1.2(1) and point (2) in the construction of R_B. Secondly, let us assume that $t = \sigma(t_1 \cdots t_k)$, where $k \geq 1$, $\sigma \in \Sigma_k$ and $t_1, \cdots, t_k$ are elements of T_Σ for which ($) is true.

The only-if part of ($) is proved as follows. Assume that $q(\sigma(t_1 \cdots t_k)) \overset{*}{\underset{T}{\Rightarrow}} s$. Suppose that the first step of this derivation results from the application of the rule $q(\sigma(x_1 \cdots x_k)) \to r$, that is, we have $q(\sigma(t_1 \cdots t_k)) \underset{T}{\Rightarrow} r[t_1, \cdots, t_k] \overset{*}{\underset{T}{\Rightarrow}} s$. Using

the definition of r' and q_i from (3) above, $r[t_1, \cdots, t_k] = r'[q_1(t_1), \cdots, q_k(t_k)]$ and so $r'[q_1(t_1), \cdots, q_k(t_k)] \overset{*}{\underset{T}{\Rightarrow}} s$. It now follows from an appropriate application of Lemma 1.2(2) that there are $s_1, \cdots, s_k$ in T_Δ such that $s = r'[s_1, \cdots, s_k]$ and, for all i in $\{1, \cdots, k\}$, $q_i(t_i) \overset{*}{\underset{T}{\Rightarrow}} s_i$ if x_i occurs in r' (or equivalently, in r), and $s_i = \delta_0$ otherwise. Now, if x_i occurs in r, then, by induction, $t_i \overset{*}{\underset{B}{\Rightarrow}} q_i(s_i)$. Otherwise $q_i = e$ and so, by (1) above, $t_i \overset{*}{\underset{B}{\Rightarrow}} e(\delta_0)$. Hence $t_i \overset{*}{\Rightarrow} q_i(s_i)$ for all i, $1 \leq i \leq k$. Consequently, since the rule $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(r')$ is in R_B, $\sigma(t_1 \cdots t_k) \overset{*}{\underset{B}{\Rightarrow}} \sigma(q_1(s_1) \cdots q_k(s_k)) \underset{B}{\Rightarrow} q(r'[s_1, \cdots, s_k]) = q(s)$, where we have used Lemma 1.1(3).

The if part of (\$) is proved as follows. Assume that $\sigma(t_1 \cdots t_k) \overset{*}{\underset{B}{\Rightarrow}} q(s)$. Suppose that the last step of the derivation results from application of the rule $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(r')$, obtained as described in (3) above (recall that $q \neq e$). By Lemma 1.1(2), there exist $s_1, \cdots, s_k$ in T_Δ such that $\sigma(t_1 \cdots t_k) \overset{*}{\underset{B}{\Rightarrow}} \sigma(q_1(s_1) \cdots q_k(s_k)) \underset{B}{\Rightarrow} q(s)$, $s = r'[s_1, \cdots, s_k]$ and, for all i in $\{1, \cdots, k\}$, $t_i \overset{*}{\underset{B}{\Rightarrow}} q_i(s_i)$. Now, if x_i occurs in r', then $q_i \neq e$ and so, by induction, $q_i(t_i) \overset{*}{\underset{T}{\Rightarrow}} s_i$. Consequently $q(\sigma(t_1 \cdots t_k)) \underset{T}{\Rightarrow} r[t_1, \cdots, t_k] = r'[q_1(t_1), \cdots, q_k(t_k)]$ and, using Lemma 1.2(3), $r'[q_1(t_1), \cdots, q_k(t_k)] \overset{*}{\underset{T}{\Rightarrow}} r'[s_1, \cdots, s_k] = s$, since in the latter derivation only states different from e are involved. This proves the if part of (\$) and the theorem. □

In the linear case, properties ($B1$) and (T) are not present and only property ($B2$) remains, making linear bottom-up stronger than linear top-down. If neither copying nor deleting is allowed, then property ($B2$) also disappears and top-down and bottom-up are equally strong.

THEOREM 2.9. *The class of nondeleting linear bottom-up fst is identical to the class of nondeleting linear top-down fst.*

Proof. To show that each nondeleting *lt-fst* is a nondeleting *lb-fst*, consider the proof of Theorem 2.8. Since we now may assume that the *lt-fst* T of that proof is nondeleting, it obviously suffices to erase all rules involving e and δ_0 from R_B to obtain a nondeleting *lb-fst* defining the same transformation as T.

Now suppose that $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ is an arbitrary nondeleting *lb-fst*. Let T be the *t-fst* $\langle \Sigma, \Delta, Q, Q_d, R_T \rangle$ where R_T is defined by the following two requirements.

(1) For each σ in Σ_0, q in Q and t in T_Δ, if $\sigma \to q(t)$ is in R, then $q(\sigma) \to t$ is in R_T.

(2) If $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ is in R, where $k \geq 1$, $\sigma \in \Sigma_k$, $q_1, \cdots, q_k$, $q \in Q$ and $t \in T_\Delta$, then $q(\sigma(x_1 \cdots x_k)) \to t[q_1(x_1), \cdots, q_k(x_k)]$ is in R_T.

Obviously T is nondeleting and linear. Now observe that the nondeleting *lb-fst* corresponding to T according to the proof of Theorem 2.8, modified as indicated above, is precisely B. Hence, by that proof, $T = B$. □

3. Decomposition of bottom-up and top-down fst. In this section we show that each *fst* (bottom-up or top-down) can be decomposed into a number of

very simple *fst*. The simple *fst* occurring in these decompositions are independent of the bottom-up/top-down distinction. In fact they are either finite tree automata or relabelings or "homomorphisms". Obviously such decompositions are very useful when proving properties of *fst*: if the property is preserved by composition, then the proof of it splits up into a number of simple cases.

Let us first define our "simple" *fst* and discuss some of their properties.

Definition 3.1. (1) We shall denote by *FTA* the class of all finite tree automata. (In other words, *FTA* consists of all *fst* that are the identity on a recognizable tree language).

(2) We shall denote by *RELAB* the class of all "relabelings". A *relabeling* is an *fst* $\langle \Sigma, \Delta, Q, Q_d, R\rangle$ such that Q is a singleton (say $\{*\}$), $Q_d = Q$ and (in the top-down case) all the rules are of the form $*(\sigma(x_1 \cdots x_k)) \to \tau(*(x_1)\cdots *(x_k))$ with $k \geq 1$, $\sigma \in \Sigma_k$ and $\tau \in \Delta_k$ or of the form $*(\sigma) \to \tau$ with σ in Σ_0 and $\tau \in \Delta_0$, or (in the bottom-up case) all rules are of the form $\sigma(*(x_1)\cdots *(x_k)) \to *(\tau(x_1 \cdots x_k))$ with $k \geq 1$, $\sigma \in \Sigma_k$ and $\tau \in \Delta_k$ or of the form $\sigma \to *(\tau)$ with $\sigma \in \Sigma_0$ and $\tau \in \Delta_0$.

(3) We shall denote by *HOM* the class of all one-state deterministic *fst* (with $Q_d = Q$). The elements of *HOM* will be called *homomorphisms* (see p. 355 of [9]). Furthermore we shall denote by *LHOM* the class of all linear homomorphisms.

Examples of such "simple" *fst* can be found in the previous examples. In Example 2.1, T_1 is in *RELAB* and T_2 in *HOM*; in Example 2.2, B_1 is in *HOM* and B_2 in *RELAB*; and in Example 2.6, T_1 is in both *FTA* and *RELAB*, and T_2 is in *LHOM*. Note also that, for every ranked alphabet Σ, the identity function on T_Σ is an element of *FTA*, *RELAB* and *(L)HOM*.

According to Definition 3.1 the classes *FTA*, *RELAB*, *HOM* and *LHOM* contain both bottom-up and top-down *fst*. It is, however, easy to see that, with regard to the tree transformations defined, we may restrict the definition to bottom-up *fst* only or to top-down *fst* only. This is expressed in the following lemma.

LEMMA 3.2. *The classes of transformations FTA, RELAB, HOM and LHOM are independent of whether they are defined by bottom-up fst or by top-down fst.*

Proof. For *FTA* and *RELAB* the lemma follows directly from the proof of Theorem 2.9. (Note that, for *FTA*, the lemma is just a restatement of the well known equivalence between top-down and bottom-up finite tree automata).

For *HOM*, we first observe that in both the bottom-up and the top-down case the requirement of determinism amounts to the fact that for each σ in Σ_k ($k \geq 1$) there is exactly one rule in the left hand side of which σ occurs. The easy proof that the bottom-up homomorphisms are equal to the top-down homomorphisms (and similarly for the linear case) is left to the reader (he should change a bottom-up rule $\sigma(*(x_1)\cdots *(x_k)) \to *(t)$ into the top-down rule $*(\sigma(x_1 \cdots x_k)) \to t[*(x_1), \cdots, *(x_k)]$ and vice versa). □

Thus, by the above lemma, *FTA*, *RELAB* and *LHOM* are contained in *LB-FST* ∩ *LT-FST*, *HOM* is contained in *DB-FST* ∩ *DT-FST*, and *RELAB*

and *HOM* consist of one-state *fst* only. Note also that the elements of *HOM* are total functions, and that *FTA* $\subseteq$ *DB-FST* (see Section 1).

It is easy to think of representations of relabelings and homomorphisms that are more reminiscent of the "string case". This is done in the next lemma the proof of which is left to the reader.

