Mappings and Grammars on Trees

by

WILLIAM C. ROUNDS*†

Case Western Reserve University

Introduction

Recent developments in the theory of automata have pointed to an extension of the domain of definition of automata from strings to trees. Here we study certain sets, functions, and relations on trees using natural generalizations of ordinary automata theory.

Why pursue such a generalization? First, because enlarging the domain of automata theory may strengthen and simplify the subject in the same way that emphasizing strings rather than natural numbers already has done. Second, because parts of mathematical linguistics can be formalized easily in a tree-automaton setting. The theories of transformational grammars and of syntax-directed compilation are two examples. A two-dimensional automata theory seems better suited to handle concepts arising in these areas than does the conventional theory.

The algebraic properties of finite automata on trees have been extensively studied; see Brainerd [5], Doner [8], Mezei and Wright [12], Thatcher [15], Thatcher and Wright [17], and Arbib and Give'on [4]. The notion of *recognizable set* is central to these papers. A finite checking scheme (automaton) is used on an input tree. The scheme analyzes a tree from the bottom (leaves) up to the top (root), classifying the tree as acceptable or not. The recognizable set associated with the automaton is the set of all acceptable trees.

Here we will define sets of trees produced by finite-state generative schemes. In this respect, making automata work from the top down instead of the bottom up is convenient. Rabin [13] was the first to use this idea; his purpose was to define recognizable sets of infinite trees. We do not consider such trees here; our emphasis is on generation, but the top-down concept is important for all our definitions.

We use Thatcher and Wright's algebraic formalism to give succinct descriptions of linguistic constructions in the tree case. Using these constructions, we investigate decision problems and closure properties. Our results should clarify

* A preliminary version of this work appeared in the *Proceedings of the ACM Symposium on the Theory of Computing*, Marina del Rey, California, May 1969.

† Some of these results were obtained in the author's doctoral dissertation, supported by USAF Contract F 44 620–68–C–0030.

the nature of syntax-directed translations and transformational grammars. (The latter prompted the definitions in Rounds [14].) Previous models of transformational grammars had the capability of producing all recursively enumerable sets as transformational languages. The models given here, however, have the property that languages produced are recursive.

We begin in Section I with a discussion of trees. We consider finite, labeled, ordered, rooted trees such that no label occurs on two nodes which have different numbers of branches. Such a tree appears in Figure 1.

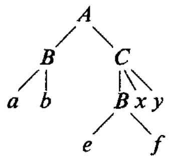

Figure 1.

The top node of this tree is labeled A, and the bottom nodes are a, b, e, f, x, and y.

We define a *dendrolanguage* to be a set of trees of this form. We then discuss recognizable dendrolanguages, relating them to derivation trees of a context-free grammar. These results also appear in Thatcher [15]; we include them because of their linguistic importance. In particular, we want to define functions on context-free derivation trees.

We then introduce the simplest of our models, the deterministic finite-state transformation. In analogy with the generalized sequential machine mapping for strings, we define a function of trees which produces an output tree from a given input tree using finite-state rules, and which works first on the top node of the input tree, then on the second level, and so forth until the bottom nodes have been processed.

Thatcher [16] and Aho and Ullman [3] have recently studied similar models; the former looks at algebraic properties, and the latter at linguistic properties of these mappings. Our definition is slightly more general in that we allow functions to be *partial*. We obtain results about the domain and range of such functions; for example, the domain is a recognizable set.

The *yield* of a tree is defined to be the string of symbols obtained by concatenating all the labels found at the bottom nodes together in left-to-right order. (The yield of the tree in Figure 1 is the string *abefxy*.) The yield of a dendrolanguage is the set of strings obtained by taking the yield of each tree in the dendrolanguage. For each tree function we have a corresponding relation obtained by taking yields of pairs of trees in the function. By considering the ranges of such relations, we obtain sets which extend the context-free languages. These sets are called *target languages*.

In Section II we propose two main variations on the model of Section I. The first is a *nondeterministic* finite-state transformation, obtained from the deterministic version by allowing more than one way to rewrite nodes in the input tree. We still insist, however, that a node be transformed at each stage. (In ordinary transducer language, this would mean that we cannot read the empty input symbol.) We extend the analysis of Section I to the new scheme. The second variation, in addition to being a nondeterministic scheme, allows a transformation to modify the *input* tree at any stage by building a new piece at the top. However, we still try to achieve top-to-bottom processing, and a generation is finished only when all bottom nodes have been transformed. We see that in this case we may produce an infinite dendrolanguage from a finite input set, and we study only this situation. This model is a *creative dendrogrammar.* The yields of creative dendrolanguages are the *indexed languages* of Aho [1]. The importance of indexed languages for transformational linguists remains to be investigated, but these languages arise at an early stage in the study of transformational grammars.

I. Deterministic Transformations

1. Trees. If we think of an automaton carrying out a recursive process on its input, it is natural to think of a recursive description of the input itself. This has been done for strings and natural numbers; in fact, a system of axioms similar to Peano's for the positive integers can be used to define all strings over a given alphabet. An inductive description of trees can be given as well: this description coincides with the ordinary description of terms in a formal system. Of course, we must show that formal terms can be identified with trees in a one-to-one manner. From the definition it should be clear that such a correspondence exists.

The definition we use, found in Thatcher and Wright [17], is a common one from universal algebra and logic. We need the idea of *ranked alphabet*; intuitively, the set of labels which can occur in a tree. We insist that a node with k descendents be labeled by a symbol of rank k. Thus:

Definition. A ranked alphabet is a pair (Σ, r) where Σ is finite, and $r: \Sigma \to \mathbb{N}$. We set

$$\Sigma_n = r^{-1}\{n\}.$$

Now we can define Σ-terms (trees).

Definition. Let (Σ, r) be a ranked alphabet. The set $\mathscr{T}_\Sigma^0$ (the constant Σ-terms) is the smallest set of strings such that:

(a) $\Sigma_0 \subseteq \mathscr{T}_\Sigma^0$;

(b) if $t_0, \cdots, t_{n-1} \in \mathscr{T}_\Sigma^0$, and $\sigma \in \Sigma_n$, then $\sigma(t_0, \cdots, t_{n-1}) \in \mathscr{T}_\Sigma^0$.

We are formally defining certain well-formed strings of symbols over a large alphabet, including parentheses and commas, but this set, rather than the set of all strings, will be the universe of discourse. It will also help to forget that we are talking formally about certain strings, and to picture them instead as geometrical objects.

Example. Let $\Sigma_0 = \{0, 1, a, y\}$, $\Sigma_1 = \{\sin, \cos, -\}$, $\Sigma_2 = \{+, \cdot\}$. A typical element of $\mathscr{T}_\Sigma^0$ is

$$+(\sin(a), \cdot(\cos(y), a));$$

in ordinary notation the term $\sin(a) + a \cdot \cos(y)$. The tree picture of this term appears in Figure 2.

The definition of term guarantees unique readability for any term. Linguistically this means that the definition is really an unambiguous context-free grammar for terms. Therefore, it is not surprising that we can associate a tree picture with a term in a unique way.

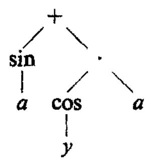

Figure 2.

2. A Preliminary Example. To illustrate the model we plan to define in this section, we will describe a function on $\mathscr{T}^0_\Sigma$, where Σ is the alphabet in the previous example. This function will be the operation of finding a formal term representing the derivative of a given term over Σ, taken with respect to y. The rules which we apply should be the familiar rules for differentiation, and we wish to apply them in a top-down manner to a given tree. Let us find the derivative of the tree in Figure 2 as a special case. This tree represents the sum of two terms. If we begin at the top, the first rule we apply is $D_y(f+g) = D_yf+D_yg$. Let us invent a *state* d which tells us to take the derivative. Then the first rewriting rule—linearity of differentiation—becomes

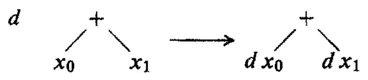

Figure 3.

This rule says: If the process is in state d, and the node to be rewritten is $+$, which may be followed by the subtrees x_0 and x_1, then put out the node $+$ and apply d to the nodes at the top of the subtrees x_0 and x_1. The result of applying such a rule to Figure 2 is

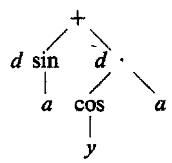

Figure 4.

At this point, two rules become applicable: the chain rule on the left, and the product rule on the right. We can symbolize these:

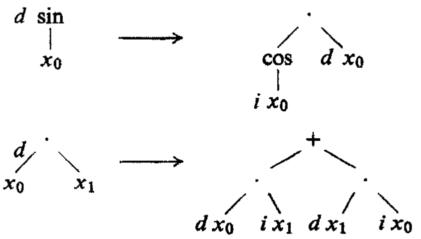

Figure 5.

Here, *i* is a new state, the *identity* or do-nothing state. We then derive

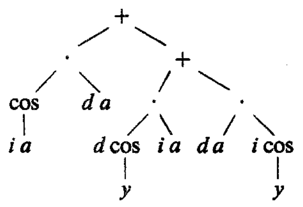

Figure 6.

The reader can easily make up productions which will finish the derivation.

Notice that in applying the product rule to derive the tree in Figure 6, we had to make two copies of the input subtree cos (y). The power to *replicate* subtrees of the input tree is a primitive operation associated with transformations. Notice also how the states sweep through the input tree from top to bottom. There is never a choice but to rewrite a given node in a unique way depending on the state. This is the *deterministic* feature of the model.

3. Recognizable Sets. Transformational theory, as developed by Chomsky [7] and many others, deals with the notion of phrase-structure grammar, and with certain mappings defined on derivation trees associated with the grammar. Derivation trees do not make much sense for context-sensitive grammars, because they depend on the order of carrying out a derivation. We will therefore assume that mappings are to be defined on context-free derivation trees. Intuitively speaking, we may describe the domain of a transformation as a set of tree structures for *simple* (kernel) sentences (e.g., "I see the cat") and a transformation as an operation on the tree for this sentence which changes it into a structure for a closely related sentence (e.g., "The cat was seen by me"). The trees representing simple sentences are called *deep structures*, and transformed trees *surface structures.*

Similarly, the theory of *syntax-directed translation* deals with changing statements in a programming language into some other language by performing operations on the derivation trees of strings in the source language. One of the original schemes of this type was developed by Irons [10]; formalizations have been given by Aho and Ullman ([2], [3]) and Lewis and Stearns [11].

We must, therefore, formalize the idea of a set of derivation trees. Here we follow Thatcher [16].

Definition. An (ϵ-free) *context-free* (CF) grammar over a finite alphabet Σ_0 is a 4-tuple $G = (V, \Sigma_0, S, \Pi)$, where $\Sigma_0 \subseteq V$, V is finite, $S \in V - \Sigma_0$, and Π is a finite set of pairs (A, w) called *productions*, where $A \in V - \Sigma_0$ and $w \in V^* - \{\epsilon\}$. (ϵ is the identity element of the free monoid V^* over V.)

A CF grammar is *ranked* if whenever (A, w) and (A, x) are in Π, then the lengths of w and x are equal.

We may form a ranked alphabet from a given ranked CFG by letting the set Σ_0 be the 0-ary symbols and letting

$$\Sigma_n = \{A \in V - \Sigma_0 | (A, w) \in \Pi \text{ and length } (w) = n\}.$$

Using this ranked alphabet we can define the set of *derivation trees* D_σ^G associated with any $\sigma \in \Sigma$, by induction:

(i) if $\lambda \in \Sigma_0$, $D_\lambda^G = \{\lambda\}$;

(ii) whenever $(\sigma, w) \in \Pi$, $\sigma \in \Sigma_n$ for $n \geq 1$,

$w = \sigma_1 \cdots \sigma_n$, and $t_1 \in D_{\sigma_1}^G, \cdots, t_n \in D_{\sigma_n}^G$, then $\sigma(t_0, \cdots, t_{n-1}) \in D_\sigma^G$.

The set D^G of derivation trees of G is the set D_S^G. Notice that under the correspondence of trees with terms, a term in D_σ^G is a tree with top node σ, and such that if τ is any node label, the labels σ_i on the immediate successors satisfy the requirement that $\tau \to \sigma_0 \cdots \sigma_m$ is a production of G. Notice also that any (ϵ-free) context-free language can be obtained from a ranked CF grammar, by relabeling non-terminal symbols. (We could avoid using ranked grammars if we discussed ranked alphabets Σ where r was a relation instead of a function). No languages will contain the empty word in our discussion.

Definition. Let Σ be a ranked alphabet. A Σ-dendrolanguage is any subset of $\mathscr{T}_\Sigma^0$. The sets D^G are thus simple Σ-dendrolanguages, which could be called *derivation dendrolanguages.*

We need a function to read off the sequence of bottom symbols on a tree. This function will be called the *yield* of a tree:

$$y(\lambda) = \lambda \quad \text{for} \quad \lambda \in \Sigma_0;$$

$$y(\sigma(t_0, t_1, \cdots, t_{n-1})) = y(t_0) \cdot y(t_1) \cdots y(t_{n-1})$$

where is concatenation in Σ_0^*. The yield of a dendrolanguage $\mathscr{R}$ is

$$y[\mathscr{R}] = \{y(t) | t \in \mathscr{R}\}.$$

A context-free language is thus the yield of a derivation dendrolanguage.

