CURRENTS IN THE THEORY OF COMPUTING

Edited by

ALFRED V. AHO

Bell Telephone Laboratories
Murray Hill, N.J

Contributing Authors:

RONALD V. BOOK
ALLAN BORODIN
ZOHAR MANNA
JAMES W. THATCHER
JEFFREY D. ULLMAN

PRENTICE-HALL, INC.

ENGLEWOOD CLIFFS, N.J.

4 TREE AUTOMATA: AN INFORMAL SURVEY

James W. Thatcher
IBM Thomas J. Watson Research Center
Yorktown Heights, New York

4-1. INTRODUCTION

We present an informal survey of an area of automata and language theory that has come to be known as "tree automata theory." The aim of this informal presentation is to convince the reader that tree automata theory is just conventional automata theory revisited or, more precisely, conventional automata theory in which labeled trees replace strings as inputs and outputs. Starting with the concept of finite-state acceptor, we show with examples how simple introductory results, generalized to trees, yield proofs of familiar theorems in context-free language theory. Put another way, we will be showing how some results in context-free language theory are proved using techniques of finite automata theory.

More important, however, is the connection between "tree automata with output" and concepts in language theory such as semantics, translations, and transformations. It is more difficult to be convincing on this point. The tree automata with output are conceptually more complicated than the acceptors (much more so than in the conventional theory), and both the definitions and the applications depend heavily on formalism, which, in this survey, we are trying to avoid.

The decision to avoid formalism is a painful one, for I believe that it is precisely that formalism [in terms of algebraic theories (Eilenberg-Wright 1967; Lawvere, 1963), for example] that offers the hope of significant application to language theory—even to computer science. The algebraic formulation is important for at least four reasons: (1) the definitions can be made

precise, yet simple and general; (2) the proofs about the concepts are direct (i.e., they are algebraic rather than combinatorial); (3) the "naturalness" of definitions can be tested as they are formulated within the algebraic framework; and (4) the algebraic framework suggests generalizations of the concepts so derived. On the other hand, our informal approach better enables us to indicate areas in which the theory of tree automata may find application (i.e., where the algebraic formulation may be of importance) in terms closer to those of the conventional theory.

Although we will not go into any details here, there is a third area of important application of tree automata and that is in the applications to decision problems in logic. There has been a series of papers applying (generalized) finite automata theory to decision problems in higher-order, applied logics starting with Büchi and Elgot's work (Büchi, 1960; Büchi and Elgot 1958; Elgot, 1961) on the weak second-order theory of arithmetic and culminating in the important decision procedure for the monadic second-order theory of multiple successors developed by Rabin (1968, 1970). Intermediate and auxiliary applications are to be found in Büchi (1962). Doner (1965a, 1965b), Landweber (1968), and Thatcher and Wright (1968).

It could be said that "tree automata theory" had its roots in the algebraic approach to the conventional theory taken by Büchi and Wright (1960). The generalizations were a natural outcome of that work and were arrived at independently (Doner, 1965; Thatcher and Wright, 1968). Other general papers in the area include Arbib (1968), Brainerd (1967, 1969), Magidor and Moran (1969), Mezei and Wright (1967), Rounds (1970), and Thatcher (1967, 1969, 1970). Rabin's survey paper (1967) contains a section on tree automata. Arbib and Give'on (1968) generalize the idea of tree automata to automata operating on directed ordered acyclic graphs (DOAGs). Brainerd (1967, 1969), in addition to considering minimal tree automata, generalizes Büchi's (1962) result concerning regular canonical systems.

4-2. TREES

Let Σ be an alphabet. A Σ tree is a tree whose nodes are labeled with elements of the alphabet Σ. Of course, a tree is a rooted, directed graph which is acyclic and ordered in the sense that the successors of any node have a specific order as is necessary when one draws a tree on paper. Figure 1 shows an $\{a, b\}$ tree, which is typical except that it is binary (every node has either two or zero successors). The root of the tree is the top node (in Figure 1) and the phrase "frontier of the tree" will be used ambiguously to mean either the set of nodes at the bottom of the tree (the end opposite the root) or the string of symbols labeling the frontier nodes. For a tree t, fr(t) will have the latter meaning. The frontier string of the tree in Figure 1 is *abaa*.

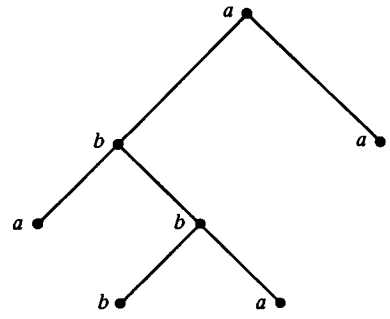

Fig. 1 Typical binary tree over $\{a, b\}$

There is a well-known correspondence between parenthesized expressions and trees. The tree in Figure 1 can be identified with the expression $a(b(a, b(b, a)), a)$, that in Figure 12 with $c(d(x_2, x_1), c(c, x_2))$.

4-3. TREE AUTOMATA AS RECOGNIZERS

We first consider finite automata as acceptors and later introduce the idea of output so the automata will be viewed as transducers. Of course, as in the Rabin and Scott (1959) theory, the acceptor approach can be considered a special case of the transducer approach, but the latter is considerably more complicated. Simplicity dictates attacking the ideas in two steps.

