INFORMATION AND CONTROL 37, 241–266 (1978)

Tree Transducers and Tree Languages*

BRENDA S. BAKER

Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974

Tree transducers (automata which read finite labeled trees and output finite labeled trees) are used to define a hierarchy of families of "tree languages" (sets of trees). In this hierarchy, families generated by "top-down" tree transducers (which read trees from the root toward the leaves) alternate with families generated by "bottom-up" tree transducers (which read trees from the leaves toward the root). A hierarchy of families of string languages is obtained from the first hierarchy by the "yield" operation (concatenating the labels of the leaves of the trees). Both hierarchies are conjectured to be infinite, and some results are presented concerning this conjecture. A study is made of the closure properties of the top-down and bottom-up families in the hierarchies under various tree and string operations. The families are shown to be closed under certain operations if and only if the hierarchies are finite.

INTRODUCTION

Most automata which have been studied in formal language theory read strings. However, Doner (1970) and Thatcher and Wright (1968) discovered that finite-state automata could be generalized to read finite labeled trees; instead of reading a string from left to right, a finite tree automaton reads a tree from the leaves toward the root ("bottom-up") or from the root toward the leaves ("top-down"). A set of trees ("tree language") accepted by a finite tree automaton is called a *recognizable* set. By generalizing proof techniques of finite string automata, Doner and Thatcher and Wright proved that the family of recognizable sets is closed under union, intersection, complement, and generalized forms of concatenation and Kleene$^+$.

Associated with each tree t is a string called its yield (denoted by $Y(t)$), obtained by concatenating the labels of the leaves of t from left to right. Thus, the yield operation generates a string language from each tree language. Now

* This research was done while the author was at Harvard University. This research was supported in part by the National Science Foundation under Grant NSF-GJ-30409. Part of the research was done while the author held a National Science Foundation Graduate Fellowship. Some of the results were included in the author's doctoral dissertation, Tree transductions and families of tree languages, Harvard University, 1973, and were announced at the Fifth Annual ACM Symposium on Theory of Computing, May, 1973.

0019-9958/78/0373-0241$02.00/0

such a string language is the yield of a recognizable set of trees if and only if it is an e-free context-free language (i.e., a context-free language not containing the empty string) (Thatcher, 1970). Therefore, the yields of recognizable sets are closed under union, concatenation, Kleene$^+$, substitution, and certain other operations (Hopcroft and Ullman, 1969).

Rounds (1968) and Thatcher (1970) added output functions to tree automata to obtain "tree transducers." A tree transducer reads an input tree either top-down or bottom-up, and generates a tree as output. Recognizability is not preserved by either top-down or bottom-up tree transductions (Thatcher, 1973). Moreover, neither the class of top-down transductions nor the class of bottom-up transductions is closed under composition (Thatcher, 1970 and Engelfriet, 1975). Therefore, the composition of tree transductions may be applied to recognizable sets of trees to obtain more complex tree languages. Let D_0 denote the family of recognizable sets. For each n, let D_n denote the family of tree languages generated from the recognizable sets by the composition of n top-down tree transductions. Also, let U_n denote the family of tree languages generated from the recognizable sets by the composition of n bottom-up tree transductions. Section 1 proves the surprising result that for each n, $D_n \subseteq U_{n+1} \subseteq D_{n+1}$; that is, the families generated by the top-down transductions alternate with the families generated by the bottom-up transductions in a single hierarchy. We conjecture that for each n, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$, so that the hierarchy is infinite; in Section 4 a proof is given that $D_0 \subsetneq U_1 \subsetneq D_1 \subsetneq U_2 \subsetneq D_2$. Taking the yield of each family in this D-U hierarchy produces a hierarchy of families of string languages; for each n, $Y(D_n) \subseteq Y(U_{n+1}) \subseteq Y(D_{n+1})$. Again, we conjecture that each inclusion is proper; in Section 4 we prove that $Y(D_0) \subsetneq Y(U_1) \subsetneq Y(D_1) \subsetneq Y(U_2) \subsetneq Y(D_2)$.

It is natural to wonder what properties all top-down (bottom-up) families share simply because they are generated by the same kind of tree transducer and how the differences between top-down and bottom-up tree transducers are reflected in properties of the families in the hierarchies.

Section 2 shows that certain compositional properties of top-down and bottom-up tree transducers translate into closure properties of the families in the hierarchies. By this method, it is shown that each family D_n is closed under transductions performed by "linear" tree transducers (tree transducers which cannot generate more than one output subtree from any input subtree), while each family U_n is closed under deterministic bottom-up transductions. The ability of top-down transducers (but not bottom-up transducers) to generate several distinct output subtrees from the same input subtree is exploited to show that each D_n is closed under tree concatenation and a related operation called tree substitution. Each U_n is closed under tree substitution into the recognizable sets (a restricted form of tree substitution), but results in Section 4 prove that at least U_1 and U_2 are not closed under unrestricted tree substitution. On the other hand, all the families in the hierarchy are closed

under intersection with recognizable sets (Ogden and Rounds, 1972, and Section 2).

In Section 3, it is shown that the families $Y(D_n)$ and $Y(U_n)$ are closed under many string operations which have been commonly studied in formal language theory. In particular, the "extended" yield (i.e., the yield modified by adding the empty string) of each D_n and U_n is closed under union, concatenation, Kleene*, homomorphism, intersection with regular sets, and substitution into context-free sets. In addition, the extended yield of each D_n is closed under inverse homomorphism and substitution.

Section 4 presents some results bearing on the conjecture that the hierarchies are infinite. In particular, for every n, if $D_n \subsetneq U_{n+1}$, then $U_{n+1} \subsetneq D_{n+1}$. Moreover, for every n, if $Y(D_n) \subsetneq Y(U_{n+1})$, then $Y(U_{n+1}) \subsetneq Y(D_{n+1})$. If it could be shown that for every n, if $U_n \subsetneq D_n$ then $D_n \subsetneq U_{n+1}$, and if $Y(U_n) \subsetneq Y(D_n)$ then $Y(D_n) \subsetneq Y(U_{n+1})$, we would have a proof that each hierarchy is infinite. Remarkably, the differences in the known closure properties of the top-down and bottom-up families noted above also are related to the conjecture that the hierarchies are infinite. It is shown that for every n, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$ if and only if U_{n+1} is not closed under tree substitution. Also, for every n, $Y(D_n) \subsetneq Y(U_{n+1}) \subsetneq Y(D_{n+1})$ if and only if $Y(U_{n+1})$ is not closed under inverse homomorphism. Although it is not known whether either hierarchy is finite, a proof is given that if the D-U hierarchy is finite, it must have an odd number k of distinct families, with $K \geqslant 5$. Recently, Perrault (1975) announced the result that $Y(D_1) \subsetneq Y(U_2)$. This result is used to show that the $Y(D) - Y(U)$ hierarchy contains at least five distinct families.

The results in this paper indicate that tree transducers generate "natural" families of tree languages, in that these families are closed under some interesting operations while their yields are closed under many of the same string operations as the "natural" families of regular, context-free, and context-sensitive languages.

Section 1

Finite labeled trees may be defined formally as strings constructed from "ranked" alphabets. An alphabet Σ is *ranked* by a function $r: \Sigma \to N$ which assigns a *rank* to each member of Σ. For each n, $\Sigma_n = r^{-1}(n)$ denotes the set of symbols in Σ which have rank n. Intuitively, the rank of a symbol is the number of sons it has when it labels a node in a tree.

Let Π denote the set containing left and right brackets and comma. To avoid possible confusion, ranked alphabets are not allowed to include elements of Π. For a ranked alphabet Σ, the set Σ_* of (finite labeled) trees over the alphabet Σ is the least set of strings in $(\Sigma \cup \Pi)^*$ such that

(1) $\Sigma_0 \subseteq \Sigma_*$, and

(2) for $n > 0$, $b \in \Sigma_n$, and $t_1, t_2, \ldots, t_n \in \Sigma_*$, $b[t_1, t_2, \ldots, t_n] \in \Sigma_*$.

By convention, if $t = b[t_1, ..., t_n] \in \Sigma_*$, it may be assumed that $n > 0$, $b \in \Sigma_n$, and $t_1, ..., t_n \in \Sigma_*$, unless otherwise specified.

An important parameter of a tree is its *depth*, which is defined inductively as follows:

(1) For $b \in \Sigma_0$, $\text{depth}(b) = 1$;

(2) For $t = b[t_1, ..., t_n] \in \Sigma_*$, $\text{depth}(t) = 1 + \max\{\text{depth}(t_i) \mid 1 \leqslant i \leqslant n\}$.

Associated with each tree t is a string called its *yield* and denoted by $Y(t)$. The yield of a tree is obtained by concatenating the labels of its leaves from left to right according to the following inductive definition:

(1) For $b \in \Sigma_0$, $Y(b) = b$;

(2) For $t = b[t_1, ..., t_n] \in \Sigma_*$, $Y(t) = Y(t_1)\, Y(t_2) \cdots Y(t_n) \in \Sigma_0{}^*$.

For a set of trees T, $Y(T) = \{Y(t) \mid t \in T\}$. If F is a family of tree languages, $Y(F) = \{Y(T) \mid T \in F\}$.

For the purpose of informal discussion, trees may be represented in the usual way by directed ordered graphs, drawn so that the "root" of the tree is at the "top" and the "leaves" of the tree are at the "bottom." With this convention, a tree transducer may read trees either "top-down" (from the root toward the leaves) or "bottom-up" (from the leaves toward the root). We describe top-down transducers first.

Intuitively, a top-down tree transducer generates output in steps as it reads from the root toward the leaves of a tree. Depending on its current state at a node α and the label of α, the transducer generates an output subtree u and starts 0, 1, or more subcomputations on each subtree of α. The output subtrees generated from the subtrees of α are substituted at specially marked leaves of u. If α has no subtrees or if no subcomputations are started on subtrees of α, then this subcomputation is complete. The transducer has completed its computation when all subcomputations are complete. Note that the number of subcomputations may grow exponentially with the size of the input tree. In general, a top-down tree transducer may be nondeterministic so that it has several possible outputs at some nodes and possibly no defined output at other nodes.

The central mechanism in the action of tree transducers is the insertion of subtrees at specially marked leaves of another tree. This process of "tree substitution" is defined as follows.