Now we can define the important class of *recognizable* dendrolanguages. These sets, a generalization of regular sets of strings, are closely related to the derivation dendrolanguages. First, we define tree automata [5], which can be viewed as finite checking schemes for a tree. Each node σ in a tree of rank n induces a finite function α_σ: $A^n \to A$, where A is intuitively the set of states of the automaton.

Definition. Let A be a set. By an assignment of Σ-operations on A we mean a function α: $\Sigma \to \{A^{(A^n)} | n \geq 0\}$ such that if $\sigma \in \Sigma_n$ then $\alpha(\sigma) \in A^{(A^n)}$. $\alpha(\sigma)$ will be written α_σ, and is simply an n-ary operation on A. If $\lambda \in \Sigma_0$, α_λ is a fixed element of A. (These α_σ will be the next-state functions.)

Definition. A Σ-algebra is a pair $\mathscr{A} = (A, \alpha)$ where A is nonempty and α is an assignment of Σ-operations on A. If A is finite $\mathscr{A}$ is said to be finite.

Definition. A finite Σ-automaton is a triple (A, α, A_F) where (A, α) is a finite Σ-algebra and $A_F \subseteq A$. A_F is the set of *designated final states.*

Speaking automaton-theoretically, we can now extend the next state function to all of $\mathscr{T}_\Sigma^0$.

Definition. The response function $\| \ \|_{\mathscr{A}}$ of a Σ-algebra is defined inductively by

(i) $\|\lambda\| = \alpha_\lambda$ for $\lambda \in \Sigma_0$;

(ii) $\|\sigma(t_0, \cdots, t_{n-1})\| = \alpha_\sigma(\|t_0\|, \cdots, \|t_{n-1}\|)$.

As is easy to verify, the evaluation of the response function on a tree corresponds to checking the tree from the bottom up.

We are in a position to define recognizable sets:

Definition. $\mathscr{R} \subseteq \mathscr{T}_\Sigma^0$ is *recognizable* if there is a Σ-automaton $\mathscr{A} = (A, \alpha, A_F)$ such that

$$\mathscr{R} = \{t \mid \|t\|_{\mathscr{A}} \in A_F\}.$$

We do not develop any properties of recognizable sets here; many standard properties still hold in the tree case; in particular, decision problems are solvable. We state two results of Thatcher [15], which relate recognizable sets to derivation dendrolanguages; these are the reasons we review recognizable sets here.

THEOREM. *Every derivation dendrolanguage is recognizable.*

THEOREM. *Every recognizable dendrolanguage can be obtained from a derivation dendrolanguage by a function (projection) which renames nodes in a tree.*

As corollaries, we find that the yield of a recognizable set is a CF language, and that every CF language can be obtained this way.

4. Deterministic finite-state transformations. We want to formalize mappings like the syntactic derivative of Section 2. As indicated in the introduction, this should be done linguistically, not algebraically, although the two approaches are equivalent. We use the idea of a *tree production.* This will also permit succinct definitions of more complicated models.

To formalize a rule like

$$\begin{tikzcd}[column sep=small]
 & d\ + \arrow[dl, dash] \arrow[dr, dash] & & \longrightarrow & & + \arrow[dl, dash] \arrow[dr, dash] & \\
x_0 & & x_1 & & d\ x_0 & & d\ x_1
\end{tikzcd}$$

we need only imitate the ordinary notation for trees as terms. We get

$$(d, +(x_0, x_1)) \to +((d, x_0), (d, x_1)).$$

The linearization of the product rule would be

$$(d, \cdot\,(x_0, x_1)) \to +(\cdot((d, x_0), (i, x_1)), \cdot\,((i, x_0), (d, x_1))).$$

Unfortunately, we have not written down well-formed terms, because pairs like (d, x_0) occur as labels. The solution is to enlarge the set of terms so that other objects besides elements of Σ_0 occur at the bottom nodes of a tree. These other elements will be called *indices* and will come from a specified set disjoint from Σ_0.

Definition. Let I be a set disjoint from Σ_0. The set of Σ-terms indexed by I, written $\mathscr{T}_\Sigma(I)$, is the smallest set of strings such that:

(i) $I \cup \Sigma_0 \subseteq \mathscr{T}_\Sigma(I)$;

(ii) $\sigma \in \Sigma_n$ and $t_0, \cdots, t_{n-1} \in \mathscr{T}_\Sigma(I)$ imply
$\sigma(t_0, \cdots, t_{n-1}) \in \mathscr{T}_\Sigma(I)$.

Particular index sets I follow.

Definition. Let X be a fixed countable set $\{x_0, x_1, \cdots\}$. The set $\mathscr{T}_\Sigma(X)$ is the set of all terms in the *variables* X.

Denote by X_n the subset $\{x_0, \cdots, x_{n-1}\}$ of X. If Q is a finite set (set of states), then we can define productions:

Definition. A finite-state (index-erasing) production over Q and Σ is a pair $((q, \sigma(x_0, \cdots, x_{n-1})), t')$ written $(q, \sigma(x_0, \cdots, x_{n-1})) \rightarrow t'$ such that $t' \in \mathscr{T}_\Sigma(Q \times X_n)$.

The reason for the name *index-erasing* is that application of a production to a given node takes place only once. Every time a node is rewritten, a new index node is designated for the next application of a production. This corresponds to the action of a finite-state machine reading and erasing its input.

The next objective is to define the entities to which productions apply. Looking at the example of Section 2, we see that they should be trees with states occurring in the branches. The subtree below a state represents undeveloped input, and the state marks an *active location*. We can represent such a configuration as an element of $\mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma^0)$, where a pair $(q, t) \in Q \times \mathscr{T}_\Sigma^0$ is an index which represents an input subtree t with the state q attached to the top.

All that remains is to describe how a production applies to an intermediate configuration. Let us do it first informally. Given a configuration v choose some $(q, t) \in Q \times \mathscr{T}_\Sigma^0$ occurring as an index in v. Let $t = \sigma(s_0, \cdots, s_{n-1})$. Suppose there is a production $(q, \sigma(x_0, \cdots, x_{n-1})) \rightarrow u$ in the given set of productions (for a given mapping). Here, $u \in \mathscr{T}_\Sigma(Q \times X_n)$. Let t' be the result of substituting s_0 for $x_0, \cdots, s_{n-1}$ for x_{n-1}, whenever these variables occur as indices in u. In other words, if (r, x_j) occurs as an index in u, replace it by the element (r, s_j) of $Q \times \mathscr{T}_\Sigma^0$. Now replace the entire index (q, t) by the new tree t'. The result is the tree v' obtained by applying the given production.

(Note: At each step we select a single occurrence of an index $(q, \sigma(s_0, \cdots, s_{n-1}))$ in v to which we apply the production $(q, \sigma(x_0, \cdots, x_{n-1})) \rightarrow u$.)

We can now give a full formal description of the class of mappings we have in mind.

Definition. A (deterministic) *finite-state transformation* is a 4-tuple

$$T = (\Sigma, Q, q_0, \Pi),$$

where Σ is a ranked alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, and Π is a finite set of index-erasing productions over Q and Σ such that for each pair $(q, \sigma) \in Q \times \Sigma$, there is at most one production $(q, \sigma) \rightarrow u$ in Π. A transformation is *total* if there is exactly one production for each pair in $Q \times \Sigma$.

Remark. We are defining transformations such that the domain and range of the mappings are trees over the same alphabet. This is a minor point, and we shall sometimes modify input and output alphabets when it is convenient.

Definition. (Direct generation) Given $t \in \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma)$, the set of trees t' such that t directly generates t' (via T) is defined inductively on t.

(i) If $t \in \Sigma_0$ then $\{t' | t \Rightarrow t'\} = \varnothing$;
(ii) if $t \in Q \times \mathscr{T}_\Sigma$ then $t = (q, \bar{t})$ where $\bar{t} \in \mathscr{T}_\Sigma^0$.

There is a subdefinition depending on the form of $\bar{t}$.

(a) If $\bar{t} = \lambda \in \Sigma_0$, then if there is a production $(q, \lambda) \rightarrow t'$ in Π, then

$$\{t' | (q, \bar{t}) \Rightarrow t'\} = \{t'\}.$$

If not, then $\{t' | (q, \bar{t}) \Rightarrow t'\} = \varnothing$.

(b) If $\bar{t} = \sigma(s_0, \cdots, s_{n-1})$, then if there is a production $(q, \sigma(x_0, \cdots, x_{n-1})) \rightarrow u$ in Π, then $\{t' | (q, \bar{t}) \Rightarrow t'\} = \{t' | t'$ can be obtained from u by substituting s_0 for x_0 in each pair (r, x_0) indexing u, substituting s_1 for x_1, and so forth (up to s_{n-1} for x_{n-1})$\}$ .If there is no such production then $\{t' | (q, \bar{t}) \Rightarrow t'\} = \varnothing$.

(iii) If $t = \sigma(t_0, \cdots, t_{n-1})$ then $t \Rightarrow t'$ if for exactly one i, $t' = \sigma(t_0, \cdots, t_i', \cdots, t_{n-1})$ and $t_i \Rightarrow t_i'$.

We can decide effectively when two trees t and t' are such that $t \Rightarrow t'$.

The previous conditions define a relation $\Rightarrow$ on $\mathcal{T}_\Sigma(Q \times \mathcal{T}_\Sigma^0)$. Let $\Rightarrow^*$ be the reflexive, transitive closure of $\Rightarrow$.

Definition. Let $T = (\Sigma, Q, q_0, \Pi)$. The function computed by T is the set of pairs $T = \{(s, s') \in \mathcal{T}^0 \times \mathcal{T}^0 | (q_0, s) \Rightarrow^* s'\}$.

One easily shows that T is a function (using induction on s). If T is total, then it computes a total function.

Examples. (a) We leave it to the reader to write out a formal description of the differentiation operator (Section 2).

(b) DeMorgan's law for Boolean polynomials. This function takes a Boolean polynomial over a finite set W of variables and transforms it into an equivalent one so that the variables are the only subexpressions occurring with complement signs on them.

Let $T = (\Sigma, Q, q_0, \Pi)$, where

(i) $Q = \{c, j\}$

(ii) $q_0 = j$

(iii) $\Sigma_0 = W$, the given set of variables

$\Sigma_1 = \{\neg\}$

$\Sigma_2 = \{\vee, \wedge\}$

(iv) Π has eight productions as follows:

$$(j, \wedge(x_0, x_1)) \rightarrow \wedge((j, x_0), (j, x_1))$$
$$(j, \vee(x_0, x_1)) \rightarrow \vee((j, x_0), (j, x_1))$$
$$(j, \neg(x_0)) \rightarrow (c, x_0)$$
$$(j, w) \rightarrow w \text{ for any } w \in W.$$

Now j is a state which looks for a complemented subexpression. When such an expression is found, the complement sign is erased and the process goes to a new state which will carry out DeMorgan's law:

$$(c, \wedge(x_0, x_1)) \rightarrow \vee((c, x_0), (c, x_1))$$
$$(c, \vee(x_0, x_1)) \rightarrow \wedge((c, x_0), (c, x_1))$$
$$(c, \neg(x_0)) \rightarrow (j, x_0)$$
$$(c, w) \rightarrow \neg(w) \text{ for } w \in W.$$

In the previous two examples, the transformations were total. Not every transformation has this property, of course. We may have an alphabet Σ and a proper subalphabet Δ, and may wish to define a mapping on $\mathscr{T}_\Delta^0$ only, with values in $\mathscr{T}_\Sigma^0$. It is convenient to leave productions which read symbols in $\Sigma\backslash\Delta$ out of the definition. If a tree with some node in $\Sigma\backslash\Delta$ occurs, we wish our transformation to be undefined. (This behavior is called *blocking* in transformational theory). Our first result about transformations is

THEOREM 1. *The domain of a partial deterministic transformation is a recognizable set (effectively obtainable).*

Proof. Let $T = (\Sigma, Q, q_0, \Pi)$ be the given transformation. We construct an automaton $\mathscr{A} = (A, \alpha, A_F)$:

(i) $A = \mathscr{P}(Q)$ (all subsets of Q);

(ii) $\alpha_\lambda = \{q \in Q \mid \exists[(q, \lambda) \to u] \in \Pi\}$;

$\alpha_\sigma(Q_0, \cdots, Q_{n-1}) = \{q \in Q \mid \exists[(q, \sigma) \to u] \in \Pi$, such that whenever (q', x_i) indexes u, then $q' \in Q_i\}$.