LEMMA 3.3. *Let* Σ *and* Δ *be ranked alphabets.*

(1) Each family of relations $l_k \subseteq \Sigma_k \times \Delta_k$ $(k \geq 0)$ determines an *fst* L in *RELAB* by the following requirements.

(i) For each σ in Σ_0, $L(\sigma) = \{\tau \in \Delta_0 | \langle \sigma, \tau \rangle \in l_0\}$.

(ii) For all $k \geq 1$, σ in Σ_k and $t_1, \cdots, t_k$ in T_Σ, $L(\sigma(t_1 \cdots t_k)) = \{\tau(s_1 \cdots s_k) | \langle \sigma, \tau \rangle \in l_k$ and $s_1, \cdots, s_k \in L(t_i)\}$.

Conversely, each *fst* in *RELAB* is determined by a family of relations as above.

(2) Each family of total functions h_k: $\Sigma_k \to T_\Delta[X_k]$ $(k \geq 0)$ determines an *fst* H in *HOM* by the following requirements.

(i) For each σ in Σ_0, $H(\sigma) = h_0(\sigma)$.

(ii) For all $k \geq 1$, σ in Σ_k and $t_1, \cdots, t_k$ in T_Σ,

$$H(\sigma(t_1 \cdots t_k)) = h_k(\sigma)\,[H(t_1), \cdots, H(t_k)].$$

Conversely, each *fst* in *HOM* is determined by a family of mappings as above.

Moreover, the *fst* H is linear if and only if, for each σ in Σ_k, $h_k(\sigma)$ is a linear tree. □

We state in passing the following useful lemma.

LEMMA 3.4. *Each of the classes of tree transformations FTA, RELAB, HOM and LHOM is closed under composition.*

Proof. For *FTA* the lemma is just a restatement of the well known fact that the recognizable tree languages are closed under intersection. For *RELAB*, *HOM* and *LHOM* the lemma easily follows from the previous lemma (compose the involved families of relations, or functions respectively, in the obvious way). □

We now proceed to the decomposition of *fst* into finite tree automata, relabelings and homomorphisms. First it is shown that each (linear) bottom-up *fst* can be decomposed into a relabeling, followed by a finite tree automaton, followed by a (linear) homomorphism (Theorem 3.5). Then it is proved that each top-down *fst* is the composition of a homomorphism and a linear top-down *fst*, and vice versa (Theorem 3.7). It follows that each top-down *fst* can be decomposed into a homomorphism, followed by a relabeling, followed by a finite tree automaton, followed by a linear homomorphism (Theorem 3.9). Roughly speaking, these results imply that both *B-FST* and *T-FST* are contained in *HOM* $\circ$ *RELAB* $\circ$ *FTA* $\circ$ *HOM*. Moreover they imply some interesting properties of bottom-up *fst* and their relation to top-down *fst* (Corollaries 3.10–3.13). The section will be concluded by an alternative decomposition of bottom-up *fst* in which the relabeling and *fta* of the decomposition of Theorem 3.5 are taken together into a "finite state relabeling" (Theorem 3.15).

In the next theorem we decompose bottom-up *fst*.

THEOREM 3.5.

(1) *B-FST* $\subseteq$ *RELAB* $\circ$ *FTA* $\circ$ *HOM*,

(2) *LB-FST* $\subseteq$ *RELAB* $\circ$ *FTA* $\circ$ *LHOM*.

Proof. Let $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be an arbitrary *b-fst*. We shall indicate how to decompose B into a relabeling, a finite tree automaton and a homomorphism. First we need an intermediate alphabet. Suppose that R contains m rules, numbered from 1 to m in some arbitrary but fixed manner. Let Ω be the ranked alphabet $\{1, 2, \cdots, m\}$ such that, for all i in Ω and all $k \geq 0$, $i \in \Omega_k$ if the left hand side of the ith rule of R starts with a symbol from Σ_k. Now let $\sigma(q_1(x_1)\cdots q_k(x_k)) \to q(t)$ and $\tau \to p(r)$ denote arbitrary rules of B, and suppose they are the ith and jth element of R respectively. We define three *fst* B_1, B_2 and B_3 such that $B = B_1 \circ B_2 \circ B_3$, as follows.

B_1 is the bottom-up *fst* $\langle \Sigma, \Omega, \{*\}, \{*\}, R_1 \rangle$, where R_1 contains the rules $\sigma(*(x_1)\cdots *(x_k)) \to *(i(x_1 \cdots x_k))$ and $\tau \to *(j)$ corresponding to the ith and jth element of R respectively. Obviously B_1 is an element of *RELAB*.

B_2 is the bottom-up *fst* $\langle \Omega, \Omega, Q, Q_d, R_2 \rangle$, where R_2 contains the rules $i(q_1(x_1)\cdots q_k(x_k)) \to q(i(x_1 \cdots x_k))$ and $j \to p(j)$ corresponding to the ith and jth element of R respectively. Obviously B_2 is a finite tree automaton.

Finally, B_3 is the homomorphism determined (as explained in Lemma 3.3(2)) by the family of total functions h_k: $\Omega_k \to T_\Delta[X_k]$ $(k \geq 0)$, such that, corresponding to the ith and jth element of R respectively, $h_k(i) = t$ and $h_0(j) = r$. Obviously, if B is linear, then B_3 is in *LHOM*.

It can be understood intuitively that $B = B_1 \circ B_2 \circ B_3$ as follows. Given some input tree from T_Σ, the relabeling B_1 replaces each symbol in the tree by the number of any rule of B which is applicable to that symbol. The finite tree automaton B_2 then checks whether the resulting labeling by rule numbers is consistent with respect to the required state-transitions of B. Finally, the homomorphism B_3 replaces each rule number by the partial tree specified by the right hand side of that rule.

Formally it can be proved that for all t in T_Σ, s in T_Δ and q in Q, $t \overset{*}{\underset{B}{\Rightarrow}} q(s)$ if and only if there exists u in T_Ω such that $t \overset{*}{\underset{B_1}{\Rightarrow}} (u)$, $u \overset{*}{\underset{B_2}{\Rightarrow}} q(u)$ and $B_3(u) = s$. The proof, which is by an easy induction on t (in both directions) and uses Lemmas 1.1 and 1.2, is left to the reader. □

It will be shown later (see Theorem 4.5) that the inclusions of Theorem 3.5 are actually equalities.

We now proceed to the decomposition of top-down *fst*. The next result is stated as a lemma, since it will be followed by a stronger theorem.

LEMMA 3.6. *T-FST* $\subseteq$ *HOM* $\circ$ *LT-FST*.

Proof. Let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be an arbitrary *t-fst*. The idea behind the decomposition of T is to use a homomorphism T_1 to produce in advance as many copies of subtrees of the input tree as T may need, and then simulate T by a linear *t-fst* T_2.

To define T_1, let, for x in X and r in R, r_x be the number of occurrences of x in the right hand side of rule r, and let $n = \max \{r_x | x \in X, r \in R\}$. Thus n is the maximal number of copies of subtrees needed by T in one step. Furthermore,

let Σ' be the ranked alphabet such that $\Sigma'_{nk} = \Sigma_k$ for $k \geq 0$, and $\Sigma'_i = \varnothing$ for all i which are not a multiple of n. Thus, we have multiplied the rank of each symbol of Σ by n. We now define T_1 to be the homomorphism determined (as described in Lemma 3.3(2)) by the total functions $h_k: \Sigma_k \to T_{\Sigma'}[X_k]$ with $h_0(\sigma) = \sigma$ for σ in Σ_0 and $h_k(\sigma) = \sigma(x_1^n x_2^n \cdots x_k^n)$ for σ in Σ_k, $k \geq 1$. Thus, for σ in Σ_0, $T_1(\sigma) = \sigma$ and, for σ in Σ_k, $T_1(\sigma(t_1 \cdots t_k)) = \sigma(T(t_1)^n \cdots T(t_k)^n)$.

Next we define $T_2 = \langle \Sigma', \Delta, Q, Q_d, R' \rangle$, where R' is obtained as follows.

(1) If $q(\sigma) \to t$ is in R, then it is also in R'.

(2) If $q(\sigma(x_1 \cdots x_k)) \to t$ is in R (with $t \in T_\Delta[X_k]$), then the rule $q(\sigma(x_1 x_2 \cdots x_{nk})) \to t'$ is in R, where t' is the result of substituting $x_{(i-1)n+j}$ for the jth occurrence of x_i in t, if that occurrence exists, counted from left to right ($1 \leq i \leq k$, $1 \leq j \leq n$).

Clearly T_2 is linear. Obviously T_2 imitates T, except that the necessary copies of subtrees are already there as the output of T_1, so that T_2 can work linearly. Note that T_2 deletes the copies which would not be needed by T. A formal proof that $T = T_1 \circ T_2$ is left to the reader. □

We immediately obtain the following theorem.

THEOREM 3.7. *T-FST = HOM ∘ LT-FST.*

Proof. Immediate from Lemma 3.6 and the fact, proved by Thatcher [9], that *DT-FST ∘ T-FST ⊆ T-FST.* □

It now follows that *T-FST* is properly included in *HOM ∘ LB-FST.*

LEMMA 3.8. *T-FST ⊂ HOM ∘ LB-FST.*

Proof. Inclusion follows from Theorems 2.8 and 3.7. Proper inclusion follows from Example 2.6: the linear bottom-up *fst B* of that example belongs to *HOM ∘ LB-FST* (since the identity transformation is in *HOM*), but not to *T-FST.* □

A characterization of the class *HOM ∘ LB-FST* will be given in Theorem 5.15.