DEFINITION. Let Σ be a ranked alphabet, with $\Delta_0 \subseteq \Sigma_0$. For each $d \in \Delta_0$, let $T_d \subseteq \Sigma_*$. For $t \in \Sigma_*$, the set of trees obtained by substituting trees from T_d at leaves labeled d, for each $d \in \Delta_0$, is written $t(d\colon T_d \mid d \in \Delta_0)$ and is defined inductively as follows:

(1) If $t \in \Sigma_0 - \Delta_0$, then $t(d\colon T_d \mid d \in \Delta_0) = \{t\}$;

(2) If $t \in \Delta_0$, then $t(d\colon T_d \mid d \in \Delta_0) = T_d$;

(3) If $t = b[t_1, \ldots, t_n] \in \Sigma_*$, then

$$t(d\colon T_d \mid d \in \Delta_0) = \{b[u_1, \ldots, u_n] \mid \text{for } 1 \leqslant i \leqslant n,\ u_i \in t_i(d\colon T_d \mid d \in \Delta_0)\}.$$

If each T_d is a singleton set, i.e., $T_d = \{t_d\}$ for some tree t_d, we abuse the above notation by writing simply $t(d\colon t_d \mid d \in \Delta_0)$.

EXAMPLE. If $t = g[f[b], c, g[d, b, d]]$, $\Delta_0 = \{b, c\}$, and $t_b = h[a, a]$, $t_c = h[b, b]$, then $t(s\colon t_s \mid s \in \Delta_0) = g[f[h[a, a]], h[b, b], g[d, h[a, a], d]]$. If $T_b = \{h[a, b], h[a, a]\}$ and $T_c = \{h[b, b]\}$ then $t(s\colon T_s \mid s \in \Delta_0)$ is the set containing the four trees:

$$g[f[h[a, b]], h[b, b], g[d, h[a, b], d]],$$
$$g[f[h[a, a]], h(b, b], g(d, h[a, b], d]],$$
$$g[f[h[a, b]], h(b, b], g[d, h[a, a], d]],$$
$$g[f[h[a, a]], h[b, b], g[d, h[a, a], d]].$$

As described above, the output tree generated in a step of a computation of a top-down tree transducer contains markers which specify which sons of the current input node to read next and which states to continue in. The sons are specified by means of a special infinite set of symbols $X = \{x_i \mid i = 1, 2, \ldots\}$, where $x_i \neq x_j$ for $i \neq j$. Thus, x_i refers to the ith son of the current input node. If a tree transducer has a set of states Q, then by convention $Q \times X$ will be a set of symbols of rank 0. For each $n > 0$, let $X_n = \{x_1, \ldots, x_n\}$, and let $X_0 = \varnothing$.

DEFINITION. A (nondeterministic) top-down tree transducer is a 5-tuple $M = (Q, \Sigma, \Delta, R, Q_0)$ where

(1) Q is a finite set of *states*,

(2) Σ is a finite ranked alphabet called the *input alphabet*,

(3) Δ is a finite ranked alphabet called the *output alphabet*,

(4) $Q_0 \subseteq Q$ is a set of *starting states*, and

(5) R is a finite set of *rules*,

$$R \subseteq \bigcup_{n \geqslant 0} (Q \times \Sigma_n) \times (\Delta \cup (Q \times X_n))_*.$$

A rule is written in the form $(q, b) \to w$, *where* $q \in Q$, $b \in \Sigma_n$, and $w \in (\Delta \cup (Q \times X_n))_*$ for some n.

The behavior of a top-down transducer is defined inductively in terms of the output produced from a tree starting in state q.

DEFINITION. Let $M = (Q, \Sigma, \Delta, R, Q_0)$ be a top-down tree transducer, and let $q \in Q$. For a tree $t \in \Sigma_*$, the set of trees output from t by M starting in state q is denoted by $M(q, t)$ and is defined inductively as follows.

(1) If $t = b \in \Sigma_0$, then $M(q, t) = \{w \mid (q, b) \to w \in R\}$;

(2) If $t = b[t_1, \ldots, t_n] \in \Sigma_*$, then

$$M(q, t) = \bigcup_{(q,b)\to w \in R} w(\langle p, x_j\rangle : M(p, t_j) \mid p \in Q, 1 \leqslant j \leqslant n).$$

We illustrate top-down transducers in the following examples.

EXAMPLE. Let $\Sigma = \{b, c\}$ be a ranked alphabet, in which b has rank 2 and c has rank 0. Let $\Delta = \{f, g, d\}$ be a ranked alphabet in which f and g have rank 2 and d has rank 0. Then $M = (\{q_0\}, \Sigma, \Delta, R, \{q_0\})$ is a nondeterministic top-down tree transducer, where

$$R = \{(q_0, b) \to f[\langle q_0, x_1\rangle, \langle q_0, x_1\rangle], (q_0, b) \to g[\langle q_0, x_1\rangle, \langle q_0, x_1\rangle], (q_0, c) \to d\}.$$

At each node labeled b, M generates either an f or a g and starts two computations on the left subtree. Therefore, if s is a tree in Σ_* such that the path from the root to the leftmost leaf has exactly k nodes, then $M(q_0, s)$ is the set of all balanced binary trees in Δ^* of depth k.

EXAMPLE. An arithmetic expression involving addition, multiplication, a constant c, and a variable y may be represented by a tree over the alphabet $\Sigma = \{+, *, y, c\}$, where $+$ and $*$ have rank 2 and y and c have rank 0. We construct a deterministic top-down transducer M which takes the formal derivative with respect to y of the expression represented by an input tree in Σ_*. Let $\Delta = \Sigma \cup \{1, 0\}$, where 1 and 0 have rank 0. Let $M = (\{D, I\}, \Sigma, \Delta, R, \{D\})$ be a top-down transducer, where R contains the following rules:

$$\begin{aligned}
&(D, +) \to +[\langle D, x_1\rangle, \langle D, x_2\rangle],\\
&(D, *) \to +[*[\langle D, x_1\rangle, \langle I, x_2\rangle], *[\langle I, x_1\rangle, \langle D, x_2\rangle]],\\
&(D, y) \to I, \quad (D, c) \to 0, \quad (I, \sigma) \to \sigma \quad \text{for} \quad \sigma \in \{y, c\}, \quad \text{and}\\
&(I, \sigma) \to \sigma[\langle I, x_1\rangle, \langle I, x_2\rangle] \quad \text{for} \quad \sigma \in \{+, *\}.
\end{aligned}$$

For $t = +[*[c, y], y]$, $M(t) = +[+[*[0, y], *[c, 1]], 1]$. In general, for a tree $t \in \Sigma_*$, $M(I, t) = \{t\}$ and $M(D, t)$ is the singleton set containing the tree representing the formal derivative of the expression represented by t.

Next, we describe the behavior of bottom-up tree transducers. A bottom-up tree transducer reads an input tree t by starting at the leaves and working upward toward the root of t. At each leaf α, the label of α determines the possible output trees and states to be entered; the transducer outputs a tree and enters some state which is then associated with α. When all the sons of a node β of rank greater than 0 have been read, the label of β and the states associated with the sons of β determine an output tree u and a state q; the transducer substitutes the output trees produced from the subtrees of β at specially marked leaves of u

and state q becomes associated with β. The output generated from the ith subtree of β may be substituted at 0, 1, or more leaves of the tree output at β. The computation ends when the transducer has read the root of the input tree. It is an accepting computation if the transducer has entered a final state at the root.

Unlike a top-down transducer, a bottom-up transducer reads each input node exactly once. Instead of starting several computations on a subtree and using the output from each exactly once, the bottom-up transducer generates one output tree from an input subtree and makes copies of it. As the transducer can make several copies of output trees at each input node as it works its way up toward the root, the size of the output tree may grow exponentially with the size of the input tree.

DEFINITION. A (nondeterministic) bottom-up tree transducer is a 5-tuple $M = (Q, \Sigma, \Delta, R, F)$, where

(1) Q is a finite set of *states*,

(2) Σ is a finite ranked alphabet called the *input alphabet*,

(3) Δ is a finite ranked alphabet called the *output alphabet*,

(4) $F \subseteq Q$ is a set of *accepting* or *final* states,

(5) R is a finite set of *rules*,

$$R \subseteq \bigcup_{n \geqslant 0} (\Sigma_n \times Q^n) \times (Q \times (\Delta \cup X_n)_*).$$

A rule is written in the form $(b, q_1, \ldots, q_n) \to (q, t)$, where $b \in \Sigma_n$, $n \geqslant 0$, $q, q_1, \ldots, q_n \in Q$, and $t \in (\Delta \cup X_n)_*$.

The behavior of a bottom-up transducer on an input tree is defined inductively as follows.

DEFINITION. Let $M = (Q, \Sigma, \Delta, R, F)$ be a bottom-up tree transducer, and let $q \in Q$. For a tree $t \in \Sigma_*$, the set of trees which M can output from t ending in state q is denoted by $M(q, t)$ and is defined inductively as follows:

(1) For $b \in \Sigma_0$, $M(q, b) = \{w \mid b \to (q, w) \in R\}$,

(2) For $t = b[t_1, \ldots, t_n] \in \Sigma_*$,

$$M(q, t) = \{v \mid \text{for some rule } (b, q_1, \ldots, q_n) \to (q, w) \in R, \text{ some } i,\ 1 \leqslant i \leqslant n, \text{ and some } u_i \in M(q_i, t_i),\ v \in W(x_i : u_i)\}$$

Note that in the second part of the above definition, the same tree u_i in $M(q_i, t_i)$ is substituted at every occurrence of x_i in w. If $M(q_i, t_i) = \varnothing$, then no output can be generated from $b[t_1, \ldots, t_n]$ even if x_i does not occur in the right side of the rule $(b, q_1, \ldots, q_n) \to (q, w)$.

EXAMPLE. Let $\Sigma = \{b, c\}$ be a ranked alphabet with b of rank 2 and c of rank 0. Let $\Delta = \{f, g, d\}$ be a ranked alphabet in which f and g have rank 2 and d has rank 0. Let $M = (\{q_0\}, \Sigma, \Delta, R, \{q_0\})$ be a bottom-up transducer, where

$$R = \{c \to (q_0, d), (b, q_0, q_0) \to (q_0, f[x_1, x_1]), (b, q_0, q_0) \to g[x_1, x_1]\}.$$

Thus, at each node labeled b, M generates either an f or a g and makes two copies of the subtree it has already generated from the left input subtree. If s is a tree in Σ_* such that the path from the root to the leftmost leaf has exactly k nodes, then

$$M(s) = \{t \in \Delta_* \mid \text{every path from the root to a leaf in } t \text{ has exactly } k \text{ nodes and the same sequence of labels}\}.$$

Also, $Y(M(s)) = \{d^{2^{k-1}}\}$.