Since Π is finite, one may effectively construct α_λ and α_σ for each $\lambda \in \Sigma_0$, $\sigma \in \Sigma_n$. We claim that for each $t \in \mathscr{T}^0$, and each $q \in Q$,

$$(\exists s \in \mathscr{T}^0)\,((q, t) \Rightarrow^* s) \text{ if and only if } q \in \|t\|_{\mathscr{A}}.$$

The proof is by induction on t. Suppose first that $t = \lambda \in \Sigma_0$. If $(q, t) \Rightarrow^* s \in \mathscr{T}^0$, it must be by a one-step process, so there is a production $(q, \lambda) \to s$ in Π. Thus, $q \in \|\lambda\|_{\mathscr{A}}$. (The converse is evident.) Suppose now that the result holds for $t_0, \cdots, t_{n-1}$, and let $t = \sigma(t_0, \cdots, t_{n-1})$. If $(q, t) \Rightarrow^* s$ where $s \in \mathscr{T}^0$, then there is a production $(q, \sigma) \to u$ in Π and a tree t' such that t' is obtained from u by substituting t_i for x_i whenever (r, x_i) indexes u. Now $(r, t_i) \Rightarrow^* s_i$, where $s_i \in \mathscr{T}^0$. By hypothesis, $r \in \|t_i\|$. Since this is true whenever (r, x_i) indexes u, we conclude by definition of α_σ that $q \in \|t\|$. Conversely, let $q \in \|t\|$. Then, there is some production $(q, \sigma) \to u$ in Π such that whenever (r, x_i) is an index of u, we have $r \in \|t_i\|$. The hypothesis applies, telling us that there is $s_i \in \mathscr{T}^0$ such that $(r, t_i) \Rightarrow^* s_i$. Since the production $(q, \sigma) \to u$ applies to t, yielding t', and the indices (r, t_i) occurring on t' all generate terminal trees, so does (q, t).

The theorem follows when we take $A_F = \{q_0\}$.

We used implicitly the fact that for any tree $t \in \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma^0)$, t generates some terminal tree (element of $\mathscr{T}^0$) if and only if every (q, t_i) occurring as an index on t generates a terminal tree. This fact is easy to prove by induction.

Now, we wish to investigate composition of transformations.

THEOREM 2. *Total deterministic transformations are effectively closed under composition.*

Proof. Define $T(q, t)$ to be the unique tree s such that $(q, t) \Rightarrow_T^* s$.

We want to make the actions of S and T take place simultaneously. As soon as T produces an output, from application of a production, S will act on that output. This suggests defining right-hand sides of productions for the composite U to be the result of S acting on the right-hand sides of productions of T. This result will of course depend as well on which state S starts in. The production of U will therefore be of the form $((q^S, q^T), \sigma) \to v$, where $(q^T, \sigma) \to u$ is in Π^T, and v is the result of S acting on u starting in state q^S. Of course, $u \in \mathscr{T}_\Sigma(Q \times X_n)$, so strictly speaking, S is not defined on u. However, it is easy to give an inductive definition of the action $\bar{S}(q^S, u)$ of S on u starting in state q^S. For constants,

$\bar{S}(q^S, \lambda) = S(q^S, \lambda)$. For variable pairs, $\bar{S}(q^S, (q^T, x)) = ((q^S, q^T), x)$. For u of the form $\sigma(t_0, \cdots, t_{n-1})$, $\bar{S}(q^S, u)$ is the result of replacing every index (r^S, x_i) (in the tree t' such that $(q^S, \sigma) \to t'$) by $\bar{S}(r^S, t_i)$. Obviously, if $u \in \mathscr{T}^0$, $\bar{S}(q^S, u) = S(q^S, u)$. Otherwise $\bar{S}(q^S, u) \in \mathscr{T}_\Sigma((Q^S \times Q^T) \times X_n)$ whenever the variables of u are in X_n.

Now we can begin the proof. Let $S = (\Sigma, Q^S, q_0^S, \Pi^S)$, $T = (\Sigma, Q^T, q_0^T, \Pi^T)$. T is to be carried out first. Define $U = (\Sigma, Q^U, q_0^U, \Pi^U)$ by letting $Q^U = Q^S \times Q^T$, $q_0^U = (q_0^S, q_0^T)$, and by putting the productions $((q^S, q^T), \sigma) \to \bar{S}(q^S, u)$ (for every $q^S \in Q^S$) into Π^U whenever $(q^T, \sigma) \to u$ is in Π^T. We want to prove by induction on $t \in \mathscr{T}^0$ that

$$U((q^S, q^T), t) = S(q^S, T(q^T, t)).$$

For $t = \lambda$, this is clear, because $T(q^T, \lambda) \in \mathscr{T}^0$, so $\bar{S}(q^S, T(q^T, \lambda)) = S(q^S, T(q^T, \lambda))$. Also, $((q^S, q^T), \lambda) \to \bar{S}(q^S, T(q^T, \lambda))$ is a production in Π^U. The result follows.

Now suppose $t = \sigma(t_0, \cdots, t_{n-1})$, and assume the result for each $q^S \in Q^S$, and $q^T \in Q^T$ when applied to $t_0, \cdots, t_{n-1}$. We calculate $U((q^S, q^T), t)$ by first applying $((q^S, q^T), \sigma) \to \bar{S}(q^T, u)$ to t, where $(q^T, \sigma) \to u$ is the applicable production of Π^T. Let v be the first stage in calculating $U((q^S, q^T), t)$. Also, let t' be the result of applying $(q^T, \sigma) \to u$ to t. A typical index on t' looks like (r^T, t_j), where (r^T, x_j) indexes u. Let us write $t' = u[(r^T, t_j)]$ by which we mean that (r^T, t_j) occurs at a fixed location in u.

We can similarly write

$$v = \bar{S}(q^T, u)\,[((r^S, r^T), t_j)],$$

but we mean to specify here that (r^T, t_j) is the same index occurring in t' that we picked out before. Thus the index $((r^S, r^T), t_j)$ depends on our previous choice of index. Now $T(q^T, t) = u[T(r^T, t_j)]$. (The index (r^T, t_j) has been transformed; assume all others have also been transformed.) Hence,

$$\begin{aligned} S(q^S, T(q^T, t)) &\Rightarrow_S^* \bar{S}(q^T, u)\,[r^S, T(r^T, t_j)] \\ &\Rightarrow_S^* \bar{S}(q^T, u)\,[S(r^S, T(r^T, t_j))] \end{aligned}$$

and by inductive hypothesis this is equal to

$$\bar{S}(q^T, u)\,[U((r^S, r^T), t_j)] \in \mathscr{T}_\Sigma^0.$$

But this last quantity is just $U((q^S, q^T), t)$.

Theorem 2 is a little special, and we naturally ask whether it can be extended to more general transformations. The answer is negative for partial deterministic ones as well as for nondeterministic ones. We present a counterexample of W. Ogden (personal communication) for the partial deterministic case. Thatcher [16] has an example for the nondeterministic case. These are counterexamples to the theorems in Rounds [14].

Example. Let $\Sigma_0 = \{\lambda, \omega\}$, $\Sigma_2 = \{a\}$. $T = (\Sigma, Q^T, q_0^T, \Pi^T)$, where $Q^T = \{q_0, q_1\}$; $q_0^T = q_0$;

Π^T consists of

$$(q_0, a(x_0, x_1)) \to a((q_1, x_0), (q_0, x_1));$$
$$(q_1, a(x_0, x_1)) \to a((q_1, x_0), (q_1, x_1));$$
$$(q_0, \lambda) \to \lambda; \quad (q_1, \lambda) \to \lambda; \quad (q_1, \omega) \to \omega.$$

Now T defines a partial function on $\mathscr{T}_\Sigma$ which is the identity on the set of all trees whose extreme right-hand bottom node is not labeled with an ω. The function is undefined for trees not in this set.

The system S is $(\Sigma, Q^S, q_0^S, \Pi^S)$, where

$$Q^S = \{r_0, r_1\}; \quad q_0^S = r_0;$$

the following productions make up Π^S:

$$(r_0, a(x_0, x_1)) \to (r_1, x_0);$$
$$(r_0, \lambda) \to \lambda; \quad (r_0, \omega) \to \omega;$$
$$(r_1, a(x_0, x_1)) \to a((r_1, x_0), (r_1, x_1));$$
$$(r_1, \lambda) \to \lambda; \quad (r_1, \omega) \to \omega.$$
$$S(a(t_0, t_1)) = t_0; \quad S(\lambda) = \lambda; \quad S(\omega) = \omega.$$

We notice that $S(T(a(t_0, t_1))) = t_0$ if t_1 is not labeled with an ω on the extreme right-hand leaf; otherwise it is undefined. We claim $S \cdot T$ is not partial deterministic.

Let $U = S \cdot T$. If U were partial deterministic, then there would be a production $(p_0, a(x_0, x_1)) \to t$ (p_0 is the initial state). Here t must have a variable index, but no more than one, because $U(a(\omega, \lambda)) = \omega$. Now t cannot have a constant node for the same reason. Thus t must be of the form (p, x_0) or (p, x_1) where p is a state. If the first case occurs, then since $U(a(\omega, \lambda)) = \omega$, $(p, \omega) \to \omega$ must be a production. Then, the derivation

$$(p_0, a(\omega, \omega)) \Rightarrow (p, \omega) \Rightarrow \omega$$

is possible; but $T(a(\omega, \omega))$ is undefined. In the other case, a similar contradiction is obtained.

5. Transformational Systems. The composite mapping U just described fails to be a partial transformation because it can act on a tree for which the first transformation is undefined. If we were not allowed to give such trees as arguments, then we could, in fact, write a partial transformation which would agree with U on all trees in the domain of T. But this domain is a recognizable set. This fact leads us to define a deterministic *transformational system* as a pair $(\mathscr{R}, T)$, where $\mathscr{R}$ is a recognizable dendrolanguage and T is a deterministic transformation. This definition makes sense from the point of view of transformational grammars, because transformations are defined on the derivation dendrolanguages associated with CF grammars. Such dendrolanguages are recognizable sets. Our idea is to restrict the transformation T to the dendrolanguage $\mathscr{R}$.

We again wish to study closure properties of restricted transformations. These fall into two categories: one, the transformations themselves as functions, and two, relations obtained by taking yields. In the remainder of this chapter we will discuss just a few of these properties.

For the first category, we have just seen that closure under composition fails unless transformations are total. Another fact is that transformations do not in general preserve recognizable sets. (*Proof.* Let $\Sigma_0 = \{\lambda, \Omega\}$, $\Sigma_2 = \{a\}$. Define

T so that $T(a(t_0, t_1)) = a(t_0, t_0)$. Then $y(T[\mathscr{T}_\Sigma]) = \{xx | x \in \Sigma_0^*\}$. If $T[\mathscr{T}_\Sigma]$ were recognizable, then $\{xx | x \in \Sigma_0^*\}$ would be a CFL. Contradiction.) We do, however, have a weak result.

Definition. Let $(\mathscr{R}, T)$ be a transformational system. The (deterministic) *surface dendrolanguage* produced by $(\mathscr{R}, T)$ is the set $T[\mathscr{R}]$.

THEOREM 3. *Deterministic transformations preserve deterministic surface dendrolanguages.*

Proof. This is essentially a modification of the proof of Theorem 2. With S, T, U given as in that proof, we observe that if T is defined on t starting in state q^T, then $U((q^S, q^T), t) = S(q^S, T(q^T, t))$.

By the equality here we mean that one side is defined if and only if the other side is (S may not be total).

Now if $(\mathscr{R}, T)$ is the system producing $T[\mathscr{R}]$ as a surface dendrolanguage, let $\mathscr{R}' = \mathscr{R} \cap \text{domain}(T)$. $\mathscr{R}'$ is recognizable because domain (T) is recognizable, and because we have closure under intersection for recognizable sets. Now $T(q_0^T, t)$ is defined for every $t \in \mathscr{R}'$. Therefore,

$$U[\mathscr{R}'] = S[T[\mathscr{R}]].$$

Yields of trees occurring in a restricted transformation will also prove to be fruitful.

Definition. Let $(\mathscr{R}, T)$ be a transformational system. The (deterministic) *translation* defined by $(\mathscr{R}, T)$ is the set

$$\{(y(s), y(t)) | (s, t) \in T \cap (\mathscr{R} \times \mathscr{T}_\Sigma^0)\}.$$

If T is total, then translations coincide with the GSDT's of Aho and Ullman [3].

It follows from work of Aho and Ullman that translations (for total transformations) are not closed under relational composition. We suspect that this is true also for partial and even nondeterministic ones, though we do not study the question here. We may, however, still consider domains and ranges of translations. From Theorem 1 it follows that the domain of a translation is context-free. The range, by our previous example, need *not* be context-free.