Let us consider the following informal description of a finite automaton as an acceptor. A *finite automaton* $\mathfrak{A}$ consists of a finite set of *states*, S; an *initial state*, s_0; a *transition function* M which maps pairs ⟨next-symbol, current-state⟩ into next-state; and a designated set F of *final states*.

The automaton $\mathfrak{A}$ operates on an input as follows. The automaton starts in its initial state s_0. This state is the current-state and the first symbol of the input is the next-symbol. Then generally, given the current-state and the next-symbol, M determines the next-state of $\mathfrak{A}$. This process is continued, producing a state "history" which is a string of states one longer than the input. The last-state of the state history is examined to determine if it is a final state; the input is *accepted* if this is the case. The set of inputs accepted by $\mathfrak{A}$ is denoted $T(\mathfrak{A})$, and a set U of inputs is *recognizable* if $U = T(\mathfrak{A})$ for some automaton $\mathfrak{A}$.

The terminology of our informal description may seem strange (current-state, next-symbol, one longer, etc.), but barring that, the reader will probably agree that the description captures the conventional idea of finite-state acceptor. The reason for the strange language is that certain words must be interpreted in the generalization to trees, but the resulting picture serves well to motivate the definition to follow.

We will operate under the simplifying restriction that all trees are binary and write T_Σ for the set of binary Σ trees. This restriction is purely for expo-

sitional purposes; the theory applies to trees with arbitrary (nonuniform) branching. In the examples to follow, the input alphabet is $\{a, b\}$.

Under these restrictions the tree illustrated in Figure 1 would be a typical input to an automaton. The first symbol of the input appears to be the symbol "a" at the root of the tree. Considering it to be such gives rise to the *root-to-frontier automaton* (RFA) which was first explicitly discussed in Magidor and Moran (1969). In accord with our informal description of the operation of an automaton, the transition function of an RFA must be defined, at least initially, on $\Sigma \times S$ since the pair $\langle$next-symbol, current-state$\rangle$ is from the set. Similarly, in order to proceed, the "next-state" must be a pair of states, one for each successor of the root. Therefore, we are motivated to require that the transition function be defined on $\Sigma \times S$, taking values in $S \times S$. The beginning of the calculation of the state history is shown in Figure 2. The calculation is continued in Figure 3 and completed in Figure 4. Note that in the process, the state history is forced (for the first time in Figure 3) to be one longer than the input in a generalized sense.

Having indicated how the state history is computed, we are left to interpret "last-state." One natural interpretation is to consider the set of all states

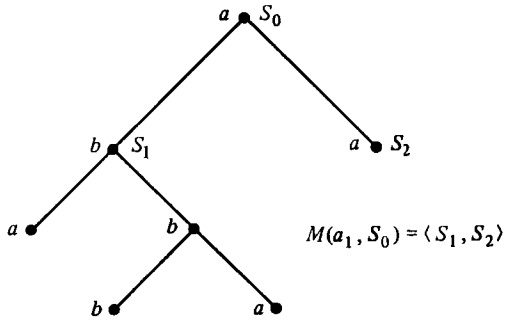

Fig. 2 Beginning the state calculation for RFA.

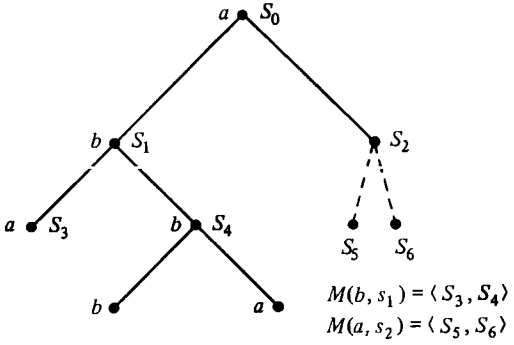

Fig. 3 State calculation (continued).

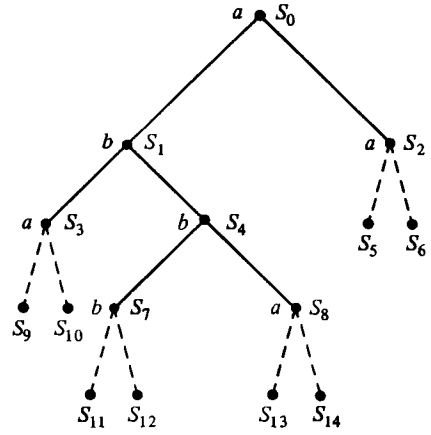

Fig. 4 State calculation (completed).

on the frontier of the state history to comprise the "last-state." Thus the behavioral condition for the input in Figure 1 would be $\{s_9, \ldots, s_{14}, s_5, s_6\} \subseteq F$.

Taking the informal concept of finite automaton, we are led to the following definition of a generalized finite automaton operating on binary trees.

DEFINITION 1

A (deterministic) *root-to-frontier automaton* $\mathfrak{A}$ (RFA) consists of a finite set S of states; a transition function, $M: \Sigma \times S \longrightarrow S \times S$; an initial state $s_0 \in S$; and a set of final states $F \subseteq S$.