In the next theorem we finally show the decomposition of top-down *fst* into simple *fst.*

THEOREM 3.9. *T-FST ⊂ HOM ∘ RELAB ∘ FTA ∘ LHOM.*

Proof. Immediate from Lemma 3.8 and Theorem 3.5(2). □

Theorems 3.5 and 3.9 together imply that

$$B\text{-}FST \cup T\text{-}FST \subseteq HOM \circ RELAB \circ FTA \circ HOM.$$

A characterization of the class *HOM ∘ RELAB ∘ FTA ∘ HOM* will be given in Theorem 5.10.

We now state a number of consequences of our decomposition results.

COROLLARY 3.10. *Let $\mathscr{L}$ be a class of tree languages. Then the following three statements are equivalent.*

(1) *$\mathscr{L}$ is closed under (linear) b-fst,*

(2) *$\mathscr{L}$ is closed under (linear) t-fst, and*

(3) *$\mathscr{L}$ is closed under fta, relabelings and (linear) homomorphisms.*

Proof. Immediate from Lemma 3.2 and Theorems 2.8, 3.5 and 3.9. □

Note that a class of tree languages is closed under *fta* if and only if it is closed under intersection with a recognizable set.

The above corollary is a generalization of the following "string theorem": a class of languages is closed under *gsm*-mappings if and only if it is closed under intersection with a regular set and under finite substitution (see Lemma 9.3 of [4]). In fact, the effort to obtain a generalization of this string theorem led to our investigation into decompositions of *fst*.

As a particular case of Corollary 3.10 we obtain the next one.

COROLLARY 3.11. *The class of all recognizable tree languages is closed under linear bottom-up fst.*

Proof. Follows directly from Corollary 3.10 and the fact that the recognizable tree languages are closed under linear top-down *fst* (see [8]). □

Next we consider domains and surface sets. Just as top-down *fst*, bottom-up *fst* have recognizable domains.

COROLLARY 3.12. *The domain of a bottom-up fst is recognizable.*

Proof. Let B be an arbitrary *b-fst*. By Theorem 3.5, there are B_1 in *RELAB*, B_2 in *FTA* and B_3 in *HOM* such that $B = B_1 \circ B_2 \circ B_3$. Since B_3 is a total function, dom $(B) = B_1^{-1}$ (dom (B_2)). By definition, dom (B_2) is a recognizable tree language. Also, the inverse of a relabeling is again a relabeling (just invert the relations l_k which determine the relabeling according to Lemma 3.3(1)).

Consequently, since *RECOG* is closed under relabelings (see Corollary 3.11), B_1^{-1} (dom (B_2)) is recognizable. □

COROLLARY 3.13. *The class of B-FST surface sets is equal to the class of HOM surface sets, which is a proper subset of the class of DT-FST surface sets.*

Proof. To show that each *B-FST* surface set is a *HOM* surface set, let B be in *B-FST* and R in *RECOG*. By Theorem 3.5(1), B has a decomposition $B_1 \circ B_2 \circ B_3$, such that $B_1 \in RELAB$, $B_2 \in FTA$ and $B_3 \in HOM$. Hence $B(R) = B_3(B_2(B_1(R)))$, Since B_1 and B_2 are linear *fst*, $B_2(B_1(R))$ is recognizable (recall Corollary 3.11). Consequently $B_3(B_2(B_1(R)))$ is a *HOM* surface set.

Conversely it is clear that each *HOM* surface set is both a *(D)B-FST* surface set and a *DT-FST* surface set.

Let us now exhibit a *DT-FST* surface set which is not a *HOM* surface set. Consider the ranked alphabet Σ such that $\Sigma_0 = \{b\}$, $\Sigma_1 = \{a, b, \sigma\}$ and $\Sigma_2 = \{\sigma\}$, and let U be the tree language in T_Σ consisting of all trees of the form $\sigma(t_1 t_2)$ such that $t_1 = a(a(a(\cdots a(b)\cdots)))$, $t_2 = a(a(a(\cdots a(b(b))\cdots)))$ and the number of a's in t_1 is equal to the number of a's in t_2. Obviously U is a *DT-FST* surface set: consider the recognizable set of all trees of the form $\sigma(a(a(a(\cdots a(b(b))\cdots))))$ and translate such a tree into a tree of U by copying the tail of a's and b's, and deleting one of the b's in the left copy. Now suppose that U is a *HOM* surface set, that is, $U = H(R)$ for some H in *HOM* and R in *RECOG*. It follows that H has to be linear, because nonlinearity of H would imply the existence of two equal subtrees in some tree of U, which is clearly impossible. But, by Corollary 3.11, this would mean that U is recognizable, which it is obviously not. □

Thus, with regard to surface sets, *B-FST* is weaker than *T-FST*, in spite of properties (*B*1) and (*B*2).

The rest of this section is concerned with an alternative decomposition of bottom-up *fst*, such that each deterministic bottom up *fst* decomposes into deterministic components. For this we need a new class of (slightly less) simple *fst*, which generalize the sequential machine mappings between strings.

Definition 3.14. We shall denote by *QRELAB* the class of all "finite state relabelings". A *finite state relabeling* is a bottom-up *fst* $\langle \Sigma, \Delta, Q, Q_d, R \rangle$ in which each rule is of one of the forms $\sigma \to q(\tau)$ with σ in Σ_0, q in Q and τ in Δ_0, or $\sigma(q_1(x_1)\cdots q_k(x_k)) \to q(\tau(x_1\cdots x_k))$ with $k \geq 1$, σ in Σ_k, $q_1, \cdots, q_k$ in Q, q in Q and τ in Δ_k. Furthermore we shall denote by *DQRELAB* the class of all deterministic finite state relabelings.

Note that one can easily define the top-down version of *QRELAB* and show that both versions give rise to the same class of tree transformations. This is no longer true for *DQRELAB*.

In the next theorem we decompose each bottom-up *fst* into a finite state relabeling followed by a homomorphism.

THEOREM 3.15.

(1) $B\text{-}FST \subseteq QRELAB \circ HOM$,

(2) $LB\text{-}FST \subseteq QRELAB \circ LHOM$,

(3) $DB\text{-}FST \subseteq DQRELAB \circ HOM$.

Proof. The proof is essentially the same as the one of Theorem 3.5. That proof can be copied up till the sentence "We define three *fst* B_1, B_2 and B_3 . . . .". From there on the proof should read as follows.

We define two *fst* B_1' and B_3' such that $B = B_1' \circ B_3'$, as follows. B_1' is the bottom-up *fst* $\langle \Sigma, \Omega, Q, Q_d, R_1' \rangle$, where R_1' contains the rules $\sigma(q_1(x_1)\cdots q_k(x_k)) \to q(i(x_1\cdots x_k))$ and $\tau \to p(j)$ corresponding to the ith and jth element of R respectively. Obviously, B_1' is a finite state relabeling, and if B is deterministic, then so is B_1'. B_3' is equal to the homomorphism B_3 of the proof of Theorem 3.5. It should be intuitively clear that $B_1' = B_1 \circ B_2$, where B_1 and B_2 are defined in the proof of Theorem 3.5. □

It will be shown later (see Theorems 4.5 and 4.6) that the inclusions of Theorem 3.15 are even equalities.

4. Composition of bottom-up fst. In Section 2 we have argued that the ability of an *fst* to copy an input tree and treat these copies in a different way is a typical top-down property, which we called (*T*). In Section 3 we have actually decomposed each top-down *fst* into two bottom-up *fst*, one for the copying and the other for treating the copies differently (Lemma 3.8). This gives us the intuitive feeling that the nonclosure of *B-FST* under composition is caused by those compositions $B_1 \circ B_2$ in which B_1 is nonlinear and B_2 is nondeterministic. In this section we strengthen this feeling by proving that, if either B_1 is linear or B_2 is deterministic, then $B_1 \circ B_2$ is in *B-FST* (Theorems 4.5 and 4.6 respectively). Moreover we prove that both *LB-FST* and *DB-FST* are closed under composition (also in Theorems 4.5 and 4.6 respectively).

To prove the above closure results it obviously suffices to consider a number

of simple cases, using the decomposition results of the previous section. In Lemmas 4.1, 4.2 and 4.4 some of these cases are considered, involving homomorphisms, finite tree automata and relabelings respectively.

We first consider compositions involving homomorphisms.

LEMMA 4.1.

(1) *B-FST* $\circ$ *HOM* $\subseteq$ *B-FST*,

(2) *LB-FST* $\circ$ *LHOM* $\subseteq$ *LB-FST*,

(3) *DB-FST* $\circ$ *HOM* $\subseteq$ *DB-FST*.

Proof. To prove (1), let $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be an arbitrary *b-fst* and let H be an arbitrary homomorphism determined (as described in Lemma 3.3(2)) by functions $h_k: \Delta_k \to T_\Omega[X_k]$, $k \geq 0$, where Ω is a ranked alphabet. We extend H by defining $H(x_i) = x_i$ for all x_i in X. Thus H is defined for each tree in $T_\Delta[X]$. (Formally one should add X to Δ_0, extend h_0 by defining $h_0(x_i) = x_i$ for all x_i in X, and prove that the resulting homomorphism is an extension of H).

We now define a *b-fst* K such that $K = B \circ H$. Let $K = \langle \Sigma, \Omega, Q, Q_d, R_K \rangle$ where R_K is obtained by the following requirements.

(1) For all σ in Σ_0, q in Q and t in T_Δ, if $\sigma \to q(t)$ is in R, then $\sigma \to q(t')$ is in R_K, where $t' = H(t)$.