Definition. A (deterministic) *target language* is the range of a (deterministic) translation. Since the range of a relation is empty if and only if the domain is, and since we may effectively obtain a CFG whose associated language is the given domain, it follows immediately that the class of deterministic target languages has a solvable emptiness problem. We know very little else about this class; most of the interesting results are obtained for the nondeterministic version. We therefore turn to these extended models.

II. Nondeterministic Models

In this section we introduce choice as a capability of transformations. We shall consider both grammars and nondeterministic mappings of trees but will use productions to define each model. *Roughly* speaking, a grammar is a nondeterministic mapping applied to a finite set of inputs (the starting configurations), whose range is, in general, infinite. In contrast, a nondeterministic transformation yields, for each input, a finite set of outputs. Such a mapping must therefore have infinite domain to produce an infinite range.

Transformational grammars and generative grammars in general are nondeterministic. Transformations, however, seem to have the property that given a deep structure for a sentence, only finitely many surface structures result from a single application. (We assume here that transformations are not iterated.) It is clear also that transformations should not be total functions. For example, only trees which satisfy a *structural description* associated with a transformation can be changed by that transformation. If a tree does not satisfy such a description, we may wish the transformation to be undefined. Another bit of evidence for non-functionality is the notion of *optional rule.* Certain transformations have choices built into them; one may decide at will whether or not to rearrange word order in some sentences, for example. The precise idea of nondeterminism is intended to approximate this feature of transformational grammars.

We shall first investigate some mathematical properties of nondeterministic transformations, indicating the merits and drawbacks of these models and certain generalizations. Then we will consider grammars on trees, concentrating on an analogue of context-free grammars in the tree case. The study of tree grammars at this point is not nearly complete.

1. Nondeterministic Finite-State Transformations. The definition of nondeterministic transformation is immediate: simply allow any finite number of productions with a given left-hand side. Allow also a *set* of starting states instead of a single initial state. Formally:

Definition. A nondeterministic FS transformation is a 4-tuple $T = (\Sigma, Q, Q_0, \Pi)$, where Σ is a ranked alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, and Π is a finite set of index-erasing productions over Q and Σ.

The definition of direct generation is the same as for deterministic transformations.

Definition. The relation computed by a nondeterministic transformation T is the set

$$\{(s, s') \in \mathscr{T}^0 \times \mathscr{T}^0 | (\exists q \in Q_0)\, ((q_0, s) \Rightarrow^* s')\}.$$

A partial deterministic transformation is an honest special case of a nondeterministic one. For some pairs (q, σ) the set of productions with these pairs for left-hand side may be empty.

We have an immediate theorem for FS relations.

THEOREM 1. *The domain of a nondeterministic FS relation is recognizable (effectively).*

The construction of an automaton to recognize domain (T) is exactly parallel to the construction given for Theorem I-1, and may be safely omitted here. We can now prove the converse result:

THEOREM 2. *Every recognizable set is the domain of a non-deterministic transformation.*

Proof. Let $\mathscr{R} \subseteq \mathscr{T}^0_\Sigma$ be recognized by $\mathscr{A} = (A, \alpha, A_F)$. Let $Q^T = A$, $\Sigma^T = \Sigma$, $Q^T_0 = A_F$, and let

$$(q, \sigma) \to \sigma((q_0, x_0), \cdots, (q_{n-1}, x_{n-1}))$$

be in Π^T exactly when $\alpha_\sigma(q_0, \cdots, q_{n-1}) = q$. We claim: For all $q \in Q^T$, $t \in \mathscr{T}^0$,

$$(\exists s \in \mathscr{T}^0)\ ((q, t) \Rightarrow^* s) \text{ if and only if } \|t\|_{\mathscr{A}} = q.$$

We prove one half of this assertion by induction on t. The statement is obvious for $t \in \Sigma_0$. Assume it therefore for $t_0, \cdots, t_{n-1}$ and all $q \in Q$. Suppose

$$(\exists s)\ (q, \sigma(t_0, \cdots, t_{n-1})) \Rightarrow^*_T s.$$

Then

$$(q, \sigma(t_0, \cdots, t_{n-1})) \Rightarrow \sigma((q_0, t_0), \cdots, (q_{n-1}, t_{n-1})) \Rightarrow^* s,$$

where $(q_0, \cdots, q_{n-1})$ is such that

$$\alpha_\sigma(q_0, \cdots, q_{n-1}) = q.$$

The hypothesis implies that $\|t_0\|_{\mathscr{A}} = q_0, \cdots, \|t_{n-1}\|_{\mathscr{A}} = q_{n-1}$, so that $\|t\|_{\mathscr{A}} = q$.

The other half of the assertion is just as easy to prove, so we omit it.

An open question: Can we construct a *deterministic* transformation T recognizing $\mathscr{R}$?

2. Transformational Systems and Surface Dendrolanguages. In keeping with previous definitions, we define a nondeterministic transformational system as a pair $(\mathscr{R}, T)$ where $\mathscr{R}$ is recognizable and T is a NDFST. Since transformations are not closed under composition (see [16] and below) we cannot immediately study the effect of arbitrary transformations on recognizable sets. This is something of a drawback, but can be remedied in the case of *linear* transformations, as we shall see.

Definition. The surface dendrolanguage associated with $(\mathscr{R}, T)$ is the range of the relation computed by T when restricted to $\mathscr{R}$. This set will be denoted by $T[\mathscr{R}]$.

An obvious property of surface dendrolanguages is effective closure under unions. To prove it, let $T[\mathscr{R}]$ and $S[\mathscr{R}']$ be given; let $\rho \notin \Sigma$ have rank 1. The set

$$\mathscr{S} = \{\rho(t) \mid t \in \mathscr{R} \cup \mathscr{R}'\}$$

is recognizable. Define a nondeterministic U by making the state sets of S and T disjoint, adding a new initial state u_0, and productions

$$(u_0, \rho(x_0)) \to (q^T_0, x_0) | (q^S_0, x_0).$$

Thus, $U[\mathscr{S}] = T[\mathscr{R}] \cup S[\mathscr{R}']$, proving the result.

We are now going to establish a result on composition of non-deterministic transformations. In general, composition fails because the second transformation applied has repeated variables in some productions. [E.g., $(q, \sigma) \to \sigma((q, x_0), (q', x_0))$.] If the first transformation is nondeterministic, then its random effect on an input tree may be duplicated in two places by the second transformation. Thus, it may be impossible to construct a third transformation which will carry out this behavior all at once.

Example. Let $\Sigma_2 = \{\sigma\}, \Sigma_1 = \{\rho, \tau\}, \Sigma_0 = \{\lambda\}$. T has state set $Q = \{q\}$, and productions

$$(q, \rho(x)) \rightarrow \rho((q, x)) | \tau((q, x))$$
$$(q, \lambda) \rightarrow \lambda.$$

(Let the input set be $\mathscr{T}^0_{\{\rho, \lambda\}}$.) S has states $\{r, s\}$, initial state r, and productions

$$(r, \rho) \rightarrow \sigma((s, x_0), (s, x_0))$$
$$(r, \tau) \rightarrow \sigma((s, x_0), (s, x_0))$$
$$(s, \rho) \rightarrow \rho(s, x_0)$$
$$(s, \tau) \rightarrow \tau(s, x_0)$$
$$(s, \lambda) \rightarrow \lambda.$$

S has the undesirable effect of reproducing the random string produced by T from $\rho^n(\lambda)$. To get rid of this duplication, we make the hypothesis that the second transformation have no repeated variables in its productions. Following category theorists, we therefore have:

Definition. An NDFST L is *linear* if whenever $(q, \sigma) \rightarrow u$ is a production, and (q, x_i) and (s, x_j) occur as indices on u, then $x_i \neq x_j$.

THEOREM 3. *Linear transformations effectively preserve non-deterministic surface sets.*

Discussion. The conjecture that $L \cdot T$ is a nondeterministic transformation is apparently false; see the example of Ogden in Section I (T is also linear in this example). We are forced, therefore, to define first an analogue of totality for nondeterministic transformations. We will replace the given surface set with one generated by a total transformation. Intuitively, totality means that no stage in a derivation is ever blocked.

Definition. Let T be a NDFST, $q \in Q^T$, and $t \in \mathscr{T}^0_\Sigma$. T is *completely defined* on *t starting in state q* if t satisfies the inductive definition

(i) if $t = \lambda \in \Sigma_0$, then there is a production $(q, \lambda) \rightarrow s$ in Π^T.

(ii) If $t = \sigma(t_0, \cdots, t_{n-1})$, then there is a production $(q, \sigma) \rightarrow u$ in Π^T, and for each such production, whenever (r, x_i) is an index on u, then T is completely defined on t_i starting in state r.

We say that T is completely defined on t if the above condition holds for T on t starting in state q for each $q \in Q_0$; and T is completely defined on $\mathscr{R}$ if T is completely defined on t for each $t \in \mathscr{R}$.

It is easy to prove by induction that if T is completely defined on t starting in q, and $(q, t) \Rightarrow^* t'$ where $t' \in \mathscr{T}_\Sigma(Q \times \mathscr{T}^0)$, then if (r, t_i) occurs as an index in t', then there is an $s \in \mathscr{T}^0$ such that $(r, t_i) \Rightarrow^* s$.

Proof of Theorem 3. First we show that, without loss of generality, the first transformation T has a single initial state. Let T_q be the same as T but with initial state $\{q\}$. Now

$$T[\mathscr{R}] = \bigcup_{q \in Q_0} T_q[\mathscr{R}].$$

This implies that

$$L[T[\mathscr{R}]] = \bigcup_{q \in Q_0} L[T_q[\mathscr{R}]].$$

Since surface sets are closed under union (effectively), it suffices to show that $L[T_q[\mathscr{R}]]$ is a surface set. Thus we may assume T has one initial state.

The result is proved in two steps:

LEMMA 1. *Given T and $\mathscr{R}$ we may find effectively T' and $\mathscr{R}'$, where T' has one initial state, such that $T'[\mathscr{R}'] = T[\mathscr{R}]$, $\mathscr{R}'$ is recognizable, and T' is completely defined on $\mathscr{R}'$.*

LEMMA 2. *If T is completely defined on $\mathscr{R}$ and L is linear, then $L[T[\mathscr{R}]]$ is effectively a surface set.*

Proof of Lemma 1. Let Π be the set of productions for T. $\pi \in \Pi$ is *legal* for t in $\mathscr{T}^0$ if π satisfies:

(i) π is legal for $\lambda \in \Sigma_0$ if π is $(q, \lambda) \to s$ for some $q \in Q$;

(ii) π is legal for $\sigma(t_0, \cdots, t_{n-1})$ if π is $(q, \sigma) \to u$, where u is such that whenever (r, x_i) is an index on t', then there is a legal production for t_j with r on its left-hand side.

The set of legal productions for a tree t is exactly the set of productions which can be successfully applied to t yielding a terminal tree. Notice that the definition of legality is really the construction of a finite automaton $\mathscr{A}$ such that

$$\|t\|_{\mathscr{A}} = \{\pi \in \Pi | \pi \text{ is legal for } t\}.$$

(We omit this part of the proof.)

Now we construct the set $\mathscr{R}'$. It will be defined over an extended alphabet Σ'. Let $\hat{\Pi}$ be the power set of the production set Π. Let $\Sigma_n' = \hat{\Pi} \times \Sigma_n$. Now let P be the projection from $\mathscr{T}_{\Sigma'}^0$ to $\mathscr{T}_{\Sigma}^0$ induced by $P(K, \sigma) = \sigma$. P and P^{-1} preserve recognizable sets (Thatcher [16]). Now set

$$\mathscr{R}_1' = \{t \in \mathscr{T}_{\Sigma'}^0 | (\forall t' \leq t)\,(t' = (K, \sigma)\,(t_0, \cdots, t_{n-1}) \text{ implies } K \neq \varnothing \text{ and } K = \{\pi | \pi \text{ is legal for } t\})\}.$$

(Here, $t' \leq t$ means that t' is a subtree of t.) Set

$$\mathscr{R}' = \{t \in \mathscr{R}_1' | P(t) \in \mathscr{R} \cap \text{domain}\,(T)\}.$$

To show that $\mathscr{R}'$ is recognizable, it is sufficient to show that $\mathscr{R}_1'$ is recognizable. To do this, moreover, it is sufficient to show it for $\mathscr{R}_0'$, which has the same definition except that the condition $K \neq \varnothing$ is omitted. (This follows by intersecting $\mathscr{R}_0'$ with a suitable recognizable set.)