The formation of the state tree is described inductively as follows:

a. The root of the state tree is labeled s_0.

b. Given that any node of the state tree is labeled s, and the corresponding node of the input is labeled σ, then the two successor nodes of the state tree are labeled with the pair $M(\sigma, s)$.

The input is accepted if every state labeling the frontier of the state tree is a final state.

The deterministic RFA are not very powerful. To illustrate this, there are finite sets which are not recognized by any deterministic RFA. For example, the doubleton $\{a(ba), a(ab)\}$ is not recognizable. [In Magidor and Moran (1969) $\{a(aa), a(bb), a(ab)\}$ is given as an example of a nonrecognizable finite set.]

A class of sets of trees recognized by deterministic RFA is obtained as follows. The notion of path in a tree is familiar; a complete linear sequence of nodes from the root to the frontier. We will say that a word $w \in \Sigma^*$ is a Σ path of a Σ tree t if w is the sequence of labels of a path of t. Thus the $\{a, b\}$

paths of the tree in Figure 1 are *aba*, *abbb*, *abba*, and *aa*. Now let R be any regular subset of Σ^* and define T_R to be the set of all trees t which have the property that every Σ path of t is in R. For any regular R the set T_R of Σ trees so defined is recognizable by a deterministic RFA. If $\langle S, M, s_0, F\rangle$ is a conventional finite automaton (Rabin and Scott, 1959) recognizing R, then $\langle S, M', s_0, F\rangle$ is a deterministic RFA recognizing T_R, where $M'(\sigma, s) = \langle M(\sigma, s), M(\sigma, s)\rangle$. Intuitively the RFA mimicks the action of the conventional automaton down every path of the input tree; the behavioral condition (all the frontier states of the state tree have to be final) requires that every Σ path of the input tree would be accepted by the original automaton.

For a specific example, if R is the subset of $\{a, b\}^*$ denoted by the regular expression, a^*b^*, then T_R is the set of trees in which all a-labeled nodes precede (in the sense of being closer to the root) all b-labeled nodes.

Looking more closely at the concept of deterministic RFA, one might have the feeling that the class of sets $\{T_R\}$ defined above just might exhaust the power of the deterministic RFA. This feeling is nearly correct as was proved in Magidor and Moran (1969) and outlined below.

Every path in a Σ tree can be represented by a string in $\{l, r\}^*$, where l and r correspond to the left and right branches from a node, respectively. Thus in Figure 1, the paths are: Λll, Λlrl, Λlrr, Λr. For convenience we have added a symbol (not the empty string) for the root of the tree so that the string representing a path will have the same length as the string of labels. A path together with its labels is now represented by a pair $\langle p, x\rangle$, where p is a path (i.e., a string in $\{\Lambda\}\cdot\{l, r\}^*$) and x is a string of labels and p and x have the same length. For example $\langle \Lambda ll, aba\rangle$ is the pair for the leftmost path in Figure 1. Such labeled path pairs can be interpreted in turn as elements of the set $\Delta_\Sigma = (\{\Lambda\} \times \Sigma)\cdot(\{l, r\} \times \Sigma)^*$ under the correspondence

$$\langle \Lambda, \sigma_1\rangle\langle \beta_2, \sigma_2\rangle \cdots \langle \beta_k, \sigma_k\rangle \rightleftharpoons \langle \Lambda\beta_2 \cdots \beta_k, \sigma_1 \cdots \sigma_k\rangle.$$

For any regular set R contained in Δ_Σ, let A_R be the set of Σ trees with the property that all label path pairs are in R. Now the characterization theorem of Magidor and Moran (1969) is:

THEOREM 1

A set U is recognizable by a deterministic RFA if and only if $U = A_R$ for some regular subset R of Δ_Σ

The decision that the label on the root was the first-symbol led to the notion of RFA. If instead we look at the symbols labeling the frontier of the input as "first-symbol," then we arrive at the concept of *frontier-to-root automaton* (FRA) which was first studied in Doner (1965b) and Thatcher and Wright (1968). It is natural, since the state history is to be "one longer" than the input, to begin the automaton's operation as shown in Figure 5.

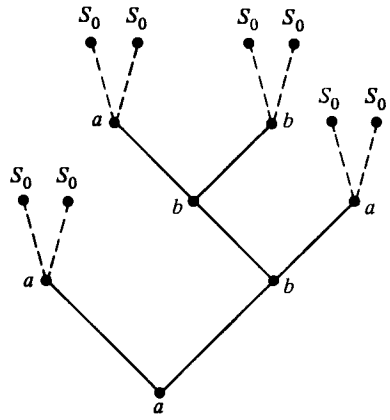

Fig. 5 Starting the FRA.