(2) For all $k \geq 1$, σ in Σ_k, $q_1, \cdots, q_k$, q in Q and t in $T_\Delta[X_k]$, if $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ is in R, then $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t')$ is in R_K, where $t' = H(t)$.

Intuitively K simulates the composition of B and H by producing, in each step, the H-translation of the right hand side of the rule applied by B. This technique is well known from [7] and [9].

Observe that if B and H are linear then so is K, and if B is deterministic then so is K. This proves (2) and (3). □

Next we consider compositions involving *fta*.

LEMMA 4.2.

(1) *B-FST* $\circ$ *FTA* $\subseteq$ *B-FST*,

(2) *LB-FST* $\circ$ *FTA* $\subseteq$ *LB-FST*.

Proof. Consider an arbitrary bottom-up *fst* $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ and an arbitrary *fta* $L = \langle \Delta, \Delta, P, P_d, S \rangle$. We may assume that L is a deterministic bottom-up *fta* (see Section 1). We extend the alphabet of L to $\Delta \cup X$ (by adding X to Δ_0), so that the variables in the rules of L are now allowed to range over $T_\Delta[X]$.

We now define a *b-fst* K which defines $B \circ L$. Let $K = \langle \Sigma, \Delta, Q \times P, Q_d \times P_d, R_K \rangle$, where R_K is obtained by the following requirements.

(1) For all σ in Σ_0, $\langle q, p \rangle$ in $Q \times P$ and t in T_Δ, if the rule $\sigma \to q(t)$ is in R and $t \overset{*}{\underset{L}{\Rightarrow}} p(t)$, then the rule $\sigma \to \langle q, p \rangle (t)$ is in R_K.

(2) For all $k \geq 1$, σ in Σ_k, $q_1, \cdots, q_k$, q in Q, $p_1, \cdots, p_k$, p in P and t in $T_\Delta[X_k]$, if the rule $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ is in R and $t[p_1(x_1), \cdots, p_k(x_k)] \overset{*}{\underset{L}{\Rightarrow}} p(t)$, then the rule $\sigma(\langle q_1, p_1 \rangle (x_1) \cdots \langle q_k, p_k \rangle (x_k)) \to \langle q, p \rangle (t)$ is in R_K.

Note that if B is linear, then K is also linear.

By straightforward induction on t and by some intuitively obvious properties of $\overset{*}{\underset{L}{\Rightarrow}}$, it can easily be shown that, for all t in T_Σ, s in T_Δ, q in Q and p in P,

$t \overset{*}{\underset{K}{\Rightarrow}} \langle q, p\rangle(s)$ if and only if $t \overset{*}{\underset{B}{\Rightarrow}} q(s)$ and $s \overset{*}{\underset{L}{\Rightarrow}} p(s)$. From this it directly follows that $K = B \circ L$. □

The following corollary is an immediate consequence of Lemma 4.2(1).

COROLLARY 4.3. *The class of B-FST surface sets is closed under intersection with a recognizable tree language.*

We now turn to compositions involving relabelings and finite state relabelings.

LEMMA 4.4.

(1) $LB\text{-}FST \circ RELAB \subseteq LB\text{-}FST$,

(2) $B\text{-}FST \circ DQRELAB \subseteq B\text{-}FST$,

(3) $DB\text{-}FST \circ DQRELAB \subseteq DB\text{-}FST$.

Proof. The proof of (1) is as follows. Consider an arbitrary *lb-fst* $B = \langle \Sigma, \Delta, Q, Q_d, R\rangle$ and an arbitrary relabeling L determined (as described in Lemma 3.3(1)) by relations $l_k \subseteq \Delta_k \times \Omega_k$, $k \geq 0$, where Ω is a ranked alphabet. We extend the relabeling L to one from $T_\Delta[X]$ into $T_\Omega[X]$ by adding X to Δ_0 and adding all pairs $\langle x_i, x_i\rangle$ with x_i in X to l_0. Let $K = \langle \Sigma, \Omega, Q, Q_d, R_K\rangle$ where R_K is obtained as follows.

(1) If $\sigma \to q(t)$ is in R and t' is in $L(t)$, then $\sigma \to q(t')$ is in R_K.

(2) If $\sigma(q_1(x_1)\cdots q_k(x_k)) \to q(t)$ is in R and t' is in $L(t)$, then $\sigma(q_1(x_1)\cdots q_k(x_k)) \to q(t')$ is in R_K.

It is easy to see that $K = B \circ L$.

The proof of (2) and (3) can be obtained by an obvious generalization of the proof of Lemma 4.2. It is left to the reader. □

We are now in a position to prove the composition results concerning linear and deterministic bottom-up *fst*.

THEOREM 4.5.

(1) $LB\text{-}FST \circ B\text{-}FST \subseteq B\text{-}FST$,

(2) $LB\text{-}FST \circ LB\text{-}FST \subseteq LB\text{-}FST$.

Proof. We prove (1) and (2) simultaneously as follows.

$$
\begin{aligned}
LB\text{-}FST \circ (L)B\text{-}FST &\subseteq LB\text{-}FST \circ RELAB \circ FTA \circ (L)HOM && \text{by Theorem 3.5}\\
&\subseteq LB\text{-}FST \circ FTA \circ (L)HOM && \text{by Lemma 4.4(1)}\\
&\subseteq LB\text{-}FST \circ (L)HOM && \text{by Lemma 4.2(2)}\\
&\subseteq (L)B\text{-}FST && \text{by Lemma 4.1. } \square
\end{aligned}
$$

THEOREM 4.6.

(1) $B\text{-}FST \circ DB\text{-}FST \subseteq B\text{-}FST$,

(2) $DB\text{-}FST \circ DB\text{-}FST \subseteq DB\text{-}FST$.

Proof. We prove (1) and (2) simultaneously as follows.

$$
\begin{aligned}
(D)B\text{-}FST \circ DB\text{-}FST &\subseteq (D)B\text{-}FST \circ DQRELAB \circ HOM && \text{by Theorem 3.15(3)}\\
&\subseteq (D)B\text{-}FST \circ HOM && \text{by Lemma 4.4}\\
&\subseteq (D)B\text{-}FST && \text{by Lemma 4.1. } \square
\end{aligned}
$$

It easily follows from the above two theorems that

$$(L)B\text{-}FST = RELAB \circ FTA \circ (L)HOM \quad \text{(cf. Theorem 3.5)},$$
$$(L)B\text{-}FST = QRELAB \circ (L)HOM \quad \text{and}$$
$$DB\text{-}FST = DQRELAB \circ HOM \quad \text{(cf. Theorem 3.15)}.$$

Let us now compare the above theorems with results concerning top-down *fst*. It is well known that results "dual" to Theorem 4.6 hold for top-down *fst*, namely $DT\text{-}FST \circ (D)T\text{-}FST \subseteq (D)T\text{-}FST$ ([7, 9]). Note that one would expect such a result from property $(B1)$, which was responsible for nonclosure under composition of *T-FST*. From that same property one would also expect results dual to Theorem 4.5, namely $(L)T\text{-}FST \circ LT\text{-}FST \subseteq (L)T\text{-}FST$. Actually we have already seen that these inclusions do not hold (recall Example 2.6). This lack of duality is caused by property $(B2)$, and we shall see in the next section that by adding capability $(B2)$ to *T-FST* we obtain a class *T'-FST* satisfying the above inclusions, and such that $LT'\text{-}FST = LB\text{-}FST$. The relation between linear bottom-up and top-down *fst* can also be expressed as follows.

COROLLARY 4.7. *The following three classes of fst are identical.*
(1) *LB-FST,*
(2) *the closure of LT-FST under composition, and*
(3) *the closure of* $FTA \cup RELAB \cup LHOM$ *under composition.*

Proof. Since $FTA \cup RELAB \cup LHOM \subseteq LT\text{-}FST \subseteq LB\text{-}FST$ and *LB-FST* is closed under composition, we have that (3) is contained in (2) and (2) in (1). But, by Theorem 3.5(2), (1) is contained in (3), and that proves the corollary. □

We conclude this section with the following property of bottom-up *fst*.

COROLLARY 4.8. *The restriction of a bottom-up fst to a recognizable tree language is again a bottom-up fst.*

Proof. If B is in *B-FST* and R in *FTA*, then the restriction of B to $\mathrm{dom}(R)$ is obviously equal to $R \circ B$. Since R is in *LB-FST*, Theorem 4.5(1) implies that $R \circ B$ is in *B-FST*. □

It is easy to see from Example 2.6 that top-down *fst* do not have this property (but the elements of the extended *T'-FST* do).

5. Generalized fst. In this section we introduce a generalized tree transformation model which embodies both bottom-up and top-down features. It can be regarded as obtained by adding properties $(B1)$ and $(B2)$ to the top-down model.

It is first shown that the generalized *fst* are a proper generalization of both bottom-up and top-down *fst* (Theorem 5.7). Then, having proved decomposition and composition results for generalized *fst* similar to those of the previous sections, we show that the class of generalized *fst* is equal to the class $HOM \circ RELAB \circ FTA \circ HOM$ (Theorem 5.10). Next some simple properties of generalized *fst* are listed in Theorem 5.11.

Then we investigate a top-down model obtained by adding $(B2)$ to the ordinary top-down model. It turns out that this new top-down model behaves in

a nice way, dual to the bottom-up model (Theorems 5.13, 5.14 and 5.15). Moreover it defines the same surface sets as the old model (Theorem 5.16).

The generalized tree transformation model is defined as follows.