But the recognizability of $\mathscr{R}_0'$ follows from the general fact that if $\mathscr{A}$ has state set A, $\Sigma' = A \times \Sigma$, and

$$\mathscr{S} = \{t \in \mathscr{T}_{\Sigma'} | (\forall t' \leq t)\,(t' = (q, \sigma)\,(t_0, \cdots, t_{n-1}) \text{ implies } q = \|P(t')\|_{\mathscr{A}})\},$$

then $\mathscr{S}$ is recognizable. [We construct below an automaton for $\mathscr{S}$. Let $B = A \cup \{\Omega\}$, $\Omega \notin A$. Let

$$\beta_{(q,\lambda)} = \begin{cases} q \text{ if } \alpha_\lambda = q \\ \Omega \text{ otherwise} \end{cases}$$

$$\beta_{(q,\sigma)}\,(q_0, \cdots, q_{n-1}) = \begin{cases} q \text{ if all } q_i \in A \text{ and} \\ \qquad \alpha_\sigma(q_0, \cdots, q_{n-1}) = q \\ \Omega \text{ otherwise.} \end{cases}$$

The inductive statement (which we do not prove) is

$$\|t\|_{\mathscr{A}'} = q \in A \Leftrightarrow (\forall t' \leq t)\,(t' = (r, \sigma)\,(s_0, \cdots, s_{n-1}) \text{ implies } r = \|P(t)\|_{\mathscr{A}} = \|t'\|_{\mathscr{A}'}).$$

The fact follows when we take $B_F = A$.]

Define T' as follows: $Q^{T'} = Q^T$, $q_0^{T'} = q_0^T$, and $(q, (K, \sigma)) \to u$ is a production in $\Pi^{T'}$ if and only if $(q, \sigma) \to u$ is a production in K. We must show that $T'[\mathscr{R}'] = T[\mathscr{R}]$ and that T is completely defined on $\mathscr{R}'$.

Let $(q_0, t) \Rightarrow_{T'}^{*} s$ where $s, t \in \mathscr{T}_{\Sigma'}^{0}$, $t \in \mathscr{R}'$. If we take $P(v)$ where v is a tree in $\mathscr{T}_{\Sigma'}(Q \times \mathscr{T}_{\Sigma'}^{0})$ such that $(q_0, t) \Rightarrow_{T'}^{*} v$ both in the index and the output tree, we obtain immediately a tree derivable (via T) from $(q_0, P(t))$. Thus s can be derived from $(q_0, P(t))$, so $T'[\mathscr{R}'] \subseteq T[\mathscr{R}]$. Conversely let $(q_0, t) \Rightarrow_{T}^{*} s$ be a derivation of a terminal tree s from $t \in \mathscr{R}$. Every step of this derivation is the application of a legal production for the subtree being transformed at that point. Label each node of t with the set of productions legal for the subtree headed by that node; we obtain a tree in $\mathscr{R}'$. We can then mimic the T-derivation with a T'-derivation. Thus, $T'[\mathscr{R}'] \supseteq T[\mathscr{R}]$.

Finally, we prove that T' is completely defined on $\mathscr{R}'$. To do this, let

$$D_q(T) = \{t \in \mathscr{T}^0 \mid (\exists s \in \mathscr{T}^0)\,((q, t) \Rightarrow_{T}^{*} s)\}.$$

We will show by induction that for each $t \in \mathscr{T}_{\Sigma'}^{0}$ and $q \in Q^{T'}$, if $t \in \mathscr{R}_1' \cap P^{-1}[D_q(T)]$, then T' is completely defined on t starting in state q.

Suppose that $t = (K, \lambda) \in \Sigma_0'$, and let q be such that $t \in \mathscr{R}_1' \cap P^{-1}[D_q(T)]$. Then there is a production $\pi \in \Pi^T$: $(q, \lambda) \to s$ and thus $(q, (K, \lambda)) \to s \in \Pi'$. Thus, T' is completely defined on t starting in q.

Now let $t = (K, \sigma)\,(t_0, \cdots, t_{n-1})$ and q be such that $t \in \mathscr{R}_1' \cap P^{-1}[D_q(T)]$. First, we must find a $\pi' \in \Pi'$: $(q, (K, \sigma)) \to u$. Now $Pt \in D_qT$, so we may find a *legal* $\pi \in \Pi'$: $(q, \sigma) \to u$. Thus, π': $(q, (K, \sigma)) \to u$ can be found in Π'. Next, we must show that if (r, x_i) indexes u, then T' is completely defined on t_i starting in r. By inductive hypothesis, we have therefore to show $t_i \in P^{-1}[D_r(T)] \cap R_1'$. But every subtree of a tree in $\mathscr{R}_1'$ is in $\mathscr{R}_1'$, so $t_i \in \mathscr{R}_1'$. Now $P(t) \Rightarrow_{\pi} P(u)$ and π is legal for $P(t)$. Thus there is a terminal tree s_i such that $(r, P(t_i)) \Rightarrow_{T}^{*} s$ because $(r, P(t_i))$ indexes $P(t')$. Thus, $t_i \in P^{-1}[D_r(T)]$. The inductive statement follows, and since $\mathscr{R}' \subseteq \mathscr{R}' \cap P^{-1}[D_{q_0}(T)]$, Lemma 1 is proved.

Proof of Lemma 2. We introduce some notation: If π is a production, let $r\pi$ be the right-hand side and $l\pi$ the left-hand side of π.

Define a new transformation U from L and T as in the proof of Theorem I-2. That is, $Q^U = Q^L \times Q^T$, and for $\pi \in \Pi^T$, $q^L \in Q^L$, define

$$B(\pi, q^L) = \{t' \mid (q^L, r\pi) \Rightarrow_{L}^{*} t'\}.$$

Here we mean $\Rightarrow_{L}^{*}$ in the sense of an L-action on $r\pi$; i.e., if (q^T, x_j) occurs as a variable node on $r\pi$, then $(q^L, (q^T, x_j)) \Rightarrow ((q^L, q^T), x_j)$. Thus, $B(\pi, q^L) \subseteq \mathscr{T}_{\Sigma}(Q^U \times X)$.

Let $((q^L, q^T), \sigma) \to t'$ be a production in Π^U exactly when there is a production $\pi = (q^T, \sigma) \to r\pi$ and $t' \in B(\pi, q^L)$.

ASSERTION. *For $t \in \mathscr{T}^0$, $q^T \in Q^T$, if T is completely defined on t starting in q^T, then*

$$\{s|((q^L, q^T), t) \Rightarrow_U^* s\} = \{s|(\exists w)((q^T, t) \Rightarrow_T^* w \text{ and } (q^L, w) \Rightarrow_L^* s)\}.$$

(Here, s, t, and w are in $\mathscr{T}^0$.) By the assertion, U is the composite $L \cdot T$ when restricted to those inputs for which T is completely defined. The theorem thus follows from the inductive statement.

Proof of the assertion. ($\supseteq$). Proceed by induction. If $t = \lambda \in \Sigma_0$, both sides are equal, by definition, to the union of the $B(\pi, q^L)$ for which q^T occurs in $l\pi$. Suppose the result for $t_0, \cdots, t_{n-1}$; let $t = \sigma(t_0, \cdots, t_{n-1})$. Assume that

$$(\exists w)((q^T, t) \Rightarrow^* w \text{ and } (q^L, w) \Rightarrow^* s).$$

Then there is a $\pi \in \Pi^T$ and

$$(q^T, t) \Rightarrow_\pi r\pi[(q_{i_0}, t_{i_0}), \cdots, (q_{i_{k-1}}, t_{i_{k-1}})]$$

(these are the indices occurring in left-to-right order in the derived tree). Now $(q_{i_0}, t_{i_0}) \Rightarrow_T^* w_0, \cdots, (q_{i_{k-1}}, t_{i_{k-1}}) \Rightarrow_T^* w_{k-1}$. If we apply q^L to $r\pi(w_0, \cdots, w_{k-1})$, we can derive as an intermediate step

$$t'[(q_{j_0}^L, w_{j_0}), \cdots, (q_{j_{p-1}}^L, w_{j_{p-1}})]$$

where $(q^L, r\pi) \Rightarrow_L^* t'$. That is, $t' \in B(\pi, q^L)$ looks like

$$t'[((q_{j_0}^L, q_{j_0}^T), y_0), \cdots, ((q_{j_{p-1}}^L, q_{j_{p-1}}^T), y_{p-1})],$$

where the y's are certain of the x_i's occurring in $r\pi$. If now we take this $t' \in B(\pi, q^L)$ and substitute the correct t_{j_m} for y_m, we know that T is completely defined in state $q_{j_m}^T$ on t_{j_m}. Now $(q_{j_m}^T, t_{j_m}) \Rightarrow_T^* w_{j_m}$, and $(q_{j_m}^L, w_{j_m}) \Rightarrow_L^* s_m$. The inductive hypothesis applies, and so

$$[(q_{j_m}^L, q_{j_m}^T), t_{j_m}] \Rightarrow^* s_m, \quad 0 \le m \le p-1.$$

But

$$v = t'[((q_{j_0}^L, q_{j_0}^T), t_{j_0}), \cdots, ((q_{j_{p-1}}^L, q_{j_{p-1}}^T), t_{j_{p-1}})]$$

is derivable from t in the system U. We conclude that since

$$t'((q_{j_0}^L, w_{j_0}), \cdots, (q_{j_{p-1}}^L, w_{j_{p-1}})) \Rightarrow_L^* s,$$

then, in fact, $v \Rightarrow_U^* s$. The inclusion thus holds in the inductive case.

Now for the other inclusion ($\subseteq$) we need to use complete definability and linearity at essential points. Again proceed by induction; the basis holds, so let $t = \sigma(t_0, \cdots, t_{n-1})$. Suppose T is completely defined on t starting in q^T, and

$$((q^L, q^T), t) \Rightarrow_U^* s, \quad s \in \mathscr{T}^0.$$

We must show that there is a w such that $(q^T, t) \Rightarrow_T^* w$ and $(q^L, w) \Rightarrow_L^* s$.

Apply one step in the derivation of s in the system U from t. We obtain

$$t'[((q_{j_0}^L, q_{j_0}^T), t_{j_0}), \cdots, ((q_{j_{p-1}}^L, q_{j_{p-1}}^T), t_{j_{p-1}})],$$

where t' comes from the right-hand side $r\pi$ of some $\pi \in \Pi^T$, so that $(q^L, r\pi) \Rightarrow_L^* t'$, and q^T occurs in $l\pi$. Notice that T is completely defined on t_{j_0}, starting in $q_{j_0}^T$, because the pair $(q^T_{j_0}, t_{j_0})$ occurs in $r\pi$ and T is completely defined on t starting in q^T. Now let

$$(q^T, t) \Rightarrow_\pi r\pi[(q^T_{i_0}, t_{i_0}), \cdots, (q^T_{i_{k-1}}, t_{i_{k-1}})].$$

Number the index positions $0, 1, \cdots, k-1$. Similarly in $t'[(q^L_{j_0}, q^T_{j_0}), t_{j_0}, \cdots, (q^L_{j_{p-1}}, q^T_{j_{p-1}}), t^L_{j_{p-1}}]$, number the indices $0, 1, \cdots, p-1$. Linearity of L guarantees that there is a subset A of $\{0, \cdots, k-1\}$ and a bijection $f: \{0, \cdots, p-1\} \rightarrowtail A$ such that if $((q^L_{j_m}, q^T_{j_m}), t_{j_m})$ occurs at the m^{th} place in t', then $(q^T_{j_m}, t_{j_m})$ occurs at the $f(m)^{th}$ place in $r\pi$. Although this is an inductive lemma in itself, its proof should be clear because as far as variables go, L can only permute them or drop them entirely. (Thus $p \leq k$.) We will use f to construct the tree w needed to establish the result.

Let

$$((q^L_{j_0}, q^T_{j_0}), t_{j_0}) \Rightarrow^*_U s_0, \cdots, ((q^L_{j_{p-1}}, q^T_{j_{p-1}}), t_{j_{p-1}}) \Rightarrow^*_U s_{p-1}.$$

Thus $s = t'[s_0, \cdots, s_{p-1}]$. Since T is completely defined on t_{j_m} starting in $q^T_{j_m}$, the hypothesis applies, giving for each m a tree $w_m \in \mathcal{T}^0$, such that

$$(q^T_{j_m}, t_{j_m}) \Rightarrow^*_T w_m \quad \text{and} \quad (q^L_{j_m}, w_m) \Rightarrow^* s_m.$$

Now in $r\pi[(q^T_{i_0}, x_{i_0}), \cdots, (q^T_{i_{k-1}}, x_{i_{k-1}})]$ substitute w_m for $(q^T_{i_{f(m)}}, x_{i_{f(m)}})$. At the other positions, say $(q^T_{i_l}, x_{i_l})$, we know that there is a tree $\bar{w}_l$ so that $(q^T_{i_l}, t_{i_l}) \Rightarrow^*_T \bar{w}_l$. This follows, because T is completely defined on t starting in q^T. Substitute $\bar{w}_l$ for the positions $(q^T_{i_l}, x_{i_l})$. We obtain a tree $w \in \mathcal{T}^0$, and clearly $(q^T, t) \Rightarrow^* w$. Also,

$$(q^L, w) \Rightarrow^*_L t'[(q^L_{j_0}, w_0), \cdots, (q^L_{j_{p-1}}, w_{p-1})]$$

because $w = r\pi[w_{i_0}, \cdots, w_{i_{k-1}}]$ and $(q^L, r\pi) \Rightarrow^*_L t'$; the definition of f ensures that $(q^T_{i_{f(m)}}, t_{i_{f(m)}}) \Rightarrow^*_T w_m$, because this pair occurs at the mth place in t'. Thus during the L-derivation from w it must occur at the mth place as well. We see that

$$\begin{aligned}(q^L, w) &\Rightarrow^* t'[(q^L_{j_0}, w_0), \cdots, (q^L_{j_{p-1}}, w_{p-1})] \\ &\Rightarrow^* t'[s_0, \cdots, s_{p-1}].\end{aligned}$$

But

$$((q^L, q^T), t) \Rightarrow^*_U t'[(q^L_{j_0}, q^T_{j_0}), t_{j_0}, \cdots, ((q^L_{j_{p-1}}, q^T_{j_{p-1}}), t_{j_{p-1}})],$$

and

$$((q^L_{j_m}, q^T_{j_m}), t_{j_m}) \Rightarrow^*_U s_m, \quad 0 \leq m \leq p-1.$$

Hence, when $((q^L, q^T), t) \Rightarrow^*_U s = t'[s_0, \cdots, s_{p-1}]$, then we have $(q^T, t) \Rightarrow^* w$ and $(q^L, w) \Rightarrow^*_L s$. This completes the proof of the lemma and the theorem.