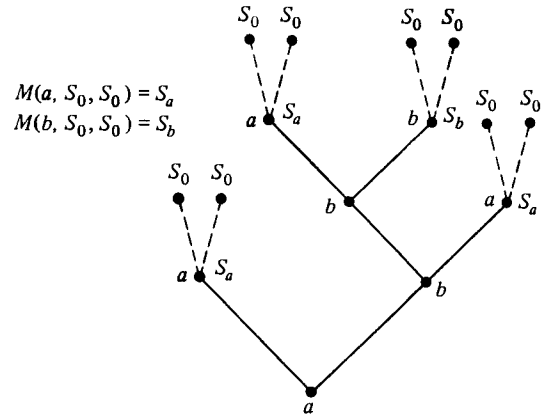

Fig. 6 First step of state calculation

For each first symbol on the frontier, the current-state is in fact the pair of states labeling the successors of the corresponding node of the state tree. Thus we want the transition function to be defined on $\Sigma \times S \times S$ taking values in S. The first step of the state history calculation is shown in Figure 6 and the completion in Figure 7. The choice of the behavioral condition is obvious. The input is accepted if the state labeling the root of the state tree is final.

The informal definition has thus led us to another definition of generalized finite automaton.

DEFINITION 2

A (deterministic) *frontier-to-root automaton* (FRA) $\mathfrak{A}$ consists of S, s_0, M, and F, where the components are as in Definition 1 with the exception that $M: \Sigma \times S \times S \to S$. The calculation of the state tree is described inductively by:

a. The frontier of the state tree is labeled s_0.

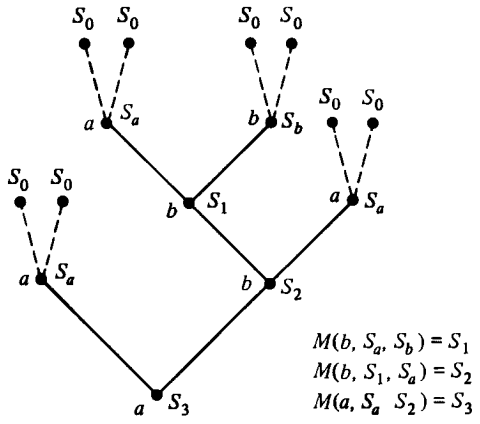

Fig. 7 Completing state calculation.

b. For any node in the input labeled σ, the corresponding node of the state tree is labeled $M(\sigma, s_1, s_2)$, where $\langle s_1, s_2 \rangle$ is the pair of states labeling the successors of that state tree node.

We will write $\bar{M}(t)$ to denote the state labeling the root of the state tree and t is accepted if $\bar{M}(t)$ is a final state.

The frontier-to-root deterministic automata are more interesting than their opposites described above. As the results below show, they mirror very closely conventional finite automata.

We will again use the family $\{T_R\}$ as a source of examples. Let R be a regular subset of Σ^* and let $\hat{R}$ be the set of reversals of strings in R. We know (Rabin-Scott, 1959) that $\hat{R}$ is regular if R is. Now let $\langle S, M, s_0, F \rangle$ be a conventional finite automaton which recognizes the set $\hat{R}$ in Σ^*. We can construct a deterministic FRA, $\langle pS, M', \{s_0\}, pF \rangle$ where $M'(\sigma, u, v) = \{M(\sigma, s) | s \in u \cup v\}$ and this FRA recognized T_R. (pS is the set of subsets of S.) The way it works is that at any node of the state tree the FRA has gathered up all states that the original finite automaton would have produced for all paths from the corresponding node of the input to the frontier. When talking about the root of the input, if all those paths are in R (i.e., all those states are final), then the input is in T_R and is accepted by the FRA.

This example makes two points quite clearly. First, one can see that closure of the FRA-recognizable sets under complementation works just as it does for conventional finite automata. The input is not in T_R just in case at least one path is not in R (i.e., just in case the set of states obtained at the root is in pS-pF). Second, the principle of the subset construction that is used to prove the equivalence of the nondeterministic and deterministic automata is implicit in the construction.

Nondeterminism is introduced into these models in the usual way, by making the range of the transition function the power set (set of subsets) of the range in the deterministic case and by allowing a set of initial states. Thus, giving only one of the definitions:

DEFINITION 2′

A *nondeterministic FRA* consists of S, S_0, M, and F, where the components are as in Definition 2 except that $S_0 \subseteq S$ is a set of initial states and $M: \Sigma \times S \times S \rightarrow pS$.

State trees in the nondeterministic case are calculated by selecting one of the elements in the value of the transition function as the next-state. Thus several state trees are possible for one input. An input is accepted if there exists a state tree for the input with the root labeled with a final state.

The nondeterministic FRA and nondeterministic RFA are equipotent because for any nondeterministic RFA $\mathfrak{A}$, it is easy to construct the transition function of a nondeterministic FRA $\mathfrak{A}'$ ($M'(\sigma, s_1, s_2) = \{s | \langle s_1, s_2 \rangle \in M(\sigma, s)\}$) such that upon switching initial and final states ($S_0' = F$, $F' = S_0$) the two automata recognize the same sets of inputs. And similarly, given a nondeterministic FRA $\mathfrak{A}$, then a nondeterministic RFA $\mathfrak{A}'$ has transition function defined by $M'(\sigma, s) = \{\langle s_1, s_2 \rangle | s \in M(\sigma, s_1, s_2)\}$ and also has initial and final sets of states interchanged. [Note that this construction corresponds to the proof that the regular sets are closed under reversal; see Rabin and Scott (1959).]

LEMMA 1

A set U of Σ trees is recognizable by a nondeterministic FRA iff U is recognizable by a nondeterministic RFA.