DEFINITION. Let $M = (Q, \Sigma, \Delta, R, P)$ be a top-down or bottom-up tree transducer. For a tree t, the set of all trees output from t by M is $M(t) = \bigcup_{p \in P} M(p, t)$. For a set T of trees, the set of trees generated from T by M is $M(T) = \bigcup_{t \in T} M(t)$. The *transduction* performed by M is a relation consisting of input-output pairs of M, where $\perp$ denotes the lack of any output:

$$T(M) = \{\langle s, t\rangle \in \Sigma_* \times \Delta_* \mid t \in M(s)\} \cup \{\langle s, \perp\rangle \mid s \in \Sigma_*, M(s) = \varnothing\}.$$

The special symbol $\perp$ is used so that $T(M)$ specifies all input trees, including the ones without any output.

Since tree transductions are relations, they may be composed. Thus,

$$T(M_2) \circ T(M_1) = \{\langle s, u\rangle \mid \text{for some } t, \langle s, t\rangle \in T(M_1) \text{ and } \langle t, u\rangle \in T(M_2)\}.$$

If B and C are classes of transductions, $B \circ C = \{T_B \circ T_C \mid T_C \in C \text{ and } T_B \in B\}$. Also, $B^1 = B$, and for $n \geqslant 1$, $B^{n+1} = B \circ B^n$.

Several restrictions on tree transducers are important in this paper. A top-down or bottom-up transducer M is *linear* if no variable x_j occurs more than once in the right side of any rule of M. M is *full* or *nondeleting* if for each rule of M, if n is the rank of the input symbol in the left side, then the right side has at least one occurrence of each x_j, $1 \leqslant j \leqslant n$. A top-down transducer $M = (Q, \Sigma, \Delta, R, Q_0)$ is *deterministic* if Q_0 is a singleton set and for each $b \in \Sigma$ and each $q \in Q$, there is exactly one rule with left side (q, b). A bottom-up transducer $M = (Q, \Sigma, \Delta, R, F)$ is *deterministic* if for each $n \geqslant 0$, each $b \in \Sigma_n$, and each $q_1, \ldots, q_n \in Q$, there is exactly one rule with left side $(b, q_1, \ldots, q_n)$.

The restrictions on tree transducers are abbreviated as follows: T, top-down; B, bottom-up; N, nondeterministic; D, deterministic; L, linear; F, full (nondeleting) and O, one-state. For example, NT in the class of nondeterministic top-down transductions, and $FDLT$ is the class of full deterministic linear top-

down transductions. Since $DOT = DOB$ (Engelfriet, 1975) this class is generally written as DO. Note that NT^n and NB^n denote the composition of n nondeterministic top-down transductions and n nondeterministic bottom-up transductions, respectively.

A deterministic bottom-up transducer $M = (Q, \Sigma, \Sigma, R, F)$ is a *deterministic bottom-up finite tree automaton* if each rule is of the form $b \to (q, b)$ where $b \in \Sigma_0$, $q \in Q$ or of the form $(b, q_1, \ldots, q_n) \to (q, b[x_1, \ldots, x_n])$, where $b \in \Sigma_n$ and $q, q_1, \ldots, q_n \in Q$. A set of trees T is *recognizable* if $T = \{s \mid M(s) \neq \varnothing\}$ for some deterministic bottom-up finite tree automaton M.

We conclude this section by defining a hierarchy of families of tree languages and showing how it is generated by top-down and bottom-up transductions.

DEFINITION. Let D_0 denote the family of recognizable sets. For $n \geqslant 0$, let $U_{n+1} = DO(D_n)$, and let $D_n = NLT(U_n)$.

Since the identity transduction is a total deterministic one-state linear top-down transduction, we have defined a hierarchy of families of tree languages: $D_0 \subseteq U_1 \subseteq D_1 \subseteq U_2 \subseteq \cdots$. We will refer to this hierarchy as the D-U hierarchy. Another characterization of the D-U hierarchy is given in Theorem 1.

THEOREM 1. *For $n > 0$, $D_n = NT^n(D_0)$. For $n > 0$, $U_n = NB^n(D_0)$.*

Proof. Baker (submitted for publication) and Engelfriet (1975) showed that $NT = NLT \circ DO$ while $NB = DO \circ NLT$. Therefore, for $n > 0$, $D_n = NLT(DO(D_{n-1})) = (NLT \circ DO)(D_{n-1}) = NT(D_{n-1})$. The first part of the theorem is obtained by induction on n.

D_0 is closed under linear top-down transductions (Thatcher, 1973). Therefore $U_1 = DO(D_0) = DO(NLT(D_0)) = (DO \circ NLT)(D_0) = NB(D_0)$. For $n > 1$, $U_n = DO(D_{n-1}) = DO(NLT(U_{n-1})) = (DO \circ NLT)(U_{n-1}) = NB(U_{n-1})$. The second part of the theorem is obtained by induction on n.

This theorem is a generalization of Engelfriet's result that $U_1 = DO(D_0)$ (Engelfriet, 1975).

Thus, the D_n families in the D-U hierarchy are obtained by starting with the recognizable sets and applying successive top-down transductions. Similarly, the U_n families in the D-U hierarchy are obtained by starting with the recognizable sets and applying successive bottom-up transductions. The interesting point about the above result is that the families generated by top-down transductions and the families generated by bottom-up transductions alternate in a *single* hierarchy. The top-down families D_n have been studied previously by Ogden and Rounds (1972), who conjecture that for every n, $D_n \subsetneq D_{n+1}$. They were able to prove only that $D_0 \subsetneq D_1 \subsetneq D_2$. Engelfriet (1975) showed that $D_0 \subsetneq U_1 \subsetneq D_1$. For the D-U hierarchy, we make the following conjecture.

Conjecture. For $n \geqslant 0$, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$.

A second hierarchy, called the $Y(D) - Y(U)$ hierarchy, may be obtained

from the D-U hierarchy by the yield operation: $Y(D_0) \subseteq Y(U_1) \subseteq Y(D_1) \subseteq \cdots$. We conjecture that each inclusion is proper for this hierarchy as well.

Conjecture. For $n \geq 0$, $Y(D_n) \subsetneq Y(U_{n+1}) \subsetneq Y(D_{n+1})$.

Although the conjectures remain open, this paper proves some results concerning the families in these hierarchies. In particular, the closure properties of these families under various operations are studied in Sections 2 and 3, and some results bearing on the conjectures are presented in Section 4.

Section 2

Many families of string languages which have been extensively studied (such as the families of context-free languages, context-sensitive languages, recursive sets, and regular sets) are closed under certain string operations such as homomorphism, concatenation, substitution, or intersection with regular sets. It is natural to wonder whether there are any operations on trees which preserve membership in the families in the D-U hierarchy. Some natural operations to consider are tree substitution, intersection with recognizable sets, and various subclasses of tree transductions. In fact, the families in the D-U hierarchy are closed under some of these operations.

It is trivial to show that each family in the D-U hierarchy is closed under union. Several additional closure results may be obtained by applying information about the closure of certain classes of transducers under composition to Theorem 1. Since $DB \circ NB = NB$ (Engelfriet, 1975) and for $n > 0$, $U_n = NB^n(D_0)$ (Theorem 1), we have the following proposition.

Proposition 2. *For $n > 0$, U_n is closed under deterministic bottom-up transductions.*

Engelfriet (1975) showed that the class of nondeterministic linear bottom-up transductions is closed under composition, and that $NLT \circ NLT = NLB$. Baker (submitted for publication) showed that $NT^n = NLB \circ NT^n$ for $n > 1$. By (Thatcher, 1973 and Rounds, 1970), D_0 and D_1 are closed under linear top-down transductions. The above observations and the fact that each $D_n = NT^n(D_0)$ yield closure for each D_n, $n > 1$ under linear transductions.

Proposition 3. *For $n \geq 0$, D_n is closed under nondeterministic linear top-down and bottom-up transductions.*

From the above propositions and the definition of each U_{n+1} as $DO(D_n)$ and D_n as $NLT(U_n)$, the relationship between the top-down and bottom-up families in the hierarchy may be described as follows.

Proposition 4. *For every $n \geq 0$, U_{n+1} is the closure of D_n under deterministic one-state transductions and also under deterministic bottom-up transductions. For*

every $n > 0$, D_n is the closure of U_n under nondeterministic linear top-down transductions and under nondeterministic linear bottom-up transductions.

Ogden and Rounds (1972) have shown that each family D_n is closed under intersection with recognizable sets. Engelfriet (1975) showed that U_1 is closed under intersection with recognizable sets. Thus, the following result is not unexpected.

PROPOSITION 5. *For $n > 0$, U_n is closed under intersection with recognizable sets.*

Proof. If R is a recognizable set, R is the domain of a deterministic bottom-up transducer M such that $T(M) = \{\langle t, t\rangle \mid t \in R\}$. For a set $T \in U_n$, $M(T) = T \cap R$ is in U_n since U_n is closed under deterministic bottom-up transductions (Proposition 2). ∎

Next, we consider closure under a restricted form of substitution called tree concatenation.

Notation. In Section 1, the notation $t(d\colon W_d \mid d \in \Delta_0)$ was introduced to represent the set of trees obtained by substituting trees in W_d at all leaves in t labeled with d. For $b \in \Delta_0$, let $t(b : W)$ denote the set of trees obtained by substituting trees in W for all leaves in t labeled b. For a set of trees T, let $T(b : W) = \{t(b : W) \mid t \in T\}$.

DEFINITION. Let F be a family of tree languages. F is closed under tree concatenation if for every T_1, $T_2 \in F$, $T_1 \subseteq \Sigma_*$, and every $b \in \Sigma_0$, $T_1(b : T_2) \in F$.

THEOREM 6. *For every $n \geq 0$, D_n is closed under tree concatenation.*

Proof. Thatcher and Wright (1968) showed that D_0 is closed under tree concatenation. For every n, D_n contains the recognizable sets and is closed under linear top-down transductions (Proposition 3). We show that for any family F which contains the recognizable sets and is closed under tree concatenation and deterministic linear one-state transductions, $NT(F)$ is also closed under tree concatenation. The theorem follows by induction on n.