We are in a position to investigate further properties of surface dendrolanguages. Notice that Theorem II-3 is effective.

COROLLARY. *The class of surface dendrolanguages is effectively closed under intersection with recognizable sets.*

Proof. Let $\mathcal{R}$ be recognizable, and let $T[\mathcal{R}']$ be a surface set. By the proof of Theorem 2 there is a linear transformation L which is a partial identity on $\mathcal{R}$. Hence $t \in L[T[\mathcal{R}']]$ if and only if $t \in T[\mathcal{R}'] \cap \mathcal{R}$.

COROLLARY. *Surface dendrolanguages form a subclass of the recursive dendrolanguages over Σ.*

Proof. Given $T[\mathscr{R}]$ and $t \in \mathscr{T}_\Sigma^0$, the set $\{t\}$ is recognizable. Now

$$t \in \mathscr{R} \Leftrightarrow T[\mathscr{R}] \cap \{t\} = \varnothing.$$

By the previous corollary, $T[\mathscr{R}] \cap \{t\}$ is a surface set $U[\mathscr{S}]$. But $U[\mathscr{S}] = \varnothing$ if and only if

$$\text{domain } (U) \cap \mathscr{S} = \varnothing.$$

Domain $(U) \cap \mathscr{S}$ is recognizable, so the result follows.

To prove further properties, we need to define the set of *paths* through a tree. Given a ranked alphabet Σ, let Σ' be the alphabet (Σ, r'), where

$$r'(\sigma) = \begin{cases} 0 \text{ if } \sigma \in \Sigma_0 \\ 1 \text{ otherwise.} \end{cases}$$

Definition. Given $t \in \mathscr{T}_\Sigma^0$, $P(t)$, the set of *paths* through t, is the subset of $\mathscr{T}_{\Sigma'}^0$, defined inductively by

$$P(\lambda) = \{\lambda\}, \quad \lambda \in \Sigma_0;$$

$$P(\sigma(t_0, \cdots, t_{n-1})) = \bigcup_{i=0}^{n-1} \{\sigma(w) | w \in P(t_i)\}.$$

If $\mathscr{R} \subseteq \mathscr{T}_\Sigma^0$, then

$$P[\mathscr{R}] = \bigcup_{t \in \mathscr{R}} P(t).$$

P is clearly definable as a linear nondeterministic transformation. Hence:

COROLLARY. *If $\mathscr{S}$ is a surface set, then so is $P[\mathscr{S}]$ (effectively).*

A similar result holds for recognizable sets:

PROPOSITION. *If $\mathscr{R}$ is recognizable, then so is $P[\mathscr{R}]$.*

Proof. Let $\mathscr{A} = (A, \alpha, A_F)$ recognize $\mathscr{R}$. Let $B \subseteq A$ be the set

$$\{q \in A | (\exists t \in \mathscr{T}^0) (\|t\|_{\mathscr{A}} = q)\}.$$

B is effectively calculable from $\mathscr{A}$ using the solvability of the emptiness problem for the sets accepted by automata $\mathscr{A}^q$ which have final states $A_F^q = \{q\}$ but are otherwise the same as $\mathscr{A}$. Let $x \in \mathscr{T}_{\Sigma'}^0$. Construct an automaton (over Σ') $\mathscr{M} = (2^Q, \mu, F)$ such that

(*) $\qquad q \in \|x\|_{\mathscr{M}}$ if and only if $(\exists t) (\|t\|_{\mathscr{A}} = q$ and $x \in P(t))$.

To do this, let $\mu_\lambda = \{\alpha_\lambda\}$ for $\lambda \in \Sigma_0$. If $\sigma \in \Sigma_n$, define μ_σ as follows: Choose $i \leq n-1$. Define

$$\mu_\sigma^{(i)}(Q) = \{\alpha_\sigma(q_0, \cdots, q_i, \cdots, q_{n-1}) | q_i \in Q, q_j \in B\}$$

$$\mu_\sigma(Q) = \bigcup_{i \leq n-1} \mu_\sigma^{(i)}(Q).$$

One verifies that (*) holds with this μ. If we let $F = \{Q \subseteq A, Q \cap A_F \neq \varnothing\}$, then the lemma follows by (*).

A ranked alphabet is *monadic* if $\Sigma_n = \varnothing$ for $n > 1$. We now prove

THEOREM 4. *If $T[\mathscr{R}]$ is contained in $\mathscr{T}_\Sigma^0$, Σ monadic, then $T[\mathscr{R}]$ is (effectively) recognizable.*

Proof. We may certainly assume that $T[\mathscr{R}]$ is the range of a transformation T whose productions have right-hand sides which are monadic. T is thus linear, and will choose certain paths through each tree in $\mathscr{R}$ as important input. We may define for $q \in Q^T$, and any t, the set $P(T, q)(t)$ of paths chosen by T starting in state q:

If $t = \lambda$, then $P(T, q)(t)$ contains λ exactly when there is a production $(q, \lambda) \to w$ in Π^T;

If $t = \sigma(t_0, \cdots, t_{n-1})$, then $P(T, q)(t)$ contains w if and only if $w = \sigma(w')$, and there is a production $(q, \sigma) \to u(r, x_j)$ in Π^T such that $w' \in P(T, r)(t_j)$. We assert that for each q,

$$\{P(T, q)(t) | t \in \mathscr{T}_\Sigma^0\} = H_q$$

is a recognizable subset of $\mathscr{T}_{\Sigma'}^0$.

We will not give the full proof, but an automaton $\mathscr{A}$ can be easily constructed such that

$$\|w\|_{\mathscr{A}} = \{r | (\exists t \in \mathscr{T}^0)(w \in P(T, r)(t))\}.$$

Taking $A_F = \{Q | q \in Q\}$, we find that $\mathscr{A}$ recognizes the asserted set H_q of paths. Now let $\mathscr{R}$ be the given recognizable set. Then $H_{q_0} \cap P[\mathscr{R}]$ is a recognizable subset (effectively) of $\mathscr{T}_{\Sigma'}^0$. T itself defines a nondeterministic finite state mapping of *strings* in this set. Such maps preserve recognizable sets, and so

$$T[\mathscr{R}] = T[H_{q_0} \cap P[\mathscr{R}]]$$

is recognizable.

COROLLARY. *The infiniteness problem for the class of surface sets is effectively solvable.*

Proof. If $\mathscr{S}$ is a surface set, $P[\mathscr{S}]$ is infinite if and only if $\mathscr{S}$ is. But $P[\mathscr{S}]$ is a recognizable set of strings effectively obtainable from $\mathscr{S}$. The infiniteness problem for such sets is decidable.

COROLLARY. *The class of surface sets is not closed under intersection.*

Proof. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_2 = \{\rho\}$, $\Sigma_1 = \{\sigma, \tau\}$, $\Sigma_0 = \{\lambda\}$. Define $\sigma^i(x) = \sigma(x)$, $\sigma^{i+1}(x) = \sigma(\sigma^i(x))$. Put

$$\mathscr{S}_1 = \{\rho(\sigma^j(\lambda), \tau^i(\sigma^j(\lambda))) | i, j \geq 1\}$$
$$\mathscr{S}_2 = \{\rho(\sigma^j(\lambda), \tau^j(\sigma^i(\lambda))) | i, j \geq 1\}.$$

$\mathscr{S}_1$ and $\mathscr{S}_2$ can both be obtained as surface sets (proof omitted), but

$$P(\mathscr{S}_1 \cap \mathscr{S}_2) = \{\rho(\sigma^j(\lambda))\} \cup \{\rho(\sigma^j(\tau^j(\lambda)))\}$$

is not recognizable.

Now we give an example of an undecidable problem.

COROLLARY. *There is no decision procedure for determining whether the intersection of two surface sets is empty or not. (Surface sets here mean over arbitrary finite ranked alphabet.)*

Proof. Let $\Sigma_0 = \{\lambda\}$, $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{\sigma\}$. We can naturally interpret an element of $\mathscr{T}_{\Sigma_0 \cup \Sigma_1}$ as a string in Σ_1^*, and conversely. Let $(\alpha_0, \cdots, \alpha_m)$ and $(\beta_0, \cdots, \beta_m)$ be two sequences of words from Σ_1^*. The Post correspondence problem for these sequences is to determine whether there is a sequence of integers $i_0, \cdots, i_k$ such that

$$\alpha_{i_0} \cdots \alpha_{i_k} = \beta_{i_0} \cdots \beta_{i_k}.$$

(There is no algorithm to solve the Post problem for arbitrary pairs of sequences $((\alpha_0, \cdots, \alpha_m), (\beta_0, \cdots, \beta_m))$ of words over Σ_1^*.)

Let $L = \{a^{i_0}ba^{i_1} \cdots ba^{i_k} | i_j \in \{0, \cdots, m\}\}$. Let $w = a^{i_0}b \cdots ba^{i_k}$ and define $\Phi(w) = \alpha_{i_0} \cdots \alpha_{i_k}$. It is easy to construct Φ as a finite-state mapping of strings. Similarly, let $\Psi(w) = \beta_{i_0} \cdots \beta_{i_k}$. Since Φ and Ψ are finite-state functions, the sets $\mathscr{S} = \{\sigma(w, \Phi(w)) | w \in L$ and $\mathscr{S}' = \{\sigma(w, \Psi(w)) | w \in L\}$ are surface sets. But $\mathscr{S} \cap \mathscr{S}' = \varnothing$ if and only if $\exists w \in L$ with $\Phi(w) = \Psi(w)$. Thus, $\mathscr{S} \cap \mathscr{S}' = \varnothing$ if and only if there is a solution for the given correspondence problem.

Other problems are shown unsolvable in [14]; for example, equality of surface sets and whether a surface set is recognizable.

3. Translations and Target Languages.

Let $(\mathscr{R}, T)$ be a transformational system. The translation defined by $(\mathscr{R}, T)$ is the set

$$\{(y(s), y(t)) | (s, t) \in T, s \in \mathscr{R}\}.$$

(Recall that $T = \{(s, t) | s \Rightarrow_T^* t\}$.)

By Theorem 1 the domain of a NDFS translation is a context-free language. We again wish to consider the ranges of such translations, because of their importance for transformational grammars. Define a target language as the range of a translation. An immediate question a grammarian asks is: "Are target languages recursive sets?" For NDFS target languages, the answer is yes, and the proof is elegant.

LEMMA 1. *The emptiness problem for the class of NDFS target languages is effectively solvable.*

Proof. Let $L = y[T[\mathscr{R}]]$. $L = \varnothing$ if and only if $T[\mathscr{R}] = \varnothing$. Whether $T[\mathscr{R}] = \varnothing$ is solvable.

LEMMA 2. *Let K be an ordinary regular subset of Σ_0^*. Then $y^{-1}[K]$ is (effectively) a recognizable subset of $\mathscr{T}_\Sigma^0$.*

Proof. Let $(Q, \Sigma_0, \delta, q_0, F)$ recognize K. For $w \in \Sigma_0^*$ let $\delta_w(q) = \delta(q, w)$. Remark: $\delta_{xy} = \delta_y \cdot \delta_x$ for all $x, y \in \Sigma_0^*$.

Now define a Σ-automaton $\mathscr{A}$ by setting

$$A = \{\varphi | \varphi: Q \to Q\};$$

$$\alpha_\lambda = \delta_\lambda \text{ for } \lambda \in \Sigma_0 \text{ (λ is \textit{not} the empty string)};$$

$$\alpha_\sigma(\varphi_0, \cdots, \varphi_{n-1}) = \varphi_{n-1} \cdot \cdots \cdot \varphi_0.$$

By our first remark, for all $t \in \mathscr{T}_\Sigma^0$, $q \in Q$

$$\|t\|_{\mathscr{A}}(q) = \delta_w(q), \text{ where } w = y(t).$$

Hence, if $A_F = \{\varphi | \varphi(q_0) \in F\}$, then $\mathscr{A}$ accepts $y^{-1}[K]$.