Definition 5.1. A *generalized finite state transformation* (abbreviated by *gfst*) is a system $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ consisting of sets Σ, Δ, Q and Q_d, defined as in the definition of a tree transducer, together with a finite set R of "*gfst*-rules", defined as follows.

A *gfst-rule* consists of six objects:

(1) a state q in Q,

(2) a nonnegative integer k,

(3) a nonnegative integer m (such that if $k = 0$, then $m = 0$),

(4) a symbol σ in Σ_k,

(5) a tree t in $T_\Delta[X_m]$, and

(6) a mapping $\varphi\colon X_m \to Q(X_k)$ (such that if $k = 0$, then $\varphi = \varnothing$), [2]

and will be denoted as $\langle q, k, m, \sigma \to t, \varphi \rangle$.

We now define the tree transformation defined by a generalized finite state transformation.

Definition 5.2. Let $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a *gfst*. For each q in Q and t in T_Σ, the set of all *q-translations of t*, denoted by $M_q(t)$, is defined inductively as follows:

(1) for σ in Σ_0, $M_q(\sigma) = \{s \in T_\Delta | \langle q, 0, 0, \sigma \to s, \varnothing \rangle \in R\}$,

(2) for $k \geq 1$, $\sigma \in \Sigma_k$ and $t_1, \cdots, t_k$ in T_Σ, $M_q(\sigma(t_1 \cdots t_k)) = \{r[s_1, \cdots, s_m] |$ $\langle q, k, m, \sigma \to r, \varphi \rangle \in R$ and for all i, $1 \leq i \leq m$, if $\varphi(x_i) = p(x_j)$, then $s_i \in M_p(t_j)\}$.

We define the *tree transformation defined by M* to be

$$M = \{\langle t, s \rangle \in T_\Sigma \times T_\Delta | \, s \in M_q(t) \text{ for some } q \text{ in } Q_d\}.$$

As usual, the tree transformation defined by a *gfst* will also be called a *gfst*. The class of all *gfst* will be denoted by *GFST*.

As an example, let us consider a *gfst* $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ with a *gfst*-rule $\langle q, k, m, \sigma \to t, \varphi \rangle$ in R, such that $k = 2$, $m = 4$, $t = \tau(x_1 x_3 x_1 x_2)$ with τ in Δ_4, and $\varphi(x_1) = q_1(x_1)$, $\varphi(x_2) = q_1(x_1)$, $\varphi(x_3) = q_2(x_1)$ and $\varphi(x_4) = q_3(x_2)$ for certain q_1, q_2 and q_3 in Q. Suppose that M has no other rule with the same q, k and σ. Then, given some input tree $\sigma(t_1 t_2)$ in T_Σ, $M_q(\sigma(t_1 t_2))$ is the set of all trees $t[s_1, s_2, s_3, s_4] = \tau(s_1 s_3 s_1 s_2)$ in T_Δ, such that s_1 and s_2 are q_1-translations of t_1, s_3 is a q_2-translation of t_1 and s_4 is a q_3-translation of t_2. Thus this rule requires the existence of a q_3-translation of t_2 although the output of this translation does not occur in the q-translation of $\sigma(t_1 t_2)$ (note that this is in accordance with the usual meaning of the set-theoretical notation used in Definition 5.2(2)).

Notation 5.3. A *gfst*-rule $\langle q, k, m, \sigma \to t, \varphi \rangle$ can be described concisely in a format similar to *b-fst* and *t-fst* rules as follows: for $k = 0$, $q(\sigma) \to t$; for

[2] Recall that $Q(X_k) = \{q(x) | q \in Q \text{ and } x \in X\}$.

$k \geq 1$, $q(\sigma(x_1 \cdots x_k)) \to t \langle q_1(x_{i_1}), q_2(x_{i_2}), \cdots, q_m(x_{i_m})\rangle$, where for $j \in \{1, 2, \cdots, m\}$, $\varphi(x_j) = q_j(x_{i_j})$.

Thus the rule discussed in the example above would be written as $q(\sigma(x_1x_2)) \to \tau(x_1x_3x_1x_2) \langle q_1(x_1), q_1(x_1), q_2(x_1), q_3(x_2)\rangle$ and the interpretation of this rule applied to $\sigma(t_1t_2)$ is that $\langle s_1, s_2, s_3, s_4\rangle$ must be found such that s_1 and s_2 are q_1-translations of t_1, s_3 is a q_2-translation of t_1 and s_4 is a q_3-translation of t_2, and, having found these, a q-translation of $\sigma(t_1t_2)$ is obtained by substituting $\langle s_1, s_2, s_3, s_4\rangle$ in $\tau(x_1x_3x_1x_2)$. In other words, one can think of the translation process as first applying rules top-down inside the angular brackets and then substituting the results in the trees outside the angular brackets (that is, the angular brackets represent the square brackets of substitution). Again, in other words, one can think of a *gfst* as a *t-fst* together with the possibilities (*B*1) of placing a copy of a processed input subtree in the output tree (by the fact that, in the *gfst*-rule of Notation 5.3, there may be repetitions of variables in t), and (*B*2) of deleting a processed subtree (by the fact that t need not contain all variables of X_m). Note that a *gfst* even has the power of, for instance, inspecting a subtree and translating it with another state. For example, a *gfst* might have the rules

$$q(\sigma(x)) \to \tau(x_1) \langle q_1(x), q_2(x)\rangle \quad \text{and}$$
$$q(\sigma(x)) \to \tau(x_1) \langle q_3(x), q_4(x)\rangle,$$

which can be interpreted as meaning that the subtree of σ is q_1- or q_3-translated depending on whether this subtree has "property" q_2 or q_4 respectively. Obviously the properties that can be checked by a *gfst* in such a way, are whether a tree belongs to a recognizable tree language or not.

Next we define the notion of a linear *gfst*. Intuitively, a *gfst* is linear if it never constructs or considers a copy of a subtree during the translation process. This is formalized as follows.

Definition 5.4. A *gfst* $\langle \Sigma, \Delta, Q, Q_d, R\rangle$ is *linear* (abbreviated by *lgfst*) if for each *gfst*-rule $\langle q, k, m, \sigma \to t, \varphi\rangle$ the following two requirements are satisfied:

(1) t is linear, and

(2) if $i \neq j$, $\varphi(x_i) = q_1(x)$ and $\varphi(x_j) = q_2(y)$, then $x \neq y$.

The second requirement in the above definition expresses that, in the notation of Notation 5.3, no variable of X_k occurs more than once inside the angular brackets.

Both top-down and bottom-up *fst* can be characterized as special cases of *gfst*. This is done in the next two lemmas.

LEMMA 5.5. *Each top-down fst can be defined by a gfst satisfying the requirement that in each of its rules $\langle q, k, m, \sigma \to t, \varphi\rangle$ the tree t is linear and nondeleting with respect to X_m. Conversely, each such gfst defines a top-down fst.*

Proof. Consider an arbitrary *t-fst* $T = \langle \Sigma, \Delta, Q, Q_d, R\rangle$. Let M be the *gfst* $\langle \Sigma, \Delta, Q, Q_d, R_M\rangle$, where R_M is obtained by the following requirements.

(1) If $q(\sigma) \to r$ is in R, then $\langle q, 0, 0, \sigma \to r, \varnothing\rangle$ is in R_M.

(2) If $q(\sigma(x_1 \cdots x_k)) \to r[q_1(x_{i_1}), \cdots, q_m(x_{i_m})]$ is in R, where r is linear and nondeleting with respect to X_m and, for each j in $\{1, \cdots, m\}$, x_{i_j} is in X_k, then

the *gfst*-rule $\langle q, k, m, \sigma \to r, \varphi \rangle$ is in R_M, where, for each j in $\{1, \cdots, m\}$, $\varphi(x_j) = q_j(x_{i_j})$. (Recall from the end of section 1 that each *t-fst* rule can be written in the above form).

Clearly M satisfies the requirement stated in the lemma. To show that $M = T$, we shall prove by induction on t, that for each t in T_Σ, q in Q and s in T_Δ, $q(t) \overset{*}{\underset{T}{\Rightarrow}} s$ if and only if $s \in M_q(t)$.

If t is in Σ_0, then the statement is obvious. Now let $t = \sigma(t_1 \cdots t_k)$ for certain $k \geq 1$, σ in Σ_k and $t_1, \cdots, t_k$ in T_Σ, and assume that the statement holds for $t_1, \cdots, t_k$.

The only-if part of the statement is proved as follows. Suppose that $q(\sigma(t_1 \cdots t_k)) \overset{*}{\Rightarrow} s$. Suppose that the first step of this derivation results from the application of a rule of the form given in (2) above. Thus we have $q(\sigma(t_1 \cdots t_k)) \Rightarrow r[q_1(t_{i_1}), \cdots, q_m(t_{i_m})] \overset{*}{\Rightarrow} s$. Then by Lemma 1.2(2), there exist $s_1, \cdots, s_m$ in T_Δ such that $s = r[s_1, \cdots, s_m]$ and $q_j(t_{i_j}) \overset{*}{\Rightarrow} s_j$ for all j, $1 \leq j \leq m$. Hence, by induction, $s_j \in M_{q_j}(t_{i_j})$. Consequently, by (2) above, it follows from Definition 5.2 that $r[s_1, \cdots, s_m] \in M_q(\sigma(t_1 \cdots t_k))$.

The proof of the if part of the statement is left to the reader. This proves the first part of the lemma.

To prove the second part of the lemma, consider an arbitrary *gfst* $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ satisfying the requirement of the lemma. Let T be the *t-fst* $\langle \Sigma, \Delta, Q, Q_d, R_T \rangle$ where R_T is obtained as follows.