Let F be a family of tree languages which contains the recognizable sets and is closed under tree concatenation and deterministic linear one-state transductions. Let Z, $W \in NT(F)$ with Z, $W \subseteq \Sigma_*$, and $b \in \Sigma_0$. We show that $Z(b : W) \in NT(F)$ by constructing a set V in F and a nondeterministic top-down transducer N such that $N(V) = Z(b : W)$.

First, consider how the sets Z and W are obtained from sets in F by top-down transductions. There exist a ranked alphabet Δ, a tree language $S \in F$ with $S \subseteq \Delta_*$, and a top-down transducer $M = (Q, \Delta, \Sigma, P, \{q_0\})$ such that $M(S) = Z$. Similarly, there exist a ranked alphabet Γ, a tree language $S_b \in F$ with $S_b \subseteq \Gamma_*$,

and a top-down transducer $M_b = (K_b, \Gamma, \Sigma, P_b, \{k_b\})$ such that $M_b(S_b) = W$. Without loss of generality, we may assume that the sets Σ, Δ, and Γ are all pairwise disjoint.

Each tree v in V will be obtained from a tree $s \in S$. The intent of the construction is that the new transducer N produces from v what M would produce from s, except that whenever M outputs b, v provides N with access to a tree in S_b, and N imitates M_b to output a tree in W rather than b. The construction is complicated by the fact that for arbitrarily large s, M may be able to output arbitrarily many b's from a single node of s, and a distinct tree in W may be substituted at each occurrence of b in M's output. Therefore, v is constructed as follows: We obtain from S_b a set U consisting of trees of the form $g[s_1, g[s_2, \ldots, g[s_m, s_{m+1}] \cdots]]$, where $m \geqslant 1$ and $s_1, \ldots, s_{m+1} \in S_b$. Then we nondeterministically insert a tree in U at each node of s to obtain v.

Formally, the construction of V has three stages:

(1) Let f be a new symbol of rank 0, and let g be a new symbol of rank 2. Let R be the set of trees defined inductively as follows: $f \in R$, and for any tree $t \in G$, $g[f, t] \in R$. Clearly, R is a recognizable set, and is therefore in F.

Set $U = R(f : S_b)$. Since F is closed under tree concatenation, U is in F.

Note that each tree in U is of the form $g(t_1, g[t_2, g[t_3, \ldots, g[t_n, t_{n+1}] \cdots]]]$ where $n \geqslant 1$ and $t_1, \ldots, t_{n+1} \in S_b$. Also, for every $n \geqslant 1$ and $t_1, t_2, \ldots, t_{n+1} \in S_b$, $g[t_1, g[t_2, \ldots, g[t_n, t_{n+1}] \cdots]]] \in U$.

(2) For each symbol $c \in \Delta_n$, $n \geqslant 0$, let $\bar{c}$ be a new symbol with rank $n + 1$. Let $\bar{\Delta} = \{\bar{c} \mid c \in \Delta\} \cup \{f\}$. Let $D = (\{p\}, \Delta, \bar{\Delta}, R_D, \{p\})$ be a one-state deterministic linear top-down transducer, where

$$R_D = \{(p, c) \to \bar{c}[f] \mid c \in \Delta_0\} \cup \{(p, c) \to \bar{c}[f, (p, x_1), \ldots, (p, x_n)] \mid n > 0, c \in \Delta_n\}.$$

Clearly, at each node labelled $c \in \Delta$, D simply relabels it $\bar{c}$ and adds a new subtree f to the left of all previous subtrees of the node. Set $S' = D(S)$. Since F is closed under deterministic linear one-state top-down transductions, $S' \in F$.

(3) Set $V = S'(f : U)$. Since S', $U \in F$ and F is closed under tree concatenation, $V \in F$.

Next, we define a top-down transducer N such that $N(V) = z(b : w)$. Intuitively, we want N to imitate M on input symbols in $\bar{\Delta}$, except that whenever M would generate an output symbol b, N begins to scan one of the subtrees in R. Now, this tree in U has two or more subtrees which are in S_b, and N nondeterministically selects one of these, and then imitates M_b on this subtree. Thus, whenever M would output a b, N instead generates a tree in W.

Formally, we let $N = (Q \cup K_b, \bar{\Delta} \cup \Gamma \cup \{g\}, \Sigma, R_N, \{q_0\})$, where q_0 is the starting state of M, and R_N is given as follows: Let

$$h: (\Delta \cup \Sigma \cup X \cup Q \cup \Pi)^* \to (\bar{\Delta} \cup \Sigma \cup X \cup Q \cup \Pi \cup \{k_b\})^*$$

be the homomorphism determined by $h(c) = \bar{c}$ for $c \in \Delta$, $h(b) = (k_b, x_1)$, $h(x_n) = x_{n+1}$ for $n \geqslant 1$, and $h(a) = a$ otherwise. Then

$$R_N = P_B \cup \{h(u) \to h(v) \mid u \to v \in P\} \cup \{(k_b, g) \to (k_b, x_1), (k_b, g) \to (k_b, x_2)\}.$$

Let us examine how N behaves on an input tree v in V. For some $s \in S'$, $v \in \rho(s)$. Thus, v is obtained by substituting trees in U at leaves of s labeled f. By applying rules of the form $h(u) \to h(v)$, where $u \to v \in P$, N imitates M on s; thus the "upper" portion of the output corresponds to a tree u in $M(s)$. However, whenever M would output a leaf labeled b, N instead enters the state k_b (the starting state of M_b) and reads the leftmost subtree of its input node (since $h(b) = (k_b, x_1)$). This subtree must be of the form $g[s_1, g(s_2, ..., g[s_{m-1}, s_m] \cdots]]$ where $m \geqslant 2$ and $s_1, ..., s_m \in S_b$. N applies the rule $(k_b, g) \to (k_b, x_2)$ 0 or more times, and finally either reaches s_m or applies the rule $(k_b, g) \to (k_b, x_1)$ and begins to read s_i, for some $i < m$. At this point, N imitates M_b on this subtree in S_b to generate a subtree in W. The result of this process is that M outputs from v a tree in $M(s)(b : W)$.

Next, we describe for an arbitrary tree u in $Z(b : W)$ how to find a tree v in V and a computation of N which generates u from v. Suppose that s is in S, and t is in $M(s) \subseteq Z$, and $u \in t(b : W)$. Suppose that t contains exactly $m \geqslant 0$ occurrences of the symbol b. For some $t_1, ..., t_m \in W$ and some $z_1, z_2, ..., z_{m+1}$ in $(\Sigma \cup \Pi)^*$,

$$t = z_1 b z_2 b \cdots z_m b z_{m+1}$$

and

$$u = z_1 t_1 z_2 t_2 \cdots z_m t_m z_{m+1}.$$

Choose trees $s_1, s_2, ..., s_m$ such that for $j = 1, 2, ..., m$, $t_i \in M_b(s_i)$. Note that there is a tree $y = g[s_1, g[s_2, ..., g[s_m, s_m) \cdots]]$ in U. Further, beginning in state k_b, N can generate from y any tree in $M_b(s_1)$, $M_b(s_2)$, ..., $M_b(s_m)$, including $t_1, ..., t_m$.

Let $s' = D(s)$. Consider a tree v in $s'(f : U) \subseteq V$ obtained by substituting y at each leaf of s labeled f. There is a computation of N on v which

(1) imitates M on the nodes of v labeled from $\bar{\Delta}$ to produce a tree z, such that z is the tree obtained from t by substituting the tree $(k_b, g[s_1, g[s_2, ..., g[s_m, s_m] \cdots]])$ at each leaf of t labeled b, and

(2) then produces t_i from the ith occurrence (from the left) of $(k, g[s_1, g[s_2, ..., g[s_m, s_m]])$, for $i = 1, 2, ..., m$. The result of this computation is the tree obtained from t by substituting t_i for the ith occurrence of b in t, for $i = 1, 2, ..., m$. But this tree is precisely u. Thus, we have found a tree v in V and a computation of N on v which produces output u.

From the above arguments, it is clear that $N(v) = Z(b : W)$. ∎

Next, we consider an operation closely related to tree concatenation.

DEFINITION. Let F_1 and F_2 be families of tree languages. Define

$$\mathrm{Sub}_T(F_1, F_2) = \{T(b : W_b \mid b \in \Sigma_0) \mid T \in F_2,\ T \subseteq \Sigma_*, \text{ and for each } b \in \Sigma_0,\ W_b \in F_1)\}.$$

F_1 is *closed under tree substitution* if $\mathrm{Sub}_T(F_1, F_1) \subseteq F_1$. F_1 is *closed under tree substitution into* F_2 if $\mathrm{Sub}_T(F_1, F_2) \subseteq F_1$.

Since tree concatenation is a restricted form of tree substitution, it is natural to investigate whether D_n is also closed under tree substitution.

THEOREM 7. *For every $n \geqslant 0$, D_n is closed under tree substitution.*

Proof. Suppose $T \in D_n$, $T \subseteq \Sigma_*$, and for every $b \in \Sigma_0$, $W_b \in D_n$. We show that $T(b : W_b \mid b \in \Sigma_0) \in D_n$. For each $b \in \Sigma_0$, let $\hat{b}$ be a new symbol of rank 0 (in particular, $\hat{b}$ does not appear in any W_c). Since D_n contains the singleton sets of trees and is closed under tree concatenation, $T' = T(b : \hat{b} \mid b \in \Sigma_0) = T(b_1 : \hat{b}_1) \cdots (b_m : \hat{b}_m)$ is in D_n, where $\Sigma_0 = \{b_1, \ldots, b_m\}$ and $b_i \neq b_j$ for $i \neq j$. Moreover, $T'(\hat{b}_1 : W_{b_1}) \cdots (\hat{b}_m : W_{b_m}) = T(b : W_b \mid b \in \Sigma_0) \in D_n$. ∎

An important factor in the proof that D_n is closed under tree concatenation is the ability of a nondeterministic top-down transducer N to generate different trees in W from the same input subtree and substitute them at different occurrences of the symbol b. In a single computation, a bottom-up transducer cannot output two distinct trees in W from the same input subtree. In Section 4, it will be shown that the class of bottom-up transducers does not preserve closure under tree concatenation; in fact, U_1 is not closed under tree concatenation. However, the class of bottom-up transducers does preserve closure of tree substitution into the recognizable sets.