THEOREM 5. *The class of target languages is closed (effectively) under intersection with regular sets.*

Proof. Let $L = y[T[\mathscr{R}]]$ and let K be regular. Then, $K \cap L = y[T[\mathscr{R}] \cap y^{-1}[K]]$. But by the corollary to Theorem 3, $T[\mathscr{R}] \cap y^{-1}[K]$ is a surface set (effectively). The result follows.

COROLLARY. *Target languages are recursive.*

Proof. Let L be a target language and $w \in \Sigma_0^*$. Then $w \in L$ if and only if $\{w\} \cap L = \varnothing$; apply Theorem 5 and Lemma 1.

Notice that Lemma 2 provides an easy proof of the fact that context-free languages are closed under intersection with regular sets. (Use the technique of Theorem 5.)

Finally, as a special result, we demonstrate that the infiniteness problem for the class of target languages is solvable.

We say a tree is a *fan* if no nodes of rank 1 occur in it. We can always prune the nodes of rank 1 from a tree withut changing its yield. Formally:

$$\begin{aligned} \text{fan}\,(\lambda) &= \lambda, \quad \lambda \in \Sigma_0 \\ \text{fan}\,\rho(t) &= t, \quad \rho \in \Sigma_1 \\ \text{fan}\,\sigma(t_0, \cdots, t_n) &= \sigma(\text{fan}\,(t_0), \cdots, \text{fan}\,(t_n)), \quad n > 0. \end{aligned}$$

A tree is a *fan* if fan $(t) = t$. Also, $y(\text{fan}\,(t)) = y(t)$, and fan is a linear FST. If $\mathscr{S}$ is a surface set then so is fan $[\mathscr{S}]$, and $y[\text{fan}\,[\mathscr{S}]] = y[\mathscr{S}]$.

THEOREM 6. *The infiniteness problem for the class of target languages is solvable.*

Proof. Let $L = y[\mathscr{S}]$. Then $L = y[\text{fan}\,[\mathscr{S}]]$. L is infinite if and only if fan $[\mathscr{S}]$ is infinite, as an easy counting argument shows.

4. A Simple Extension of the Nondeterministic Model. When carrying out a transformational derivation, one checks trees to see whether or not transformations apply. For example, a transformation which changes sentences to the passive voice applies only to structures of the form "noun phrase—verb—noun phrase)".

Our transformations, as defined, do not have this checking ability, because only one node at a time is read and transformed. In the example just described, however, we are required to check the level of nodes *NP-V-NP* below the top node *S* of the input tree. In other examples, a structural condition may have to be satisfied which could occur at any level in the input tree.

To remedy (partially) this defect in the basic model, we may modify our productions. We give them a look-ahead capacity—the local output tree (right-hand side) will depend on the state, the symbol being read and transformed, and a specified number of look-ahead symbols, arranged in a tree form.

The productions will have the form

$$(q, \sigma(s_0, \cdots, s_{n-1})) \to u,$$

where $s_i \in \mathscr{T}_\Sigma(X)$, $u \in \mathscr{T}_\Sigma(Q \times X^n)$, $\sigma \in \Sigma_n$. A production will apply to $\sigma(t_0, \cdots, t_{n-1})$ if each s_i occurs at the top of t_i. [This can be formalized as a definition.] The result of application will be the tree in $\mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma^0)$ obtained by substituting (q', t_j) for each pair (q', x_j) occurring in u.

An FS transformation *with templates* is a transformation with productions like the above. The extended definition provides a limited look-ahead capability for nondeterministic mappings. One can prove, however, that if $(\mathscr{R}, T)$ is a transformational system, where T has templates, then $T[\mathscr{R}]$ is an ordinary surface set. The idea is to use the transitions of T in a nondeterministic mapping U which guesses that the template expected by T will actually appear. If this is the case, U performs the action of T; if not, U becomes undefined. Details are omitted.

5. Creative Grammars on Trees.

5. Creative Grammars on Trees. We turn now to a new type of production which will grow input trees to be processed as well as read and destroy input nodes. One system using these productions provides an extension of context-free grammars to trees. Brainerd [6] has considered regular tree grammars; his definition can be subsumed here.

Consider an FS index-erasing production; for example

$$\begin{tikzcd}
 & & \sigma \arrow[dl, dash] \arrow[dr, dash] & \\
q\,\sigma(x_1, x_2) \arrow[r] & \rho \arrow[d, dash] & & (r, x_2) \\
 & (q, x_1) & &
\end{tikzcd}$$

Figure 1.

Here, the next states occur as labels on the variables at the bottom of the output tree. Another possibility, however, would be to allow productions like

$$\begin{tikzcd}
 & & \sigma \arrow[dl, dash] \arrow[dr, dash] & \\
(q, \sigma(x_1, x_2)) \arrow[r] & (q, \rho) \arrow[d, dash] & & (r, x_2) \\
 & x_1 & &
\end{tikzcd}$$

Figure 2.

or even

$$\begin{tikzcd}
 & & (q, \sigma) \arrow[dl, dash] \arrow[dr, dash] & \\
(q, \sigma(x_1, x_2)) \arrow[r] & \rho \arrow[d, dash] & & x_2 \\
 & x_1 & &
\end{tikzcd}$$

Figure 3.

In the first of these cases, we would operate next on the tree x_1 starting in state r, and on the tree $\rho(x_2)$ starting in state q. In the second case, the next operation would be performed on $\sigma(\rho(x_1), x_2)$ starting in state q.

This idea lets us define a new operation on trees (which may be nondeterministic). If we select a starting configuration, it may be possible to grow index trees nondeterministically *ad infinitum* before the application of index-erasing productions takes place. We will call the new productions *index-creating*. In the first example of an index-creating production, no new input was actually created; the state q remained stationary. This, of course, is the analogue of a pointer remaining stationary in an input string. The creation of a new index in the second example is *not* the analogue of moving backward in the input string but, of using the input string both as a push-down memory and as an input.

Example. Consider the monadic productions

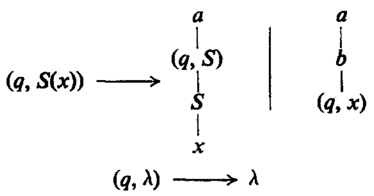

Figure 4.

and the starting configuration $(q, S(\lambda))$. This system produces a dendrolanguage which can be identified with $L(G)$, where G is a CFG with productions

$$S \to aSS | ab.$$

Notice, however, that derivations in the tree case correspond to left-to-right derivations in the grammar. As is well known, there is no loss of generality in doing left-to-right derivations exclusively in a grammar. We shall not prove it, but this property is also true for a class of tree grammars.

One more word—we shall not use creative productions to define mappings. We shall fix one configuration to start from, and will consider sequences of productions which from this configuration eventually produce state-free trees or terminal trees. Thus we are really doing grammars.

Consider the whole tree t which may occur labeled by a state q on the right-hand side of some creative production. The pair (q, t) may itself be considered an index. If $t = x \in X$, then we get an index in the old sense. The new index set is, however, $Q \times \mathscr{T}_\Sigma(X)$, instead of $Q \times X$. Formally:

Definition. A pair $(q, \sigma(x_0, \cdots, x_{n-1})) \to u$ is an *index-creating production* if $u \in \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma(X_n))$.

Definition. For Π a given set of creative productions, and $t \in \mathscr{T}_\Sigma(Q \times \mathscr{T}^0_\Sigma)$, the set of trees t' directly generated by t is defined inductively:

(i) if $t \in \Sigma_0$, then $\{t' | t \Rightarrow t'\} = \varnothing$;

(ii) if $t = (q, \bar{t})$, there are 2 cases depending on the form of $\bar{t}$:

(a) if $\bar{t} \in \Sigma_0$, then there is some production $(q, \bar{t}) \to u$ in π; and $t' = u$;

(b) if $\bar{t} = \sigma(\bar{t}_0, \cdots, \bar{t}_{n-1})$, then there is a production $(q, \sigma) \to u$ in π and t' is obtained from u by substituting $\bar{t}_j$ for x_j, $j = 0, \cdots, n-1$, whenever x_j occurs in an index of u;

(iii) if $t = \sigma(t_0, \cdots, t_{n-1})$, then there is an $i < n$ such that t_i generates t'_i and $t' = \sigma(t_0, \cdots, t'_i, \cdots, t_{n-1})$.

At this point we had better say something about substitution as mentioned in part (b) of the last definition. We shall give a formal definition and use it later to prove a result. This definition can also be used to justify formally what we said in previous sections.

Definition. Let $u \in \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma(X_m))$ and let $(s_0, \cdots, s_{m-1})$ be a fixed sequence of terms. The function $\mathrm{Sub}(\substack{s_0 \cdots s_{m-1} \\ x_0 \cdots x_{m-1}}|u)$, or $S(\substack{s_i \\ x_i}|u)$ is defined by induction on u:

(i) if $u = \lambda$, $S(\substack{s_i \\ x_i}|u) = \lambda$;

(ii) if $u = (q, \lambda)$, $S(\substack{s_i \\ x_i}|u) = (q, \lambda)$;
if $u = (q, x_j)$, $S(\substack{s_i \\ x_i}|u) = (q, s_j)$
if $u = (q, \rho(t_0, \cdots, t_m))$, then
$S(\substack{s_i \\ x_i}|u) = (q, \rho(S(\substack{s_i \\ x_i}|t_0), \cdots, S(\substack{s_i \\ x_i}|t_m)))$;

(iii) if $u = \sigma(u_0, \cdots, u_{k-1})$, then
$S(\substack{s_i \\ x_i}|u) = \sigma(S(\substack{s_i \\ x_i}|u_0), \cdots, S(\substack{s_i \\ x_i}|u_{k-1}))$.

Definition. A top-down creative tree grammar over Σ is a tuple (Σ, Q, S, Π) where Q is a set of states, Π is a set of index-creating productions over Q and Σ, and S is a *finite* subset of $\mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma^0)$ (the starting configurations).

As before, let $\Rightarrow^*$ be the reflexive, transitive closure of the direct generation relation.

Definition. For G a tree grammar, the dendrolanguage generated by G is the set

$$\mathscr{L}(G) = \{t \in \mathscr{T}_\Sigma^0 | (\exists s \in S) s \Rightarrow^* t\}.$$

Example. Let $\Sigma_4 = \{C\}$, $\Sigma_2 = \{B\}$, $\Sigma_1 = \{H\}$, $\Sigma_0 = \{a\}$. Let $Q = \{q_0, q_1, q_2\}$ and let the initial configuration be $(q_0, H(a))$. We have the productions

$$(q_0, H(x)) \to (q_0, H(H(x))) | C((q_1, x), (q_2, x), (q_2, x), a)$$
$$(q_1, H(x)) \to C((q_1, x), (q_2, x), (q_2, x), a)$$
$$(q_2, H(x)) \to B(a, (q_2, x))$$
$$(q_i, a) \to a \qquad (i = 0, 1, 2).$$

Applying the first production, we derive a "string" $(q_0, H^n(a))$, $n \geq 1$. We then apply index-erasing productions which at each level add $2n+1$ a's to the yield. The yield of the resulting tree is a^{n^2}.

The index-erasing productions in this grammar correspond to the application of the recursion equation

$$f(n+1) = f(n) + 2n + 1.$$

If $P_i \in \mathbb{N}[X]$ is a polynomial, and if $k_i \in \mathbb{N}$, then the language $\{a^{\varphi(n)}| \geq 1\}$ where $\varphi(x) = \Sigma_{i=1}^{j} p_i(x) k_i^x$, can be obtained as the yield of a creative dendrolanguage. A grammar for such a dendrolanguage would employ a state q_f for each function f in a system of recursion equations needed to describe φ. The following theorem, therefore, may be surprising.

THEOREM 7. *Every creative dendrolanguage can be generated by a one-state creative dendrogrammar.*

Proof. The problem with reducing many states to one is that during application of index-erasing productions, index subtrees may be duplicated and then processed in different ways. The index subtrees, however, are obtained from the starting trees by application of creative productions. Therefore, when a new index symbol is created, we must take into account the possible states in which it could be read off by an index-erasing production. The creative productions in a grammar will therefore be modified to encode state-transitions in their indices. If Q is the set of states in the original grammar, the new index labels will be of the form $\sigma^{(q)}$ where $\sigma \in \Sigma$, and $q \in Q$. The rank of $\sigma^{(q)}$ will also be changed. If Q has p states, ($p \geq 2$), then $r'(\sigma^{(q)}) = p \cdot r(\sigma)$. For notational convenience, we will relabel variables as follows:

$$x_i^{(q)} = x_{pi+q-1}$$

where $q \in Q = \{1, \cdots, p\}$.