As suggested above, the subset construction familiar in the conventional theory works for the frontier-to-root automata. That is, given a nondeterministic FRA $\mathfrak{A}$ one can construct a deterministic FRA $\mathfrak{A}'$ with pS as the set of states such that $T(\mathfrak{A}) = T(\mathfrak{A}')$. In particular, if $\mathfrak{A} = \langle S, M, S_0, F \rangle$, then $\mathfrak{A}' = \langle pS, M', S_0, \{u | u \cap F \neq \varnothing\} \rangle$, where $M'(\sigma, u_1, u_2) = \bigcup_{s_i \in u_i} M(\sigma, s_1, s_2)$.

LEMMA 2

A set U of Σ trees is recognizable by a nondeterministic FRA iff U is recognizable by a deterministic FRA.

Boolean closure of the FRA recognizable sets is also obtained in a manner identical to that in (Rabin and Scott, 1959). Thus $T_\Sigma - T(\langle S, M, s_0, F \rangle) = T(\langle S, M, s_0, S - F \rangle)$ and the direct product construction yields an FRA which recognizes the intersection of the sets recognized by the component

automata. If $\mathfrak{A} = \langle S, M, s_0, F\rangle$ and $\mathfrak{B} = \langle T, N, t_0, G\rangle$, then

$$\mathfrak{A} \times \mathfrak{B} = \langle S \times T, M \times N, \langle s_0, t_0\rangle, F \times G\rangle$$

where $M \times N(\sigma, \langle s, t\rangle) = \langle M(\sigma, s), N(\sigma, t)\rangle$; $T(\mathfrak{A}) \cap T(\mathfrak{B}) = T(\mathfrak{A} \times \mathfrak{B})$.

THEOREM 2

The FRA-recognizable subsets of T_Σ form a Boolean algebra.

Let $hg(t)$ denote the height of the tree t; i.e., $hg(t)$ is the length of the longest path in t. Also, define $T_\Sigma^{(n)}$ to be the set of trees of height less than or equal to n. For any set $U \subseteq T_\Sigma^{(n)}$ define the FRA automaton, $\mathfrak{A}_U = \langle T_\Sigma^{(n)} \cup \{*, \Lambda\}, M, \Lambda, U\rangle$, where

$$M(\sigma, t_1, t_2) = \begin{cases} \sigma(t_1, t_2) & \text{if } t_1, t_2 \in T_\Sigma^{(n-1)} \\ \sigma & \text{if } t_1 = t_2 = \Lambda \\ * & \text{otherwise} \end{cases}$$

The automaton $\mathfrak{A}_U$ recognizes U. For this example we are depending on the identification (mentioned above) between Σ trees and parenthesized expressions (i.e., Σ terms). What is happening with $\mathfrak{A}_U$ is that for any input t with $hg(t) \leq n$, $\bar{M}(t) = t$ and otherwise $\bar{M}(t) = *$. Therefore, an input t is accepted iff $\bar{M}(t) = t \in U$.

Since any finite set is a subset of $T_\Sigma^{(n)}$ for some n, it follows that all finite subsets of T_Σ are FRA-recognizable:

LEMMA 3

All finite subsets of T_Σ are FRA-recognizable.

Now Lemmas 1 and 2 show that every RFA-recognizable set is FRA-recognizable but the converse is not true, as is seen from Lemma 3 together with the example of a finite set (see above) that is not RFA-recognizable.

In the sequel we will use the term "recognizable" without modification to mean FRA-recognizable.

Another important result in the Rabin–Scott theory of finite automata is what might be called the "uv^nw lemma," Lemma 8 (Rabin and Scott, 1959). To state the generalization we do need to introduce some notation.

Consider an FRA with state set S. The set $T_{\Sigma,S} \subseteq T_{\Sigma \cup S}$ is like the set of Σ trees except that in addition, elements of S are allowed to label the frontier nodes of trees in $T_{\Sigma,S}$. The operation of the FRA is extended to $T_{\Sigma,S}$ by requiring that $M(s, s_1, s_2) = s$ for $s, s_1, s_2 \in S$. Thus on the frontier nodes of a tree with labels from S, the corresponding labels of the state tree are identical. (This is closely related to the idea of startable automata [Arbib and Give'on, 1968].)

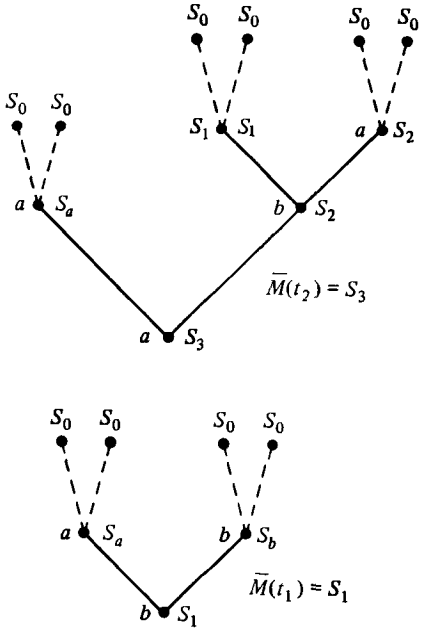

Fig. 8 Substitution of trees.