(1) If $\langle q, 0, 0, \sigma \to t, \varnothing \rangle$ is in R, then $q(\sigma) \to t$ is in R_T.

(2) If $\langle q, k, m, \sigma \to t, \varphi \rangle$ is in R (with $k \geq 1$), then the rule $q(\sigma(x_1 \cdots x_k)) \to t[\varphi(x_1), \cdots, \varphi(x_m)]$ is in R_T.

It is easy to see that the *gfst* corresponding to T according to the construction at the beginning of this proof is M itself. Hence, by the above proof, $T = M$. □

LEMMA 5.6. *Each bottom-up fst can be defined by a gfst satisfying the requirement that, for each of its rules* $\langle q, k, m, \sigma \to t, \varphi \rangle$, *if* $i \neq j$, $\varphi(x_i) = q_1(x)$ *and* $\varphi(x_j) = q_2(y)$, *then* $x \neq y$. *Conversely, each such gfst defines a bottom-up fst.*

Proof. Consider an arbitrary *b-fst* $B = \langle \Sigma, \Delta, Q, Q_d, R \rangle$. Let M be the *gfst* $\langle \Sigma, \Delta, Q, Q_d, R_M \rangle$ where R_M is obtained as follows.

(1) If $\sigma \to q(t)$ is in R, then $\langle q, 0, 0, \sigma \to t, \varnothing \rangle$ is in R_M.

(2) If $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ is in R, then the *gfst*-rule $\langle q, k, k, \sigma \to t, \varphi \rangle$ is in R_M, where, for each j in $\{1, \cdots, k\}$, $\varphi(x_j) = q_j(x_j)$.

Clearly M satisfies the requirement stated in the lemma. It is left to the reader to prove, using Lemma 1.1, that $M = B$.

To prove the second part of the lemma, consider an arbitrary *gfst* $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ satisfying the requirement of the lemma. Obviously, by that requirement, $m \leq k$ in each of its rules $\langle q, k, m, \sigma \to t, \varphi \rangle$. Let us first suppose that, in each of its rules, $m = k$. Then it follows from the requirement on M that we may assume (by an eventual application of a suitable permutation to the variables of t) that for each rule $\langle q, k, k, \sigma \to t, \varphi \rangle$ there are states $q_1, \cdots, q_k$ such that, for all j in $\{1, \cdots, k\}$, $\varphi(x_j) = q_j(x_j)$. If we construct the *b-fst* $B = \langle \Sigma, \Delta, Q, Q_d, R_B \rangle$ which has, corresponding to each such rule, a rule $\sigma(q_1(x_1) \cdots q_k(x_k)) \to q(t)$ in R_B, then obviously $B = M$.

Now suppose that there are rules of M with $m < k$. Then we can construct a *gfst* $M' = \langle \Sigma, \Delta, Q \cup \{e\}, Q_d, R' \rangle$, defining the same transformation as M and still satisfying the requirement, such that all rules satisfy $m = k$, as follows. Just as in the proof of Theorem 2.8, M' has an "erasing state" e. If $\langle q, k, m, \sigma \to t, \varphi \rangle$ is in R ($k \geq 1$), then $\langle q, k, k, \sigma \to t, \varphi' \rangle$ is in R', where $\varphi'(x_j) = \varphi(x_j)$ if $j \leq m$ and $\varphi'(x_j) = e(x_i)$ if $j > m$, i being the $(j-m)$th number in $\{1, 2, \cdots, k\}$ such that for no n in $\{1, \cdots, m\}$ and no p in Q $\varphi(x_n) = p(x_i)$. If $\langle q, 0, 0, \sigma \to t, \varnothing \rangle$ is in R, then it is in R'. Finally, all rules of the form $\langle e, k, k, \sigma \to \delta_0, \varphi \rangle$ where $\varphi(x_j) = e(x_j)$ and δ_0 is an arbitrary element of Δ_0, are in R'. It is left to the reader to show that $M' = M$. From this the lemma follows. □

Having characterized both *t-fst* and *b-fst* as special *gfst* in the above lemmas, we can clearly see that the difference between *b-fst* and *t-fst* is caused by the different ways in which the q-translation of a tree $\sigma(t_1 \cdots t_k)$ depends on the translation of its subtrees. In both cases (in fact in every *gfst*), substitution is involved: $M_q(\sigma(t_1 \cdots t_k))$ consists of trees r' such that $\langle q, k, m, \sigma \to r, \varphi \rangle$ is a *gfst*-rule and r' is the result of substituting certain elements of $M_p(t_i)$ for each string $p(x_i)$ in $r[\varphi(x_1), \cdots, \varphi(x_m)]$. In the bottom-up case (that is, for *gfst* satisfying the requirement of Lemma 5.6) this substitution is "deterministic", in the sense that *one* element of $M_p(t_i)$ should be substituted for all occurrences of $p(x_i)$ in $r[\varphi(x_1), \cdots, \varphi(x_m)]$ (cf. property (*B*1)). In the top-down case (see Lemma 5.5) this substitution is "nondeterministic" in the sense that for each occurrence of $p(x_i)$ in $r[\varphi(x_1), \cdots, \varphi(x_m)]$ *some* element of $M_p(t_i)$ should be substituted (cf. property (*T*)). In a general *gfst* both "deterministic" and "nondeterministic" substitutions are present, regulated by the mapping φ.

The previous lemmas imply that *B-FST* and *T-FST* are contained in *GFST*. The next theorem shows that containment is proper.

THEOREM 5.7. *B-FST* $\cup$ *T-FST* $\subset$ *GFST*.

Proof. Inclusion follows from Lemmas 5.5 and 5.6. To show proper inclusion, consider the *gfst* $M = \langle \Sigma, \Omega, Q, Q_d, R \rangle$, where $\Sigma_0 = \{a\}$, $\Sigma_1 = \{a, \sigma\}$, $\Omega_0 = \Omega_1 = \{a, b\}$, $\Omega_3 = \{\sigma\}$, $Q = Q_d = \{*\}$ and R contains the following rules (in Notation 5.3):

$$*(\sigma(x)) \to \sigma(x_1x_1x_2)\ \langle *(x), *(x) \rangle,$$
$$*(a(x)) \to a(x_1)\ \langle *(x) \rangle,$$
$$*(a(x)) \to b(x_1)\ \langle *(x) \rangle,$$
$$*(a) \to a \quad \text{and} \quad *(a) \to b.$$

We first show that no bottom-up *fst* can define the transformation M. Consider in T_Σ a tree with top symbol σ and a tail of n symbols a, where $n \geq 1$. Such a tree is translated by M into all trees in T_Ω of the form $\sigma(t_1t_1t_2)$, where t_1 and t_2 are tails of length n, arbitrarily labeled by a's and b's. Suppose that a bottom-up *fst* B exists such that $B = M$, and consider the homomorphism H from T_Ω into T_Δ (where Δ is the ranked alphabet of Examples 2.1 and 2.2) determined by the functions h_0, h_1, h_3 (see Lemma 3.3(2)), where $h_0(a) = a$, $h_0(b) = b$, $h_1(a) = a(x_1)$, $h_1(b) = b(x_1)$ and $h_3(\sigma) = \sigma(x_2x_3)$. Obviously, H transforms a tree $\sigma(t_1t_1t_2)$ as above into the tree $\sigma(t_1t_2)$. By Theorem 4.6(1), $B \circ H$ is in *B-FST*.

But it easily follows that $B \circ H$ is a counterexample to Example 2.2, and therefore such a B cannot exist.

It is intuitively clear from a comparison with Example 2.1 that M can neither be defined by any *t-fst*. A formal proof, similar to the above but using a homomorphism with $h_3(\sigma) = \sigma(x_1x_2)$ instead, can easily be given to show that M can even not be defined by any *t'-fst* (to be defined in Definition 5.12). We leave it to the reader. □

In Theorem 5.10 we shall prove that *GFST* is identical to the class $HOM \circ RELAB \circ FTA \circ HOM$. To do this we need a decomposition and a composition result.

LEMMA 5.8. $GFST \subseteq HOM \circ B\text{-}FST$.

Proof. The proof of this lemma is similar to that of Lemma 3.6. Intuitively, viewing the *gfst* as consisting of a top-down part and a substitution part (recall the discussion following Notation 5.3), we can decompose the top-down part into a homomorphism and a linear top-down part, as in Lemma 3.6. But, in view of Theorem 2.8, it obviously suffices to have one bottom-up *fst* to simulate both the linearized top-down part and the substitution part of the *gfst*.

Let us give some hints to the reader who wishes to formally prove the lemma. Let $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ be a *gfst*. Let, for each x in X and each *gfst*-rule $r = \langle q, k, m, \sigma \to t, \varphi \rangle$, r_x denote the cardinality of the set $\{i | 1 \leq i \leq m$ and $\varphi(x_i) = q(x)$ for some q in $Q\}$, and let $n = \max \{r_x | x \in X$ and $r \in R\}$. Thus, intuitively, n is the maximal number of copies of subtrees needed by the top-down part of M in any step. The homomorphism H can now be defined as in the proof of Lemma 3.6. A precise formal definition of the bottom-up *fst* B is tedious. Let us give an example of the kind of rules needed in B. Suppose that R contains the rule

$$q(\sigma(x_1x_2)) \to \tau(x_1x_3x_1x_2)\ \langle q_1(x_1), q_1(x_1), q_2(x_1), q_3(x_2) \rangle.$$

Then n is at least 3. Suppose that $n = 3$ and denote, for convenience, the variables $x_1, \cdots, x_6$ by $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}$ and x_{23} respectively. Then the rule

$$\sigma(q_1(x_{11})q_1(x_{12})q_2(x_{13})q_3(x_{21})e(x_{22})e(x_{23})) \to \tau(x_{11}x_{13}x_{11}x_{12})$$

could be in B, where e is an "erasing state". □

LEMMA 5.9.