THEOREM 8. *For every $n > 0$, U_n is closed under tree substitution into the recognizable sets.*

Proof. By Theorem 7, D_0 is closed under tree substitution. We show that for each n, if U_n is closed under tree substitution into the recognizable sets, U_{n+1} is also closed under tree substitution into the recognizable sets. The theorem follows by induction on n.

Let R be a recognizable set with $R \subseteq \Sigma_*$, and for each $b \in \Sigma_0$, let $W_b \in U_{n+1}$, $W_b \subseteq \Sigma_*$. We show that $R(b : W_b \mid b \in \Sigma_0) \in U_{n+1}$ by obtaining a set $V \in U_n$ and a bottom-up transducer N such that $N(V) = R(b : W_b \mid b \in \Sigma_0)$.

For each $b \in \Sigma_0$, there exist a ranked alphabet Γ_b, a tree language $S_b \subseteq (\Gamma_b)_*$, $S_b \in U_n$, and a bottom-up transducer $M_b = (Q_b, \Gamma_b, \Sigma, P_b, F_b)$ such that $M_b(S_b) = W_b$. Let $\Gamma = \bigcup_{b \in \Sigma_0} \Gamma_b$. Without loss of generality we may assume

that the sets Γ_b, Q_b, and Σ are pairwise disjoint. (If not, each S_b may be relabeled in a new alphabet by a deterministic linear bottom-up transducer, and corresponding changes may be made in each M_b. Since each U_n is closed under deterministic bottom-up transductions, each relabeled S_b is in U_n.)

Set $V = R(b : S_b \mid b \in \Sigma_0)$. Since F is closed under tree substitution into recognizable sets, $V \in F$.

Let f be a new state, and let $Q' = \{f\} \cup \bigcup_{b\in\Sigma_0} Q_b$, $F' = \{f\} \cup \bigcup_{b\in\Sigma_0} F_b$. Construct a bottom-up transducer $N = (Q', \hat{\Sigma} \cup \Gamma, \Sigma, P_N, F)$ by setting

$$P_N = \{(c, f_1, \ldots, f_n) \to (f, c[x_1, \ldots, x_n]) \mid n \geqslant 1, c \in \Sigma_n \text{ and } f_1, \ldots, f_n \in F'\} \cup \bigcup_{b\in\Sigma_0} P_b .$$

When N reads a tree $t \in V$, it imitates M_b on subtrees of t which are in S_b, and is the identity on symbols in Σ. We show that $N(V) = R(b : W_b \mid b \in \Sigma_0)$ by giving a proof by induction on the depth of r that for every $r \in \Sigma_*$, $t \in r(b : W_b \mid b \in \Sigma_0)$ if and only if $t \in N(r(b : S_b \mid b \in \Sigma_0))$.

For $r \in \Sigma_0$, $t \in r(b : W_b \mid b \in \Sigma_0)$ if and only if $t \in W_r$ if and only if $t \in M_b(S_r)$ if and only if $t \in N(r(b : S_b \mid b \in \Sigma_0))$. For some $k \geqslant 1$, assume that the assertion holds whenever r has depth at most k. Suppose $r = B[r_1, \ldots, r_n] \in \Sigma_*$ has depth $k + 1$, and $t \in r(b : W_b \mid b \in \Sigma_0)$. There exist trees $t_1, \ldots, t_n \in \Sigma_*$ such that $t = B[t_1, \ldots, t_n]$ and for each i, $t_i \in r_i(b : W_b \mid b \in \Sigma_0)$. The induction hypothesis may be applied to each r_i to show that for each i, there are a final state $f_i \in F'$ and a tree $u_i \in r_i(b : W_b \mid b \in \Sigma_0)$ such that $t_i \in N(f_i, u_i)$. Since N has a rule $(B, f_1, \ldots, f_n) \to (f, B[x_1, \ldots, x_n])$, $t = B[t_1, \ldots, t_n] \in N(B[u_1, \ldots, u_n]) \subseteq N(r(b : W_b \mid b \in \Sigma_0))$. This argument is easily reversed for the other direction of the assertion. ∎

We conclude this section by observing that the union of all the families in the D-U hierarchy is closed under all the operations studied above.

THEOREM 9. *$\bigcup_{n=0}^{\infty} D_n = \bigcup_{n=0}^{\infty} U_n$ is closed under bottom-up and top-down transductions, tree substitution, and intersection with recognizable sets.*

Proof. Obviously, for any language T in $\bigcup_{n=0}^{\infty} D_n$, the image of T under a top-down or bottom-up transduction is in D_n, for some n, and thus is in $\bigcup_{n=0}^{\infty} D_n$. Also, since every language in $\bigcup_{n=0}^{\infty} D_n$ is in D_n for some n, and D_n is closed under both tree substitution and intersection with recognizable sets, $\bigcup_{n=0}^{\infty} D_n$ is closed under tree substitution and intersection with recognizable sets. ∎

SECTION 3

A natural question to ask about the $Y(D) - Y(U)$ hierarchy is whether the families in it are closed under the string operations which have been commonly

studied in formal language theory. In this section, we show that they are closed under a number of string operations. However, since $Y(D_n)$ and $Y(U_n)$ cannot contain the empty string (denoted by e), they cannot be closed under operations such as homomorphism which generate the empty string. Therefore, we define an extended yield operation Y_e on families of languages, defined by

$$Y_e(F) = \{T, T \cup \{e\} \mid T \in F\}$$

for each family of tree languages F. The closure properties of each $Y(D_n)$, $Y_e(D_n)$, $Y(U_n)$, and $Y_e(U_n)$ presented in this section are all derived from the closure properties of D_n and U_n studied in Section 2.

Rounds (1970) showed that for any regular set R, $R \subseteq \Sigma_0^*$, $Y_\Sigma^{-1}(R) = \{t \in \Sigma_* \mid Y(t) \in R\}$ is a recognizable set. Ogden and Rounds (1972) applied this fact and the closure of each D_n under intersection with recognizable sets to show that for $T \in D_n$, $T \subseteq \Sigma_*$, $n \geqslant 0$, $Y(T) \cap R = Y(T \cap Y_\Sigma^{-1}(R))$ is in $Y(D_n)$. That is, each $Y(D_n)$ is closed under intersection with regular sets. Obviously, each $Y_e(D_n)$ is also closed under intersection with regular sets. The same argument may be applied to each U_n to obtain the following proposition.

PROPOSITION 10. *For $n > 0$, $Y(U_n)$ and $Y_e(U_n)$ are closed under intersection with regular sets.*

The above result cannot be strengthened to state that $Y(D_n)$ and $Y(U_n)$ are closed under intersection, since the intersection of the context-free languages is not contained in $\bigcup_{n=0}^{\infty} Y(D_n)$ (Baker, to appear).

LEMMA 11. *For every $n \geqslant 0$, $Y_e(D_n)$ and $Y_e(U_n)$ are closed under string homomorphism.*

Proof. Let $h: \Sigma_0^* \to \Delta_0^*$ be a string homomorphism, and $T \subseteq \Sigma_*$. Since each D_n and U_n is closed under deterministic linear transductions by Propositions 2 and 3, it suffices to construct a deterministic linear bottom-up tree transducer M such that $Y(M(T)) = h(Y(T)) - \{e\}$.

If h is nonerasing, it may be performed by a deterministic linear bottom-up transducer which simply replaces each $b \in \Sigma_0$ by a subtree whose yield is $h(b)$. If h can erase, the transducer may do this replacement when $h(b) \neq e$, but it must also delete every input tree whose leaves are all erased by h.

Let m be the maximum rank of any symbol in Δ. Let $\Gamma = \{g_i \mid 0 \leqslant i \leqslant m\}$, where g_i has rank i. Construct a deterministic bottom-up transducer $M = (K, \Sigma, \Gamma, X, P, \{N\})$ where $K = \{E, N\}$, and P is given as follows.

(1) For $b \in \Sigma_0$, if $h(b) = e$, then the rule $b \to (E, g_0)$ is in P. If $h(b) = a_1 a_2 \cdots a_n \neq e$, each $a_i \in \Delta_0$, then the rule $b \to (N, g_n[a_1, a_2, \ldots, a_n])$ is in P.

(2) Suppose that $b \in \Sigma_m$, $m \geqslant 1$, and $S_1, S_2, \ldots, S_m \in K$. If $S_1 = S_2 = \cdots = S_m = E$, then the rule $(b, S_1, \ldots, S_m) \to (E, g_0)$ is in P. If $1 \leqslant k \leqslant m$,

$1 \leqslant j_1 < j_2 < \cdots < j_k \leqslant m$, and $S_{j_1} = S_{j_2} = \cdots = S_{j_k} = N$, while $S_i = E$ for $i \notin \{j_1, j_2, \ldots, j_k\}$, then the rule $(b, S_1, \ldots, S_m) \to (N, g_k[x_{j_1}, x_{j_2}, \ldots, x_{j_k}])$ is in P.

M starts at the leaves of an input tree t and determines which leaves are erased by the homomorphism h. At any point during M's computation when M has read all of a subtree t_1 of t, M has reached state E if every leaf of this subtree is to be erased, and state N otherwise. If M has reached state N, then the output from the subtree t_1 is a tree whose yield is $h(\text{yield}(t_1)) \neq e$. If M has reached state E, then $h(\text{yield}(t_1)) = e$, and the output from the subtree t_1 will be deleted in the next step. Since the only final state is N, M produces a tree as output from t only if not all the leaves are erased by the homomorphism h. Thus, $\text{yield}(M(T)) = h(\text{yield}(T)) - \{e\}$. ∎

The next operation to be considered is string substitution, which is defined as follows.

DEFINITION. Let F be a family of string languages, and Σ an alphabet. An F-(string) substitution is a function $\tau: \Sigma \to F$. It is extended to Σ^* by

(1) $\tau(e) = \{e\}$:

(2) for $a_1, \ldots, a_n \in \Sigma$, $\tau(a_1 \cdots a_n) = \tau(a_1) \cdots \tau(a_n)$. For families F_1 and F_2, define

$$\text{Sub}_S(F_1, F_2) = \{\tau(L) \mid L \in F_2 \text{ and } \tau \text{ is an } F_1\text{-substitution}\}$$

and

$$\text{Sub}_N(F_1, F_2) = \{\tau(L) \mid L \in F_2 \text{ and } \tau \text{ is an } F_1\text{-substitution such that for every } b,\ \tau(b) \neq \{e\}\}.$$

A family F_1 is *closed under substitution* if $\text{Sub}_S(F_1, F_1) \subseteq F_1$; *closed under nonerasing substitution* if $\text{Sub}_N(F_1, F_1) \subseteq F_1$; *closed under substitution into a family* F_2 if $\text{Sub}_S(F_1, F_2) \subseteq F_1$; and *closed under nonerasing substitution into* F_2 if $\text{Sub}_N(F_1, F_2) \subseteq F_1$.