It should be very clear that if $M(t_1) = s$ and t results from substituting t_1 for any number of occurrences of s in a tree t_2, then $\bar{M}(t) = M(t_2)$. This is illustrated in Figure 8, where t is the tree in Figure 1.

For convenience, consider trees in $T_{\Sigma,S}$ which have at most one occurrence of an element of S on the frontier. Under this circumstance we can write t_1t_2 to mean the result of substituting t_2 for the occurrence of the element of S in t_1.

A generalization of Rabin and Scott's Lemma 8 can now be stated as follows (there is actually a much more general statement of this lemma):

LEMMA 4

If t is accepted by an automaton with r states and $hg(t) > r$, then there exist trees t_1, t_2, r_3 such that

a. $\text{fr}(t_i) = u_i s v_i$; $u_i, v_i \in \Sigma^*$, $s \in S$, $i = 1, 2$; $hg(t_2) > 1$.
b. $\text{fr}(t_3) = w \in \Sigma^*$.
c. $t = t_1t_2t_3$.
d. $t_1t_1^nt_3$ is accepted for all n.

This lemma, as is the case in the conventional theory, leads to the decidability of the emptiness and finiteness problems.

THEOREM 3

There exists an effective procedure for determining whether or not $T(\mathfrak{A})$ is empty and there is an effective procedure to determine whether $T(\mathfrak{A})$ is finite.

The notion of substitution necessary for Lemma 4 also gives rise to a theory of regularity (similar to the Kleene theory for recognizable sets of strings) which was developed in Thatcher and Wright (1968) and Magidor and Moran (1969) and simplified in Arbib and Give'on (1968). For sets U and V in T_Σ and for any $\sigma \in \Sigma$, the product $U \cdot_\sigma V$ is the set of all trees t that are obtained by taking $t_1 \in U$ and substituting elements of V for all occurrences of σ on the frontier of t_1. The iterated substitution, $U^{*\sigma}$ is defined in a manner similar to the definition of the Kleene star: $U^{*\sigma} = \bigcup X_i$, where $X_0 = \{\sigma\}$ and $X_{n+1} = X_n \cup U \cdot_\sigma X_n$. The Σ-*regular* sets are defined to be the least class of subsets of T_Σ containing the finite sets and closed under the operations $\cdot_\sigma$ and $^{*\sigma}$ for all $\sigma \in \Sigma$ and under set union. A set of trees is *regular* if it is Σ-regular for some Σ.

Recall that $T_{a^*b^*}$ is the set of $\{a, b\}$ trees with the property that all paths have labels in the set a^*b^*. A "regular expression" for this set of trees is $\{a(aa)\}^{*a} \cdot_\sigma (\{a\} \cup \{b(bb)\}^{*b})$.

The simplified proof in (Arbib and Give'on, 1968) of the analysis theorem (every recognizable set if regular) provides a good example of the uses of substitution and iterated substitution. We do not intend to give the whole proof but just the construction to illustrate the uses of these operations. Let $\mathfrak{A} = \langle S, M, s_0, F\rangle$ be an FRA and define $T[s, S_1, S_2]$ to be the set of trees $t \in T_{\Sigma, S_2}$ for which $\bar{M}(t) = s$ and all states in the state tree for t outside the frontier and root are in S_1. In effect, $T[s, S_1, S_2]$ can be read as the set of trees carrying the automaton $\mathfrak{A}$ from S_2 through S_1 to s. The proof of the analysis theorem goes through by induction on the cardinality of S_1 showing that all $T[s, S_1, S_2]$ are regular and since $T(\mathfrak{A}) = \bigcup_{s \in F} T[s, S, \varnothing]$, it follows that $T(\mathfrak{A})$ is regular. For the case $S_1 = \varnothing$ we have

$$T[s, \varnothing, S_2] = \{\delta(\beta_1\beta_2) \mid \beta_i \in \Sigma \cup S_2 \text{ and } \bar{M}(\sigma(\beta_1\beta_2)) = s\}$$

which is finite and hence regular. For the induction step,

$$T[s, S_1 \cup \{s'\}, S_2] = T[s, S_1, S_2 \cup \{s'\}] \cdot_{s'} T[s', S_1, S_2 \cup \{s'\}]^{*s'} \cdot_{s'} T[s', S_1, S_2]$$

This last equation can be visualized by imagining an input and corresponding state tree which has s at the root, internal nodes labeled from $S_2 \cup \{s'\}$ and possibly frontier nodes labeled from S_1. Such a state tree (and the input tree in the same way) can be broken up at every occurrence of s' resulting in an initial segment (including the root) in $T[s, S_1, S_2 \cup \{s'\}]$ perhaps several segments going from $S_2 \cup \{s'\}$ to s' through S_1 and several final segments going from S_2 to s' through S_1.

The main theorem relating the regular and the recognizable sets is:

THEOREM 4

If $U \subseteq T_\Sigma$ is recognizable, then it is Σ'-regular for some $\Sigma' \supseteq \Sigma$. If U is Σ-regular, then U is recognizable.