(1) $GFST \circ HOM \subseteq GFST$,

(2) $GFST \circ FTA \subseteq GFST$.

Proof. (1) Let M be a *gfst* and H a homomorphism. Extend H to trees with variables by defining $H(x) = x$ for all x in X. Let K be the *gfst*, such that, if $\langle q, k, m, \sigma \to t, \varphi \rangle$ is a rule of M, then $\langle q, k, m, \sigma \to H(t), \varphi \rangle$ is a rule of K. Then $K = M \circ H$.

(2) Similar to Lemma 4.2. □

We are now able to prove the following main result about *GFST*.

THEOREM 5.10. *The following three classes of tree transformations are equal:*

(1) *GFST*,

(2) $HOM \circ RELAB \circ FTA \circ HOM$, *and*

(3) $HOM \circ B\text{-}FST$.

Proof. Since $GFST \subseteq HOM \circ B\text{-}FST$ by Lemma 5.8, and $B\text{-}FST \subseteq RELAB \circ FTA \circ HOM$ by Theorem 3.5, it suffices to prove that $HOM \circ RELAB \circ FTA \circ HOM \subseteq GFST$. First, by the fact that $DT\text{-}FST \circ T\text{-}FST \subseteq T\text{-}FST$ (see [9]), $HOM \circ RELAB \circ FTA \circ HOM \subseteq T\text{-}FST \circ FTA \circ HOM$. Secondly,

$$\begin{aligned} T\text{-}FST \circ FTA \circ HOM &\subseteq GFST \circ FTA \circ HOM && \text{(by Theorem 5.7)} \\ &\subseteq GFST \circ HOM && \text{(by Lemma 5.9(2))} \\ &\subseteq GFST && \text{(by Lemma 5.9(1)).} \ \square \end{aligned}$$

Note that, since $GFST = HOM \circ B\text{-}FST$ and $B\text{-}FST = QRELAB \circ HOM$, it follows that $GFST = HOM \circ QRELAB \circ HOM$.

In the next theorem we list some simple properties of *gfst*.

THEOREM 5.11.

(1) *The domain of a gfst is recognizable.*
(2) *GFST surface sets are closed under intersection with a recognizable tree language and under homomorphisms.*
(3) *GFST surface sets are recursive.*
(4) *GFST is not closed under composition.*

Proof. (1) Let M be an arbitrary *gfst*. By Theorem 5.10, $M = H \circ B$, where H is a homomorphism and B a bottom-up *fst*. Since the domain of a bottom-up *fst* is recognizable (Corollary 3.12), there is an *fta* R such that $\mathrm{dom}(R) = \mathrm{dom}(B)$, and so $\mathrm{dom}(M) = \mathrm{dom}(H \circ R)$. But, by Lemma 4.2, $H \circ R$ is a *b-fst*, and so, again by Corollary 3.12, its domain is recognizable.

(2) Immediate from Lemma 5.9.

(3) Follows from (1) and (2) as for top-down *fst* (see [7]).

(4) Consider the *gfst* in the proof of Theorem 5.7 and let it be followed by a relabeling which changes all a's nondeterministically into, say, c's and d's. It is intuitively clear that the resulting tree transformation is not a *gfst*. A formal proof is left to the reader. □

We conjecture that the class of *T-FST* surface sets is a proper subset of the class of *GFST* surface sets. In fact, consider the *gfst* $M = \langle \Sigma, \Delta, Q, Q_d, R \rangle$, where $\Sigma_0 = \{a\}$, $\Sigma_1 = \{a, \sigma\}$, $\Delta_0 = \{a, b\}$, $\Delta_2 = \{a, b, \sigma\}$, $Q = Q_d = \{*\}$ and R consists of the rules (in Notation 5.3)

$$\begin{aligned} &*(\sigma(x)) \to \sigma(x_1 x_1)\ \langle *(x) \rangle, \\ &*(a(x)) \to a(x_1 x_2)\ \langle *(x), *(x) \rangle, \\ &*(a(x)) \to b(x_1 x_2)\ \langle *(x), *(x) \rangle, \\ &*(a) \to a \quad \text{and} \quad *(a) \to b. \end{aligned}$$

Let M work on the recognizable tree language of all trees of the form $\sigma(a(a(\cdots a(a)\cdots)))$. The resulting *GFST* surface set consists of all trees $\sigma(tt)$, where t is any balanced binary tree labeled by a's and b's. This set does not seem to be a *T-FST* surface set.

The failure of property (*B*2) for top-down *fst* stood in the way of some nice results about *T-FST*.

We now turn to a generalization of the top-down model which is obtained by adding the capability of inspecting subtrees before processing them (cf. property (*B*2)). The resulting class of tree transformations, denoted by *T'-FST*, behaves in a nice, dual way with respect to *B-FST*. Furthermore, it will turn out that the class of *T'-FST* surface sets is equal to the class of *T-FST* surface sets. Thus, theoretically speaking, *T'-FST* is to preferred above *T-FST* in many ways.

Definition 5.12. A *t'-fst* is a *gfst* satisfying the requirement that in each of its rules $\langle q, k, m, \sigma \to t, \varphi \rangle$ the tree t is linear. Furthermore, an *lt'-fst* (that is, a linear *t'-fst*) is a *t'-fst* such that, for each of its rules $\langle q, k, m, \sigma \to t, \varphi \rangle$, if $i \neq j$, $\varphi(x_i) = q_1(x)$ and $\varphi(x_j) = q_2(y)$, then $x \neq y$.

The classes of *t'-fst* and *lt'*-fst will be denoted by *T'-FST* and *LT'-FST* respectively.

The reader should compare the above definition with the definition of a linear *gfst* (Definition 5.4) and with the *gfst* characterizations of top-down and bottom-up *fst* in Lemmas 5.5 and 5.6. From this comparison the following theorem results.

THEOREM 5.13. *LGFST = LB-FST = LT'-FST.*

Proof. Consider requirements (1) and (2) in the definition of an *lgfst* (Definition 5.4). Obviously, requirement (1) is exactly the definition of a *t'-fst*, and requirement (2) expresses the linearity of the *t'-fst*. Hence *LGFST = LT'-FST* (even as classes of 5-tuples). On the other hand, requirement (2) is exactly the *gfst* characterization of a *b-fst* (given in Lemma 5.6), and it is easy to see that requirement (1) expresses the linearity of that *b-fst*. Consequently *LGFST = LB-FST*, and the theorem is proved. □

We now give an alternative, more intuitive and practical, formulation of the notion of a *t'-fst*.

Suppose we have a *t-fst* $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$ and we associate with each of its rules $q(\sigma(x_1 \cdots x_k)) \to t$ a finite set of pairs $\langle x, F \rangle$, where x is in X and F is a finite tree automaton in T_Σ. Suppose that, during the translation process, we only allow application of the rule $q(\sigma(x_1 \cdots x_k)) \to t$ to a tree $q(\sigma(t_1 \cdots t_k))$ if $t_i \in \mathrm{dom}(F)$ for all pairs $\langle x_i, F \rangle$ associated with the rule. Then the resulting transformation will be a *t'-fst*, and, vice versa, each *t'-fst* can be obtained in such a way. Thus the class of such "*t-fst* with recognizable conditions" has exactly the same translating power as the class of *t'-fst*. A formal proof is left to the reader.

Thus we see that a *t'-fst* has actually more "non *t-fst*" capabilities than just (*B*2). However, one may give a more careful definition of the (*B*2) capability and show that, by adding it to the top-down model, the same class of transformations (namely *T'-FST*) results. Note that an *lt'-fst* is precisely a *t-fst* together with property (*B*2). It follows from Theorem 5.13 that property (*B*2) was indeed entirely responsible for the difference between *LT-FST* and *LB-FST* (Theorem 2.8), and that by addition of (*B*2) the linear top-down *fst* become closed under composition (cf. Theorems 2.7 and 4.5(2)).

In the next theorem we show a composition result for *t'-fst* dual to that for *b-fst* in Theorem 4.5(1).

THEOREM 5.14. *T'-FST* ∘ *LT'-FST* ⊆ *T'-FST.*

Proof. Since *LT'-FST* = *LB-FST* = *RELAB* ∘ *FTA* ∘ *LHOM*, it suffices to prove that *T'-FST* ∘ *RELAB* ⊆ *T'-FST*, *T'-FST* ∘ *FTA* ⊆ *T'-FST* and *T'-FST* ∘ *LHOM* ⊆ *T'-FST*. The first two inclusions can be proved using the alternative formulation of a *t'-fst* discussed above. The third inclusion can easily be seen from Lemma 5.9(1). □

We now show how to decompose *t'-fst* (cf. Lemma 3.8 and Theorem 3.9).