LEMMA 12. *For every $n \geqslant 0$, $Y(D_n)$ and $Y_e(D_n)$ are closed under string substitution. For every $n > 0$, $Y(U_n)$ is closed under string substitution into the e-free context-free languages and $Y_e(U_n)$ is closed under substitution into the context-free languages.*

Proof. The lemma is obtained from closure properties of D_n and U_n by showing that for any families F_1 and F_2 of tree languages, $Y(\text{Sub}_T(F_1, F_2)) = \text{Sub}_S(Y(F_1), Y(F_2))$. Suppose $T \in F_2$ with $T \subseteq \Sigma_*$ and $\tau: \Sigma_0 \to Y(F_1)$ is a string substitution. For each $b \in \Sigma_0$, there is a tree language $S_b \in F_1$ with $Y(S_b) = \tau(b)$. We claim that $Y(T(b: S_b \mid b \in \Sigma_0)) = \tau(Y(T))$. For suppose that $w \in \tau(Y(T))$. For some $t = t_1 b_1 t_2 \cdots t_m b_m t_{m+1} \in T$ with each $b_j \in \Sigma_0$ and each $t_j \in ((\Sigma \cup \Pi) - \Sigma_0)^*$

and some $u_1, \ldots, u_m$ with each $u_j \in S_{b_j}$, $w = Y(u_1) \cdots Y(u_m)$. But $w' = t_1 u_1 t_2 \cdots t_m u_m t_{m+1} \in T(b : S_b \mid b \in \Sigma_0)$ and $Y(w') = Y(u_1) \cdots Y(u_m) = w$. Therefore, $\tau(Y(T)) \subseteq Y(T(b : S_b \mid b \in \Sigma_0))$. The argument is easily reversed to show that $Y(T(b : S_b \mid b \in \Sigma_0)) \subseteq \tau(Y(T))$.

By Theorem 7, $\mathrm{Sub}_S(Y(D_n), Y(D_n)) = Y(\mathrm{Sub}_T(D_n, D_n)) = Y(D_n)$. Since $Y(D_0)$ is the family of e-free context-free languages [10] and U_n is closed under tree substitution into recognizable sets (Theorem 8), $\mathrm{Sub}_S(Y(U_n), e\text{-free } CF) = \mathrm{Sub}_S(Y(U_n), Y(D_0)) = Y(\mathrm{Sub}_T(U_n, D_0)) = Y(U_n)$. If τ is an F-substitution which erases some symbols, τ can be accomplished by applying first a homomorphism which erases the appropriate symbols and then a nonerasing F-substitution. Since $Y_e(D_n)$ and $Y_e(U_n)$ are closed under homomorphism (Lemma 11), $\mathrm{Sub}_S(Y_e(D_n), Y_e(D_n)) = Y_e(D_n)$ and $\mathrm{Sub}_S(Y_e(U_n), Y_e(D_0)) = Y_e(U_n)$. ∎

The next theorem summarizes the known positive closure results for the families in the $Y_e(D) - Y_e(U)$ hierarchy.

THEOREM 13. *For every $n \geqslant 0$, the family $Y_e(D_n)$ is a substitution-closed full abstract family of languages* (AFL); *that is, it is closed under union, concatenation, Kleene *, arbitrary homomorphism, inverse homomorphism, intersection with regular sets, and string substitution. For every $n \geqslant 0$, $Y_e(U_n)$ is closed under union, concatenation, Kleene *, arbitrary homomorphism, intersection with regular sets, and string substitution into the context-free languages.*

Proof. Let $\mathscr{R}$ denote the family of regular sets, and $\mathscr{R}_0$ the family of regular sets not containing the empty string. Ginsburg and Spanier (1970) showed that a family $\mathscr{L}$ of string languages is a full AFL if $\mathscr{R}_0 \subseteq \mathscr{L}$, $\mathrm{Sub}_S(\mathscr{R}, \mathscr{L}) \subseteq \mathscr{L}$, $\mathrm{Sub}_S(\mathscr{L}, \mathscr{R}_0) \subseteq \mathscr{L}$, and $\mathscr{L}$ is closed under intersection with regular sets. Since $\mathscr{R}_0 \subseteq \mathscr{R} \subseteq Y_e(D_n)$ and $Y_e(D_n)$ is closed under string substitution (Lemma 12), and intersection with regular sets (Hopcroft and Ullman, 1969), $Y_e(U_n)$ is a substitution-closed full AFL.

For every $n \geqslant 0$, $Y_e(U_n)$ is closed under substitution into context-free languages (Lemma 12), homomorphism (Lemma 11), and intersection with regular sets (Proposition 10). Closure under union, concatenation, and Kleene * follows since any family of languages closed under substitution into regular sets is also closed under union, concatenation, and Kleene * (Ginsburg and Spanier, 1970). ∎

The above theorem does not state that each $Y_e(U_n)$ is closed under inverse homomorphism. In fact, it is shown in Section 4 that $Y_e(U_n)$ is closed under inverse homomorphism if and only if $Y(D_{n-1}) = Y(U_n)$. Therefore, we conjecture that each $Y_e(U_n)$ is not closed under inverse homomorphism and is not an AFL.

The proofs in this section derive the closure properties of the families in the $Y(D) - Y(U)$ hierarchy from the closure properties of the families in the

D-U hierarchy. This proof technique may be applied to other families of tree languages and their yields; it provides a means of studying string languages derived from tree languages without directly specifying the string languages except as the yield of the tree languages.

SECTION 4

Earlier, it was conjectured that the *D-U* hierarchy and the $Y(D) - Y(U)$ hierarchy are infinite. The closure properties obtained in Sections 2 and 3 are applied here to obtain some results related to these conjectures. The basic theorems of this section are that for every n, if $D_n \subsetneq U_{n+1}$, then $U_{n+1} \subsetneq D_{n+1}$, and if $Y(D_n) \subsetneq Y(U_{n+1})$, then $Y(U_{n+1}) \subsetneq Y(D_{n+1})$. It would follow that the hierarchies are infinite if proofs could be found that for every n, if $U_n \subsetneq D_n$ then $D_n \subsetneq U_{n+1}$ and if $Y(U_n) \subsetneq Y(D_n)$ then $Y(D_n) \subsetneq Y(U_n)$. Unfortunately, attempts to prove the latter statements have been unsuccessful. Nevertheless, the theorems proved here lead to some strong statements about the number of families in the hierarchies if the hierarchies are finite.

In order to make the proofs of the theorems a little simpler, we first prove a proposition concerning one-state transductions.

PROPOSITION 14. $DO = FDOB \circ DOLB$.

Proof. Since *FDOB* and *DOLB* are contained in the class of deterministic one-state transductions, which is closed under composition (Engelfriet, 1975),

$$FDOB \circ DOLB \subseteq DO$$

For the reverse inclusion, let $M = (\{q\}, \Sigma, \Delta, R, \{q\})$ be a deterministic one-state bottom-up transducer. We construct a one-state deterministic linear bottom-up transducer M_1 and a full one-state deterministic bottom-up transducer M_2 such that $T(M_2) \circ T(M_1) = T(M)$. The strategy is to have M_1 imitate M, except that M_1 merely records a rule number of M at each node and deletes whatever subtrees M deletes in that rule. From each rule number, M_2 generates the output contained in the right side of the rule.

Order the rules of R. If R contains m rules, let $\Gamma = \{r_1, \ldots, r_m\}$ be a set of m distinct new ranked symbols. For $i = 1, \ldots, m$, the rank of r_i is k if the right side of rule i contains occurrences of exactly k distinct variables (symbols of the form x_j). Let $M_1 = (\{q\}, \Sigma, \Gamma, R_1, \{q\})$ be a bottom-up transducer, where

$$\begin{aligned} R_1 = {} & \{u \to r_i \mid \text{the right side of rule } i \text{ contains no variables}\} \\ & \cup \{u \to r_i[x_{i_1}, \ldots, x_{i_k}] \mid k > 0 \text{ and the variables occurring in} \\ & \quad \text{the right side of rule } i \text{ are } x_{i_1}, \ldots, x_{i_k},\ 1 \leqslant i_1 < \cdots < i_k\}. \end{aligned}$$

Let $M_2 = (\{q\}, \Gamma, \Delta, R_2, \{q\})$ be a bottom-up transducer, where

$$R_2 = \{r_i \to u \mid u \text{ is the right side of rule } i \text{ and } u \in \Delta_*\}$$
$$\cup \{(r_i, q, \ldots, q) \to u \mid \text{if the right side } v \text{ of rule } i \text{ contains distinct variables } x_{i_1}, \ldots, x_{i_k},\ 1 \leqslant i_1 < \cdots < i_k, \text{ then } u \text{ is obtained by replacing each } x_{ij} \text{ in } v \text{ by } x_j\}.$$

It is straightforward to prove by induction on the depth of s that $u \in M(s)$ if and only if there exists $t \in M_1(s)$ such that $u \in M_2(t)$. Therefore, $T(M) = T(M_2) \circ T(M_1)$. ∎

THEOREM 15. *For every* $n \geqslant 0$, *if* $D_n \subsetneq U_{n+1}$, *then* $U_{n+1} \subsetneq D_{n+1}$.

Proof. We will prove that for any family G of tree languages closed under one-state linear bottom-up transductions, if $G \subsetneq DO(G)$, then $DO(G) \subsetneq \mathrm{Sub}_T(D_0, DO(G))$. The theorem follows by setting $G = D_n$, since $U_{n+1} = DO(D_n) = DO(G)$ and $\mathrm{Sub}_T(D_0, DO(G)) = \mathrm{Sub}_T(D_0, U_{n+1}) \subseteq \mathrm{Sub}_T(D_0, D_{n+1}) \subseteq D_{n+1}$ by Theorem 7.