Thus, if $H \in \Sigma$ has old rank 1, then $H^{(q)}$ will have new rank p; selection by the new grammar of $x_j^{(i)}$ occurring as an index on $H^{(q)}$ will correspond, in the old grammar, to selecting x_i and going to state j.

As illustrations, let us encode some productions. Suppose $Q = \{1, 2, 3\}$. Let

$$(1, H(x_0)) \to (2, H(H(x_0)) = \pi$$

be an old production. Let $\cdot$ be the single state of the new grammar. We rewrite π as shown in Figure 5.

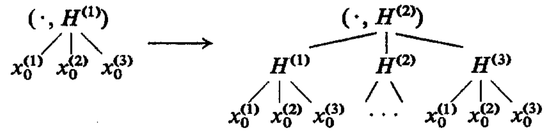
$(\cdot, H^{(1)})$
$x_0^{(1)}$ $x_0^{(2)}$ $x_0^{(3)}$
$(\cdot, H^{(2)})$
$H^{(1)}$ $H^{(2)}$ $H^{(3)}$
$x_0^{(1)}$ $x_0^{(2)}$ $x_0^{(3)}$ $\cdots$ $x_0^{(1)}$ $x_0^{(2)}$ $x_0^{(3)}$

Figure 5.

Next, suppose

$$(3, H(x_0)) \to C((2, x_0), (2, x_0), (3, x_0))$$

is another production. Its encoding would be

$$(\cdot, H^{(3)}(x_0^{(1)}, x_0^{(2)}, x_0^{(3)})) \to C((\cdot, x_0^{(2)}), (\cdot, x_0^{(2)}), (\cdot, x_0^{(3)})).$$

Finally, if

$$(2, K(x_0, x_1)) \to C((1, x_0), (3, x_1), (2, x_1))$$

is a production, its encoding is

$$(\cdot, K^{(2)}(x_0^{(1)}, x_0^{(2)}, x_0^{(3)}, x_1^{(1)}, x_1^{(2)}, x_1^{(3)})) \to C((\cdot, x_0^{(1)}), (\cdot, x_1^{(3)}), (\cdot, x_1^{(2)})).$$

We can now proceed with the proof. We must encode productions whose right-hand sides are elements of $\mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma(X_n))$. Let $\Delta = \{\sigma^{(q)} | \sigma \in \Sigma, q \in Q\}$. We first encode members of $\mathscr{T}_\Sigma(X_n)$ into $\mathscr{T}_\Delta(X_{p\cdot n})$; this is done as follows. Let $q \in Q = \{1, \cdots, p\}$. Define maps e^q: $\mathscr{T}_\Sigma(X_n) \to \mathscr{T}_\Delta(X_{p\cdot n})$ by simultaneous induction:

$$e^q(\lambda) = \lambda^{(q)};$$
$$e^q(x_j) = x_j^{(q)};$$
$$e^q(\rho(s_0, \cdots, s_m)) = \rho^{(q)}(e^1(s_0), \cdots, e^p(s_0), \cdots, e^1(s_m), \cdots, e^p(s_m)).$$

Now we can encode any $u \in \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma(X_n))$. Define

$$\bar{e}(\lambda) = \lambda;$$
$$\bar{e}((i, s)) = (\cdot, e^i(s));$$
$$\bar{e}(\sigma(t_0, \cdots, t_{n-1})) = \sigma(\bar{e}(t_0), \cdots, \bar{e}(t_{n-1})).$$

Here, $\cdot$ is the unique new state; and so $\bar{e}: \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma(X_n)) \to \mathscr{T}_\Sigma(\{\cdot\} \times \mathscr{T}_\Delta(X_{pn}))$. One proves with a tedious but straightforward argument by induction on u, using our previous definition of substitution, that for fixed $s_0, \cdots, s_m \in \mathscr{T}_\Sigma(X_{m+1})$ and any $u \in \mathscr{T}_\Sigma(Q \times \mathscr{T}_\Sigma(X_{m+1}))$ that

$$(*) \qquad \bar{e}(S(\tbinom{s_j}{x_j}|u)) = S(\tbinom{e^i(s_j)}{x_j^{(i)}}|\bar{e}(u))$$

where $1 \leq i \leq p$, $0 \leq j \leq m$. Also (by induction) $\bar{e}$ is a one-to-one function; and if $t \in \mathscr{T}^0_\Sigma$, then $\bar{e}(t) = t$.

Now let $G' = (\Sigma \cup \Delta, \{\cdot\}, \bar{e}[S_0], \Pi')$ where $G = (\Sigma, Q, S_0, \Pi)$. If $\pi \in \Pi$, say $(q, \sigma(x_0, \cdots, x_{n-1}) \to u)$, let $\bar{e}(\pi)$ be the production

$$(\cdot, \sigma^{(q)}(x_j^{(i)}) \to \bar{e}(u)).$$

Set $\Pi' = \{\bar{e}(\pi) | \pi \in \Pi\}$.

Let $t \Rightarrow_\pi t'$ mean that t directly generates t' by application of the production π. Assertion: For any $t \in \mathscr{T}_\Sigma(Q \times \mathscr{T}^0_\Sigma)$, if

$$\bar{e}(t) \Rightarrow_{\bar{e}(\pi)} w,$$

then there is a $t' \in \mathscr{T}_\Sigma(Q \times \mathscr{T}^0_\Sigma)$ such that

$$t \Rightarrow_\pi t' \text{ and } w = \bar{e}(t').$$

The theorem follows from this assertion, because $\bar{e}$ is 1-1 and if $t' \in \mathscr{T}^0_\Sigma$, then $\bar{e}(t') = t'$. Thus, derivations correspond exactly in both grammars.

The proof of the assertion is by induction on t, and has three main cases:

(i) If $t = \lambda \in \Sigma_0$. the assertion is vacuously true.

(ii) If $t = (q, s)$ where $s \in \mathscr{T}^0$, then two subcases arise:

(a) $s = \lambda \in \Sigma_0$. Then $\bar{e}(t) = (\cdot, e^q(\lambda))$. Now $\bar{e}(\pi)$ must be $((\cdot, e^q(\lambda)) \to \bar{e}(u))$, where $u \in \mathscr{T}^0$. Taking $t' = u$ satisfies the assertion.

(b) $s = \rho(s_0, \cdots, s_m)$. Then,

$$\bar{e}(t) = (\cdot, \rho^{(q)}(s_j^{(i)})), \quad 0 \leq j \leq m, \quad 1 \leq i \leq p.$$

Thus,

$$\bar{e}(\pi) = (\cdot, \rho^{(q)}(x_j^{(i)})) \to \bar{e}(u)$$

and so

$$\pi = (q, \rho(x_0, \cdots, x_m) \to u).$$

Now by hypothesis of the assertion

$$\bar{e}(t) \Rightarrow_{\bar{e}(\pi)} w = S(\tbinom{e^i(s_j)}{x_j^{(i)}}|\bar{e}(u))$$

and we know that

$$t \Rightarrow_\pi t' = S(\tbinom{s_j}{x_j}|u).$$

By (*) above, $\bar{e}(t') = w$, so the assertion holds in case (ii).

(iii) $t = \sigma(t_0, \cdots, t_{n-1})$ and the assertion holds for $t_0, \cdots, t_{n-1}$. Now

$$\bar{e}(t) = \sigma(\bar{e}(t_0), \cdots, \bar{e}(t_{n-1})).$$

Since $\bar{e}(t) \Rightarrow_{\bar{e}(\pi)} w$, there must be $i<n$ so that $\bar{e}(t_i) =_{\bar{e}(\pi)} w_i$, and $w = \sigma(\bar{e}(t_0, \cdots, w_i, \cdots, \bar{e}(t_{n-1}))$. By inductive hypothesis, $t_i \Rightarrow t_i'$ and $\bar{e}(t_i') = w_i$ for some t_i'. Thus,

$$t \Rightarrow_\pi t' = \sigma(t_0, \cdots, t_i', \cdots, t_{n-1})$$

so that $w = \bar{e}(t')$, which concludes the proof.

We shall not repeat the definition here, but an OI (outside-in) *macro-grammar* Fischer [9] is exactly a one-state creative dendrogrammar which produces only the yield of the terminal tree. OI grammars produce exactly the *indexed languages* of Aho [1]. Thus the yield of a creative dendrolanguage is always an indexed language, and conversely (modulo the empty string).

In the spirit of Brainerd [6], one can define context-free tree grammars in a natural way. Let $\Sigma = N \cup T$, $N \cap T = \varnothing$, be a ranked alphabet. Consider productions of the form $\sigma(x_0, \cdots, x_{n-1}) \rightarrow u$, where $\sigma \in N$, and $u \in \mathscr{T}_\Sigma(X_n)$. Suppose $s = \sigma(s_0, \cdots, s_{n-1})$ is a subtree of a tree $t \in \mathscr{T}_\Sigma^0$.

Let $s' = S(\substack{s_i \\ x_i}|u)$. Replace the subtree s by s'. The resulting tree t' is defined to be the tree obtained from t by the given production.

Definition. Let $G = (\Sigma, S_0, \Pi)$, where Σ is as above, S_0 is a finite subset of $\mathscr{T}_\Sigma^0$, and Π is a finite set of productions. G is a *context-free dendrogrammar.*

Definition. The dendrolanguage generated by G is

$$\mathscr{L}(G) = \{w \in \mathscr{T}_T | (\exists s_0 \in S_0)\,(s_0 \Rightarrow^* w).$$

A derivation in a CF dendrogrammar is said to be *top-down* if whenever a symbol σ is rewritten using a production, σ is not a descendant of any node in N. This is the analogue of a left-to-right derivation in an ordinary context-free grammar. It is not hard to show (in fact it follows from work of Fischer [9]) that if G is any CF dendogrammar (CFDG), then any tree in $\mathscr{L}(G)$ may be obtained by a top-down derivation. Since the one-state creative dendrogrammars also work from the top down, it is clear that the context-free dendrolanguages are exactly the creative dendrolanguages. Taking yields, we have the equation

$$\frac{\text{recognizable dendrolanguages}}{\text{context-free languages}} = \frac{\text{context-free dendrolanguages}}{\text{indexed languages}}.$$

One may also use creative productions to define transformations on trees, thus obtaining creative surface sets and target languages. Decision problems for these sets remain solvable; in particular, recursive target languages are still obtained. Creative transformations, however, do not seem to reflect properties of transformations proposed for natural languages, so we have not studied them here.

REFERENCES

[1] A. V. AHO, Indexed grammars—an extension of the context-free grammars, *J. Assoc. Comput. Mach.* **15** (1968), 647–671.

[2] A. V. AHO and J. D. ULLMAN, Automaton analogs of syntax-directed translation schemata, *Proc. 9th IEEE Ann. Symp. on Switching and Automata Theory*, October, 1968.

[3] A. V. AHO and J. D. ULLMAN, Translations on a context-free grammar, *Proc. Assoc. Comput. Mach. Symp. on Theory of Computing*, May, 1969.

[4] M. A. ARBIB and Y. GIVE'ON, Algebra automata I: Parallel programming as a prolegomena to the categorical approach, *Information and Control* **12** (1968), 331–345.

[5] W. S. BRAINERD, The minimalization of tree automata, *Information and Control* **13** (1968), 484–491.

[6] W. S. BRAINERD, Tree generating regular systems, *Information and Control* **14** (1969), 217–231.

[7] N. CHOMSKY, *Aspects of the Theory of Syntax*, M.I.T. Press, Cambridge, Mass., 1965.
[8] J. E. DONER, Tree acceptors and some of their applications, System Development Corp., Scientific Report No. 8 (1967).
[9] M. J. FISCHER, Grammars with macro-like productions, *Proc. 9th IEEE Symp. on Switching and Automata Theory*, October, 1968.
[10] E. T. IRONS, A syntax-directed compiler for ALGOL 60, *Comm. ACM* **4** (1961), 51–55.
[11] P. M. LEWIS and R. E. STEARNS, Syntax-directed transduction, *J. Assoc. Comput. Mach.* **15** (1968), 465–488.
[12] J. MEZEI and J. B. WRIGHT, Algebraic automata and context-free sets, *Information and Control* **11** (1967), 3-29.
[13] M. O. RABIN, Mathematical Theory of Automata, *Mathematical Aspects of Computer Science*, Proc. Symposia Appl. Math., XIX, 173–175, American Mathematical Society, Providence, R.I., 1967.
[14] W. C. ROUNDS, Trees, transducers, and transformations, Dissertation, Stanford University, August, 1968.
[15] J. W. THATCHER, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, *J. Comput. System Sci.* **1** (1967), 317–322.
[16] J. W. THATCHER, Transformations and translations from the point of view of generalized finite automata theory, *Proc. Assoc. Comput. Mach. Conference on Theory of Computing*, May, 1969.
[17] 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 4 December 1969*)