THEOREM 5.15. *The following three classes of tree transformations are equal:*

(1) *T'-FST,*

(2) *HOM* ∘ *RELAB* ∘ *FTA* ∘ *LHOM, and*

(3) *HOM* ∘ *LB-FST.*

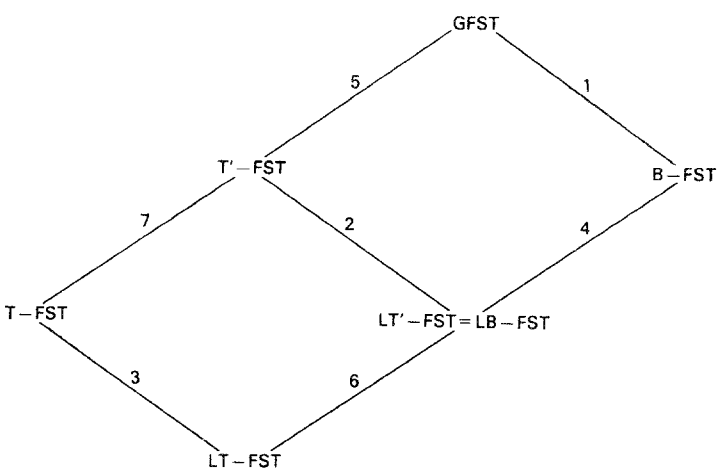
Proof. By previous results it suffices to show that *T'-FST* = *HOM* ∘ *LT'-FST*. Generalizing the proof of Lemma 3.6 in an obvious way it follows that *T'-FST* ⊆ *HOM* ∘ *LT'-FST*. Furthermore, by Theorem 5.14, *HOM* ∘ *LT'-FST* ⊆ *T'-FST*. □

Finally we show that with respect to surface sets *T'-FST* and *T-FST* are the same.

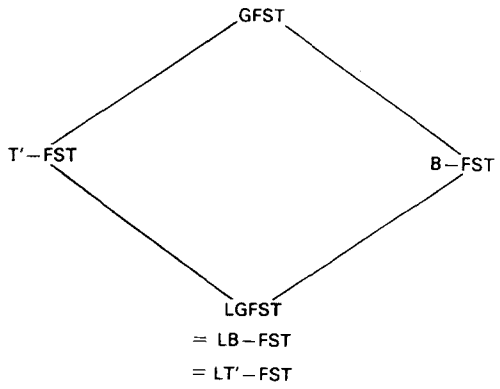
THEOREM 5.16. *The class of T'-FST surface sets is equal to the class of T-FST surface sets.*

Proof. Trivially, every *T-FST* surface set is a *T'-FST* surface set (recall Lemma 5.5). To show the converse, let *M* be an arbitrary *t'-fst* and let *U* be an arbitrary recognizable tree language. We have to prove that *M*(*U*) is a *T-FST* surface set. By the previous theorem, *M* can be decomposed into *H* ∘ *R* ∘ *F* ∘ *L*, where *H* ∈ *HOM*, *R* ∈ *RELAB*, *F* ∈ *FTA* and *L* ∈ *LHOM*. Obviously *H*(*U*) is a *T-FST* surface set, and *M*(*U*) = *L*(*F*(*R*(*H*(*U*)))). Hence *M*(*U*) can be obtained from *H*(*U*) by applying three linear *t-fst* (viz. *R*, *F* and *L*) to it. Since Rounds has proved in [7] that the class of *T-FST* surface sets is closed under linear *t-fst*, it follows that *M*(*U*) is a *T-FST* surface set. □

6. Relationships between classes of fst. Below we give an inclusion diagram of the most important classes of *fst* discussed, together with a number of decomposition results.



An interesting part of this diagram is



where

$$\begin{aligned} GFST &= HOM \circ RELAB \circ FTA \circ HOM \\ T'\text{-}FST &= HOM \circ RELAB \circ FTA \circ LHOM \\ B\text{-}FST &= RELAB \circ FTA \circ HOM \\ LGFST &= RELAB \circ FTA \circ LHOM \end{aligned}$$

(see Theorems 5.10, 5.15, 3.5 and 4.5).

We now consider some relationships between the classes of the first diagram (referring to it by the numbers 1 to 7). It turns out that these classes can be obtained from each other by either "pre-composing" or "post-composing" with homomorphisms.

(1) $GFST = HOM \circ B\text{-}FST$

(2) $T'\text{-}FST = HOM \circ LB\text{-}FST$

(3) $T\text{-}FST = HOM \circ LT\text{-}FST$

(4) $B\text{-}FST = LB\text{-}FST \circ HOM$

(5) $GFST = T'\text{-}FST \circ HOM$

(6) $LB\text{-}FST = LT\text{-}FST \circ LHOM$

(7) $T'\text{-}FST = T\text{-}FST \circ LHOM$

(4, 6) $B\text{-}FST = LT\text{-}FST \circ HOM$

(5, 7) $GFST = T\text{-}FST \circ HOM$

Proofs. (1), (2) and (3) have been proved in Theorems 5.10, 5.15 and 3.7 respectively. The other equalities can easily be proved, using known decompositions, as follows.

(4)
$$\begin{aligned} B\text{-}FST &= RELAB \circ FTA \circ HOM \\ &= RELAB \circ FTA \circ LHOM \circ HOM \\ &= LB\text{-}FST \circ HOM. \end{aligned}$$

We have used that $HOM = LHOM \circ HOM$. That equation is clear from the fact that HOM is closed under composition (Lemma 3.4) and the fact that $LHOM$ contains the identity transformation.

(5) $$\begin{aligned} GFST &= HOM \circ RELAB \circ FTA \circ HOM \\ &= HOM \circ RELAB \circ FTA \circ LHOM \circ HOM \\ &= T'\text{-}FST \circ HOM. \end{aligned}$$

(6) $$\begin{aligned} LB\text{-}FST &\subseteq QRELAB \circ LHOM \\ &\subseteq LT\text{-}FST \circ LHOM \end{aligned}$$

(see the end of Section 3), and

$$\begin{aligned} LT\text{-}FST \circ LHOM &\subseteq LB\text{-}FST \circ LB\text{-}FST \\ &\subseteq LB\text{-}FST \qquad \text{(Theorem 4.5(2)).} \end{aligned}$$

(7) $$\begin{aligned} T'\text{-}FST &= HOM \circ LB\text{-}FST && \text{by (2)} \\ &= HOM \circ LT\text{-}FST \circ LHOM && \text{by (6)} \\ &= T\text{-}FST \circ LHOM && \text{by (3).} \end{aligned}$$

(4, 6) $$\begin{aligned} B\text{-}FST &= LB\text{-}FST \circ HOM && \text{by (4)} \\ &= LT\text{-}FST \circ LHOM \circ HOM && \text{by (6)} \\ &= LT\text{-}FST \circ HOM. \end{aligned}$$

(5, 7) $$\begin{aligned} GFST &= T'\text{-}FST \circ HOM && \text{by (5)} \\ &= T\text{-}FST \circ LHOM \circ HOM && \text{by (7)} \\ &= T\text{-}FST \circ HOM. \end{aligned}$$

7. Conclusion. We have defined three classes of very simple *fst*: *FTA*, *RELAB* and *HOM*, which are independent of the bottom-up/top-down distinction. We have shown that all bottom-up and top-down *fst* are contained in the class $HOM \circ RELAB \circ FTA \circ HOM$, that is, every *fst* can be decomposed into (at most) four very simple *fst*. We have characterized the class $HOM \circ RELAB \circ FTA \circ HOM$ as the class of all generalized *fst*, which combine bottom-up and top-down capabilities.

We have characterized the essential difference between top-down and bottom-up tree transformations as the difference in using the copying facility, informally expressed in properties (*B*1) and (*T*), and formally expressed in a number of decomposition and composition results.

Acknowledgments. I am grateful to J. W. Thatcher and W. C. Rounds, who, by their lucid papers, stimulated my interest in trees. I would like to thank the referee for struggling through the first version of this paper and suggesting so many improvements to it both in style and presentation of ideas. If this paper is still unreadable, it is not his fault.

I would like to mention that part of the results in this paper have been obtained independently by B. S. Baker [1].

REFERENCES

[1] B. S. BAKER, Tree transductions and families of tree languages, *Ph.D.Thesis*, Harvard University, 1973, and *Proc. 5th Ann. ACM Symp. on Theory of Computing* (1973), 200–206.

[2] J. DONER, Tree acceptors and some of their applications, *J. Comp. Syst. Sci.* **4** (1970), 406–451.

[3] S. GINSBURG, *The mathematical theory of context-free languages*, McGraw-Hill Book Co., New York, 1966.

[4] J. E. HOPCROFT and J. D. ULLMAN, *Formal languages and their relation to automata*, Addison-Wesley Publ. Co., Reading, Mass., 1969.

[5] M. MAGIDOR and G. MORAN, Finite automata over finite trees, Hebrew University, *Technical Report No. 30*, 1969.

[6] W. C. ROUNDS, Trees, transducers and transformations, *Ph. D. Thesis*, Stanford University, 1968.

[7] W. C. ROUNDS, Mappings and grammars on trees, *Math. Systems Theory* **4** (1970), 257–287.

[8] J. W. THATCHER, Generalized^2 sequential machine maps, *IBM Res. Report RC 2466*, 1969.

[9] J. W. THATCHER, Generalized^2 sequential machine maps, *J. Comp. Syst. Sci.* **4** (1970), 339–367.

[10] J. W. THATCHER, There's a lot more to finite automata theory than you would have thought, *Proc. 4th Ann. Princeton Conf. on Informations Sciences and Systems* (1970), 263–276. Also published in revised form under the title "Tree automata: an informal survey" in *Currents in the Theory of Computing* (ed. A. V. Aho), Prentice-Hall, 1973, 143–172.

[11] J. W. THATCHER and J. B. WRIGHT, Generalized finite automata theory with an application to a decision-problem of second-order logic, *Math. Systems Theory* **2** (1968), 57–81.

(Received November 16, 1971,
and in revised form August 14, 1974)