So let G be a family of tree languages closed under one-state linear bottom-up transductions. Suppose $T \in DO(G) - G$, $T \subseteq \Sigma_*$. From T, we construct a tree language $E \in \mathrm{Sub}_T(D_0, DO(G)) - DO(G)$.

Intuitively, E is obtained from T by substituting chains containing any number of b's at the leaves of T, where b is a new symbol. In order to demonstrate that $E \notin DO(G) = FDOB(G)$ (Proposition 14), we assume initally that $E \in DO(G)$, and obtain a contradiction by exploiting the fact that bottom-up transducers copy subtrees after producing output. In particular, we find a subset V of E such that all the subtrees of certain nodes of trees in V are distinct. Since copying by bottom-up transducers results in two or more identical subtrees, we show that any deterministic nondeleting one-state bottom-up transducer generating E from a tree language in F must generate some set Z, $V \subseteq Z \subseteq E$, by using only linear rules. Thus, $Z \in G$. Finally, T can be recovered from Z by a one-state linear transduction, implying that $T \in G$, which contradicts the choice of $T \notin G$.

We begin by constructing E from T. Let b and a be new symbols of ranks 1 and 0, respectively. For each $c \in \Sigma_0$, let c_1 be a new symbol of rank 1. Let $\Gamma = \Sigma \cup \{a, b\} \cup \{c_1 \mid c \in \Sigma\}$. For each $c \in \Sigma_0$, let W_c be the smallest set of trees such that $c_1[a] \in W_c$, and for each tree $c_1[t] \in W_c$, $c_1[b[t]] \in W_c$. Clearly, each W_c is a recognizable set.

Set $E = T(c : W_c \mid c \in \Sigma_0)$. Since $T \in DO(G)$, $E \in \mathrm{Sub}(D_0, DO(G))$. We obtain a contradiction from supposing that E is also in $DO(G)$.

Suppose $E \in DO(G)$. By Proposition 14, $DO = FDOB \circ DOLB$. Since G is closed under one-state linear bottom-up transductions, there exist a tree language $S \in G$, $S \subseteq \Delta_*$, and a full total deterministic one-state bottom-up transducer $M = (\{r\}, \Delta, \Gamma, P, \{r\})$ such that $E = M(S)$.

Let p denote the maximum number of symbols in the right side of any rule. Consider the subset V of E, defined by

$V = \{t \in E \mid$ if $t = u_1 t_1 u_2 \cdots u_m t_m u_{m+1}$, where each $t_j \in W_c$ for some c and each $u_j \in (\Sigma \cup \Pi)^*$, then for $i \neq j$, t_i and t_j differ in length (as strings) by at least $p + 1$ symbols$\}$.

In particular, consider a computation of M on a tree $s \in S$ with output $u \in V$. Suppose that a nonlinear rule $y \to z$ is applied during this computation. For some i, z contains at least two occurrences of x_i. When this rule is applied, some tree t is substituted for each occurrence of x_i in z in obtaining the output w from this step. Consequently, an application of this nonlinear rule results in one of the following two situations, according to whether or not $t \in W_c$ for some $c \in \Sigma_0$.

(1) If $t \in W_c$ for some $c \in \Sigma_0$, suppose $z = z_1 x_i z_2 x_i z_3$. Since a tree in W_c has no symbols of rank greater than one, but z does, at least two subtrees y_1 and y_2 in W_c are generated in this step, such that t is a subtree of each of them. Moreover, neither y_1 nor y_2 contains more than $| t | + p$ symbols, since z contains at most p symbols, but each contains at least $| t |$ symbols.

(2) If $t \in W_c$, then t has a proper subtree in W_c, for some $c \in \Sigma_0$. Copying t causes the output to contain two identical trees in W_c.

Since M is full and total, both of the above cases force the final output u from the computation not to be in V. We conclude that M never applies a nonlinear rule in a computation which generates a tree in V. Therefore, if M_1 is the one-state linear bottom-up transducer obtained by deleting all nonlinear rules from M, $V \subseteq M_1(S) \subseteq M(S)$.

Finally we construct a one-state linear bottom-up transducer M_2 such that $M_2(M_1(S)) = T$. In particular, let $M_2 = (\{p\}, \Gamma, \Sigma, R_2, \{p\})$ where $R_2 = \{a \to (p, c) \mid c \in \Sigma_0\} \cup \{(c_1, p) \to (p, c) \mid c \in \Sigma_0\} \cup \{(b, p) \to (p, x_1)\} \cup \{(d, p, \ldots, p) \to (p, d[x_1, \ldots, x_n]) \mid n > 0, d \in \Sigma_n\}$. All that M_2 does to a tree in $M(S)$ is to erase all of the a's and b's and change c_1's back to c's. Thus, it is clear that $M_2(M_1(S)) \subseteq T$. On the other hand, for every $t \in T$, $t(c : W_c \mid c \in \Sigma_0) \cap V = \varnothing$. Consequently, for every $t \in T$, there exists $u \in V \subseteq M_1(S)$ such that $M_2(u) = t$. Thus, $T = M_2(M_1(S))$.

Since M_1 and M_2 are one-state linear bottom-up transducers and F is closed under such transductions, $T \in G$. But T was chosen so that $T \notin G$. Therefore, the assumption $E \in DO(G)$ is false. We conclude that $E \in \mathrm{Sub}(D_0, DO(G)) - DO(G)$. ∎

The above proof can be modified to obtain the corresponding theorem for the $Y(D) - Y(U)$ hierarchy.

THEOREM 16. *For every $n \geq 0$, if $Y(D_n) \subsetneq Y(U_{n+1})$, then $Y(U_{n+1}) \subsetneq Y(D_{n+1})$.*

Proof. Let L_1 be any language in $Y(U_{n+1}) - Y(D_n)$, $L_1 \subseteq \Sigma_0{}^*$. From L_1 we construct a language $L_2 \in Y(D_{n+1}) - Y(U_{n+1})$. In particular, let a be a new symbol, and let $h: (\Sigma_0 \cup \{a\})^* \to \Sigma_0{}^*$ be the homomorphism determined by $h(a) = e$, $h(b) = b$ for $b \in \Sigma_0$. Set $L_2 = h^{-1}(L_1)$. Since $Y(D_{n+1})$ is closed under inverse homomorphism, $L_2 \in Y(D_{n+1})$.

We will obtain a contraduction by assuming that $L_2 \in Y(U_{n+1})$. Suppose $L_2 \in Y(U_{n+1})$. Since $U_{n+1} = DO(D_n)$, $DO = FDOB\ DOLB$, and D_n is closed under linear transductions, $U_{n+1} = FDOB(D_n)$. Therefore, for some $S \in D_n$ and some full deterministic bottom-up transducer M, $L_2 = Y(M(S))$. Let $W = M(S)$.

Let V be the subset of W defined by

$$V = \{t \in W \mid \text{if } Y(t) = u_1 a^{i_1} u_2 \cdots u_m a^{i_m} u_{m+1}, \text{ where each } u_j \in \Sigma_0{}^*, \text{ then for } k \neq j,\ i_k \neq i_j\}.$$

Consider a computation of M on a tree $s \in S$ with output $u \in V$. Suppose a nonlinear rule $y \to z$ is applied during this computation. For some i, z contains at least two occurrences of x_i. When this rule is applied, some tree t is substituted for each occurrence of x_i in z to obtain the output at this step. Moreover, this output cannot be deleted later in the computation since M is nondeleting and total. If $Y(s)$ contains a substring ba^kc, with $b, c \in \Sigma_0$, the final output u contains at least two occurrences of ba^kc in its yield, and therefore is not in V.

We will construct a linear bottom-up transducer N which imitates computations of M which never copy more than one symbol of Σ_0, except that N generates no a's: that is, if M outputs w, N outputs $h(w)$.

The new transducer N will operate by using its states to keep track of whether 0, 1, or more symbols in Σ_0 have been generated in each output subtree. Let $Q' = Q \cup (\Sigma_0 \cup \{e, 2\})$. Let r denote the maximum number of symbols occurring in the right hand side of any rule. For each k, $0 \leqslant k \leqslant r$, let b_k be a new symbol of rank k; let Ω denote the set consisting of these new symbols. Let $M = (Q', \Delta, \Sigma \cup \Omega, R', F \times (\Sigma_0 \cup \{2\})$ be a bottom-up transducer, where R' is constructed as follows.

For each rule $(b, q_1, \ldots, q_n) \to (q, u) \in R$, where $n \geqslant 0$, and for each $c_1, \ldots, c_n \in \Sigma_0 \cup \{2, e\}$ such that $c_j \neq 2$ whenever x_j occurs more than once in u, a rule $(b, (q_1, c_1), \ldots, (q_n, c_n) \to ((q, c), u')$ is added to R', where c and u' are obtained as follows. Let $g: (\Sigma \cup \{a\} \cup \Pi \cup X)^* \to (\Sigma_0 \cup X)^*$ be a homomorphism determined by $g(b) = b$ for $b \in \Sigma_0$, $g(x_j) = x_j$ if x_j occurs at most once in u, $g(x_j) = c_j$ if x_j occurs at least twice in u, and $g(b) = e$ if $b \in (\Sigma - \Sigma_0) \cup \{a\} \cup \Pi$. Then

(1) if $g(u) = e$, $u' = b_0$ and $c = e$;

(2) if $g(u) = d \in \Sigma_0$, $u' = c = d$;

(3) if $g(u) = f_1 \cdots d_m \in \Sigma_0^*$, $m > 0$, each $d_j \in \Sigma_0 \cup X$, and $g(u) \notin \Sigma_0$, then $u' = b_m[d_1, \ldots, d_m]$ and $c = 2$.

Intuitively, N enters state e whenever M has output no symbols in Σ_0. N outputs a place-marking symbol b_0 which will be deleted later if symbols in Σ_0 are generated. If M has output exactly one symbol $c \in \Sigma_0$, N outputs c and enters state c. If M has output more than one symbol of Σ_0, and has never copied a subtree containing more than one symbol of Σ_0, then N enters state 2 and outputs individually any symbols which M copied. If M tries to copy a subtree containing more than one symbol in Σ_0, N's action is undefined.

It is straightforward to prove by induction on the depth of s that for $w \in N((q, c), s)$, either (1) $w = b_0$, $c = e$, and $e \in h(Y(M(q, s)))$ or (2) $Y(w) = h(Y(w'))$ for some $w' \in M(q, s)$ and either $c = 2$ and $Y(w)$ contains two or more symbols of Σ_0 or $c = h(Y(w)) \in \Sigma_0$. Therefore, $Y(M(S)) \subseteq h(Y(M(S))) = L_1$.