As a final example of a theorem familiar in the conventional theory, we will discuss the closure of the recognizable sets under projections or, as these operations are sometimes called, length-preserving homomorphisms or relabelings. We will return to this subject when we discuss automata with output, but for the time being, let μ and μ_0 be two maps defined on an alphabet Σ taking values in another alphabet Ω. Then $\bar{\mu}$ is a map from T_Σ into T_Ω; $\mu(t)$ is the result of relabeling every frontier node of t labeled σ with $\mu_0(\sigma)$ and every interior node labeled σ with $\mu(\sigma)$. Maps $\bar{\mu}$ defined in this way will be called *projections*. The following theorem is proved exactly as it is in the case of finite automata over strings.

THEOREM 5

The recognizable sets are closed under projections and inverse projections.

4-4. CONTEXT-FREE SETS AND RECOGNIZABILITY

For purposes of exposition, we have restricted ourselves to binary trees. Thus, in this section, we are forced to consider only "binary context-free grammars." It must be emphasized, however, that the results discussed here apply to arbitrary nonerasing context-free grammars (see Thatcher, 1967).

DEFINITION 3

A *binary context-free grammar* (CFG for short) is a 4-tuple $\langle N, \Sigma, P, S_0\rangle$, where the components are as follows: N is the set of nonterminals; Σ is the set of terminals; P is the set of productions, $P \subseteq N \times (\Sigma \cup N)^2$; and $S_0 \subseteq N$ is the set of start symbols. For a CFG, G, $T(G)$ is the set of *derivation trees* of G defined by the property that $t \in T(G)$ iff (a) the root of t is labeled with an element of S_0, (b) for every node of t labeled with σ having successor nodes labeled σ_1, σ_2, the triple $\langle\sigma, \sigma_1, \sigma_2\rangle \in P$, and (c) every frontier node of t is labeled from Σ. $L(G)$ is the set of frontier words of $T(G)$, $L(G) = \text{fr}T(G)$, and a set of trees U is called *local* if there is a CFG G such that $U = T(G)$, while a set U of words is *context-free* if there is a CFG G with $U = L(G)$.

Besides the restriction to binary trees, our definition of context-free grammar differs from the usual in inessential respects. First, a set of initial symbols

is allowed and, second, there is no requirement that Σ and N be disjoint. This relaxing of the definition permits a uniformity in the theory with no loss of generality.

As is usual, we can write $\sigma \longrightarrow \sigma_1\sigma_2$ for the production $\langle \sigma, \sigma_1, \sigma_2 \rangle \in P$ and $\sigma \longrightarrow w_1$, $\sigma \longrightarrow w_2$ is abbreviated $\sigma \longrightarrow w_1 \mid w_2$. The following grammar is illustrative of the relaxed conditions on the definition. Its set of derivation trees is $T_{a^*b^*}$.

$$N = \{a, b\}$$
$$\Sigma = \{a, b\}$$
$$P: \; a \longrightarrow aa \mid ab \mid ba \mid bb$$
$$\quad b \longrightarrow bb$$
$$S_0 = \{a, b\}$$

For this grammar, $L(G)$ is the set of all $\{a, b\}$ strings of length at least 2.

It should be fairly clear that every local set is recognizable because given a CFG, an FRA can be constructed which checks each of the local conditions, i.e., (a), (b), and (c) of Definition 3. We have from Thatcher (1967):

THEOREM 6

For any CFG G, $T(G)$ is recognizable.

The converse of Theorem 6 is not true. As an example, the set of $\{a, b\}$ trees with exactly one occurrence of the label b is recognizable but not local. There is a result that is close to a converse, however. For an automaton $\mathfrak{A}$, define $CS(\mathfrak{A})$ (for *complete state* trees of $\mathfrak{A}$) as a set of trees over the alphabet $\Sigma \times S \cup \Sigma$. The trees in $CS(\mathfrak{A})$ are nearly those indicated be Figure 7. If t is an input tree, $CS(t)$ is the tree of the same shape as t with the same labels on the frontier. The interior nodes are labeled with pairs $\langle \sigma, s \rangle$, where σ is the label of the input and s is the label of the corresponding node of the state tree [$CS(t)$ would be a set of trees if $\mathfrak{A}$ were nondeterministic]. For the tree in Figures 1 and 7, $CS(t)$ is shown is Figure 9. $CS(\mathfrak{A})$ is $CS(T(\mathfrak{A}))$.

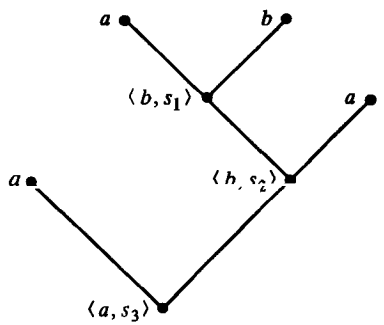

Fig. 9 Complete state tree.

THEOREM 7

If $\mathfrak{A}$ is an FRA, then $CS(\mathfrak{A}) = T(G)$ for some context-free grammar G.

Now since $\mathrm{fr}CS(\mathfrak{A}) = \mathrm{fr}T(\mathfrak{A})$ we obtain from this theorem:

THEOREM 8

If U is a recognizable set of trees, then $\mathrm{fr}(U)$ is context free.