On the other hand, it is easy to see that if w is generated by M in a computation which never copies more than one symbol of Σ_0, there is a computation of N which generates a tree y with $Y(y) = h(Y(w))$. Therefore, $L_1 \subseteq h(V) \subseteq Y(N(S))$. ∎

The above theorem may be applied to obtain an example of a language in $Y(D_1) - Y(U_1)$.

COROLLARY 17. $Y(D_0) \subsetneq Y(U_1) \subsetneq Y(D_1)$. *In particular,* $L_1 = \{b^{2^n} \mid n > 0\} \in Y(U_1) - Y(D_1)$ *and* $L_2 = \{w \in \{a, b\}^* \mid$ *for some* $n > 0$, w *contains* 2^n b's$\} \in Y(D_1) - Y(U_1)$.

Proof. Since $Y(D_0)$ is the family of e-free context-free languages, $L_1 \notin Y(D_0)$. It is trivial to show that $L_1 \in Y(U_1)$. By the proof of Theorem 16, $L_2 = h^{-1}(L_1) \in Y(D_1) - Y(U_1)$, where h is a homomorphism which erases a's and is the identity on b's. ∎

We now apply various results from this section and the previous section to obtain more information about the number of distinct families in the D-U hierarchy.

LEMMA 18. *For* $n > 0$, *if* $D_n = U_n$, *then for* $m > n$, $D_m = U_m = U_n$. *For* $n \geqslant 0$, *if* $U_{n+1} = D_n$, *then for* $m > n$, $D_m = U_m = D_n$.

Proof. For $n > 0$, if $D_n = U_n$, then $U_{n+1} = DO(D_n) = DO(U_n) = U_n$ (by Proposition 2).

Similarly, if $U_{n+1} = D_n$, then $D_{n+1} = NLT(U_{n+1}) = NLT(D_n) = D_n$ (by Proposition 3). The lemma follows by induction on m. ∎

LEMMA 19. *Either for every* $n \geqslant 0$, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$, *or there exists* $n > 0$ *such that for* $i < n$, $D_i \subsetneq U_{i+1} \subsetneq D_{i+1}$ *and for* $i > n$, $D_i = U_i = D_n$. *Therefore, if the D-U hierarchy is finite, it contains an odd number of distinct families.*

Proof. Let M be the set of all $m \geqslant 0$ such that $D_m = U_{m+1}$. If $M = \varnothing$, then by Lemma 18, for every $n \geqslant 0$, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$ and the hierarchy is infinite. If $M \neq \varnothing$, let k be the least element of M, so that $D_k = U_{k+1}$. Since $D_0 \subsetneq U_1$, $k \geqslant 1$. By choice of k, $D_{k-1} \subsetneq U_k$. But then, by Theorem 15, $U_k \subsetneq D_k$. Therefore, by Lemma 18, for $i < k$, $D_i \subsetneq U_{i+1} \subsetneq D_{i+1}$. Moreover, by Lemma 18, for $m > k$, $D_m = U_m = D_k$. Finally, since $D_0 \subsetneq U_1 \subsetneq D_1 \subsetneq \cdots \subsetneq U_k \subsetneq D_k$, and all other families in the hierarchy are equal to D_k, the hierarchy must contain an odd number of distinct families. ∎

Now, we can show that the D-U hierarchy must contain at least five distinct families.

LEMMA 20. $D_0 \subsetneq U_1 \subsetneq D_1 \subsetneq U_2 \subsetneq D_2$.

Proof. Ogden and Rounds (1972) have shown that $D_0 \subsetneq D_1 \subsetneq D_2$. If $D_1 = U_2$, then by Lemma 18, $D_2 = U_2 = D_1$. Therefore, $D_1 \subsetneq U_2$, and by Theorem 15, $U_2 \subsetneq D_2$. But by Lemma 18, $U_2 \subsetneq D_2$ implies that $D_0 \subsetneq U_1 \subsetneq D_1 \subsetneq U_2 \subsetneq D_2$. ∎

In Section 2 it was shown that for every n, D_n is closed under tree substitution. It seems unlikely that any U_n, $n > 0$, is closed under tree substitution, for reasons discussed in Section 2. In fact, the proof of Theorem 15 showed that if $D_n \subsetneq U_{n+1}$, then $U_{n+1} \subsetneq \mathrm{Sub}_T(D_0, U_{n+1})$. Therefore, we have the following result.

LEMMA 21. *For every $n \geqslant 0$, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$ if and only if U_{n+1} is not closed under tree substitution.*

The closure of each D_n under deterministic one-state transductions and the closure of each U_n under linear transductions yield a result similar to Lemma 21.

PROPOSITION 22. *For every $n > 0$, $U_n \subsetneq D_n \subsetneq U_{n+1}$ if and only if D_n is not closed under deterministic one-state transductions. For every $n \geqslant 0$, $D_n \subsetneq U_{n+1} \subsetneq D_{n+1}$ if and only if U_n is not closed under linear top-down transductions.*

Proof. For each $n > 0$, $D_n = NLT(U_n)$ and $U_n = DO(D_{n-1})$. Also, for each $n \geqslant 0$, D_n is closed under linear top-down transductions (Proposition 3) and U_{n+1} is closed under deterministic one-state transductions (Proposition 2). Therefore, for $n > 0$, $NLT(U_n) = U_n$ if and only if $U_n = D_n = NLT(U_n)$, and for $n \geqslant 0$, $DO(D_n) = D_n$ if and only if $D_n = U_{n+1} = DO(D_n)$. ∎

COROLLARY 23. *D_0 and D_1 are not closed under deterministic one-state transductions. U_1 and U_2 are not closed under linear top-down transductions or under tree substitution.*

The closure properties of the families in the D-U hierarchy are summarized in Table 1.

TABLE 1

Closure Properties of the Tree Heirarchy. "Yes" Indicates the Family is Closed under the Operation. "No" that the family is not Closed under the Operation

	D_n, $n \geqslant 0$, if $D_n \subsetneqq U_{n+1}$	U_n, $n > 0$, if $U_n \subsetneqq D_n$	$\bigcup_{n=0}^{\infty} D_n$
Linear top-down and bottom-up transductions	yes Proposition 3	no Proposition 22	yes Theorem 9
Deterministic bottom-up transductions	no Proposition 22	yes Proposition 2	yes Theorem 9
Deterministic one-state transductions	yes Proposition 22	yes Proposition 2	yes Theorem 9
Tree substitution	yes Theorem 7	no Lemma 21	yes Theorem 9
Tree substitution into the recognizable sets	yes Theorem 7	yes Theorem 8	yes Theorem 9
Intersection with recognizable sets	yes (Ogden and Rounds, 1972)	yes Proposition 5	yes Theorem 9

Techniques similar to the above may be applied to the $Y(D) - Y(U)$ hierarchy.

LEMMA 24. *For every* $n \geqslant 0$, $Y_e(D_n) \subsetneqq Y_e(U_{n+1}) \subsetneqq Y_e(D_{n+1})$ *if and only if* $Y_e(U_{n+1})$ *is not closed under inverse homomorphism.*

Proof. For each n, $Y_e(D_n)$ and $Y_e(D_{n+1})$, and therefore $Y(D_n)$ and $Y(D_{n+1})$ as well, are closed under inverse homomorphism (Theorem 13). Thus, if $Y(U_{n+1})$ is not closed under inverse homomorphism, $Y(D_n) \subsetneqq Y(U_{n+1}) \subsetneqq Y(D_{n+1})$. Conversely, if $Y(D_n) \subsetneqq Y(U_{n+1})$, then $Y(U_{n+1})$ is not closed under inverse homomorphism by the proof of Theorem 16. ∎

The results in this paper indicate that the families $Y_e(D_n)$ and $Y_e(U_n)$ are "natural" families of languages, since their closure properties are similar to those of other "natural" families like the context-free languages and context-sensitive languages. The differences in the known closure properties of the top-down and bottom-up families shed light on the conjecture that the hierarchy is infinite.

Perrault (1975) announced the result that $Y(D_1) \subsetneqq Y(U_2)$. By Theorem 16, $Y(U_2) \subsetneqq Y(D_2)$. Therefore, we have the following theorem.

THEOREM 25. $Y(D_0) \subsetneq Y(U_1) \subsetneq Y(D_1) \subsetneq Y(U_2) \subsetneq Y(D_2)$.

Finally, we note that for every n, $Y(D_n)$ and $Y(U_n)$ are properly contained in the family of context-sensitive languages (Baker, to appear).

ACKNOWLEDGMENT

The author would like to thank R. V. Book for his many helpful comments concerning results of this paper.

RECEIVED: December 31, 1975; REVISED: April 29, 1977

REFERENCES

BAKER, B. Composition of top-down and bottom-up tree transducers, submitted for publication.

BAKER, B. Generalized syntax directed translation, tree transducers, and linear space, *SIAM J. on Computing*, to appear.

DONER, J. (1970), Tree acceptors and some of their applications, *J. Comput. and System Sci.* **4**, 406–451.

ENGELFRIET, J. (1975), Bottom-up and top-down tree transformations, *Math. Systems Theory* **9**, 198–231.

GINSBURG, S., AND SPANIER, E. (1970), Substitution in families of languages, *Info. Sci.* **2**, 83–110.

HOPCROFT, J., AND ULLMAN, J. (1969), *Formal Languages and Their Relation to Automata*, Addison–Wesley, Reading, Massachusetts.

OGDEN, W., AND ROUNDS, W. (1972), Compositions of n tree transducers, *in* "Proceedings Fourth ACM Symposium on Theory of Computing," pp. 198–206.

PERRAULT, C. R. (1975), Inrercalation theorems for tree transducer languages, *in* "Proceedings of Seventh ACM Symposium on Theory of Computing," 126–136.

ROUNDS, W. (1968), Trees, transducers, and transformations, Ph.D. thesis, Stanford University.

ROUNDS, W. (1970), Mappings and grammers on trees, *Math. Systems Theory* **4**, 257–287.

THATCHER, J. (1970), Generalized sequential machine maps, *J. Comput. and System Sci.* **4**, 339–367.

THATCHER, J. (1973), Tree automata, an informal survey, in *Currents in the Theory of Computing* (A. Aho, Ed.), Prentice–Hall, Englewood Cliffs, New Jersey.

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