Also, we can define the natural projection $\bar{\pi}$ from $T_{\Sigma \times S \cup \Sigma}$ onto T_Σ so that $\bar{\pi}CS(\mathfrak{A}) = T(\mathfrak{A})$. Therefore, we have a charcterization of the recognizable sets in terms of the local sets (see Thatcher, 1967).

THEOREM 9

Every recognizable set is the projection of a local set.

To illustrate this last theorem, it was mentioned above that the set B_1 of $\{a, b\}$ trees with exactly one label b is recognizable but not local. But the local set obtained as the set of derivation trees of the following grammar yields B_1 under projection.

$$N = \{a, b, c\}$$
$$\Sigma = \{a, b\}$$
$$P: \; a \longrightarrow aa$$
$$\quad b \longrightarrow aa$$
$$\quad c \longrightarrow ca \mid ac \mid ab \mid ba$$
$$S_0 = \{b, c\}$$

For this grammar $T(G)$ is the set of $\{a, b, c\}$ trees with exactly one path of c labels terminating in a b. The projection that is the identity on $\{a, b\}$ and maps c into a takes $T(G)$ into B_1.

We are not going to use these connections between generalized finite automata theory and context-free language theory to prove any new results about context-free sets. Instead, we will try to show with the help of several examples how old results follow quite easily from what we have so far (and what we have so far is essentially finite automata theory!). I believe that looking at the results (and proofs) from the point of view of finite automata generalized to trees yields a clearer understanding of those results.

Example 1

The Peters–Ritchie result concerning node admissibility conditions and context-sensitive grammars (Peters and Ritchie (1969))

A context-sensitive grammar G with rules of the form $A \longrightarrow w(u, v)$,†

†In the notation of Chapter 1 we would write $uAv \longrightarrow uwv$.

read A is rewritten as w in the context u-v is usually interpreted in the derivation sense resulting in the context-sensitive languages. Instead, Peters and Ritchie look at a context-sensitive grammar as describing a set $T(G)$ of trees Informally speaking, a tree t is in $T(G)$ if every node in t can be justified by one of the context-sensitive rules; i.e., for every node labeled A with successor nodes labeled by the string w, there must exist a "context" u-v at that node such that $A \rightarrow w(u, v)$ is a rule of G For example, the node labeled c in Figure 10 could be "justified" by any rule $c \rightarrow ab(u, v)$ where $u \in \{\lambda, b, a, aa\}$ and $v \in \{\lambda, d, b, ba\}$.† Then defining $L(G)$ to be fr$T(G)$, the theorem is:

THEOREM
(*Peters and Ritchie*)

Under the node-admissibility interpretation of context-sensitive grammars, $L(G)$ is context-free.

The proof of this theorem from the finite automata point of view is to show that $T(G)$ is recognizable, and this is fairly clear. Then apply Theorem 8. This proof is not essentially different than the originators' proof, but it does provide some additional insight and offers the possibility (not yet realized) of giving a quantitative analysis of the extent to which the use of context-sensitive node-admissibility rules "simplifies" the description of context-free languages.

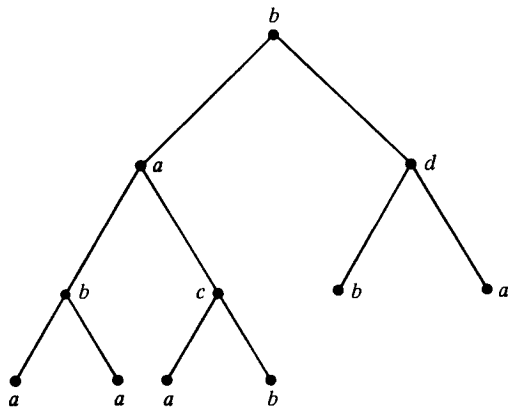

Fig. 10

Example 2

The structural equivalence results of Paull and Unger (1967) and McNaughton (1967)

†λ is the empty string.

For any alphabet Σ define the projection $\bar{\beta}$, where $\beta(\sigma) = *$ and $\beta_0(\sigma) = \sigma$; $\sigma \in \Sigma$ and $*$ is a special symbol which may or may not be in Σ. Thus $\bar{\beta}$ (which should be indexed by the alphabet) maps all the internal node labels into $*$ and leaves the frontier nodes as is. One can read $\bar{\beta}(t)$ as the *structure* of t.

DEFINITION 4

Generally two sets of trees U and V will be called *structurally equivalent* if $\bar{\beta}(U) = \bar{\beta}(V)$ and in particular, context-free grammars G_1 and G_2 are structurally equivalent if $\bar{\beta}(T(G_1)) = \bar{\beta}(T(G_2))$.

THEOREM
(*McNaughton, Paull, and Unger*)

There is a decision procedure for structural equivalence.

By Theorem 6 and Theorem 5, $\bar{\beta}(T(G))$ is recognizable. Theorems 2 and 3 show that there is a decision procedure for equality of recognizable sets.

Example 3

The decidability of the structural ambiguity problem (Paull and Unger, 1967)

A context-free grammar is structurally ambiguous if there exist two distinct derivation trees of the same shape with the same frontier; i.e., the grammar assigns to a word in the language two distinct derivations, which yield the same structure in the sense of Example 2. Put generally, we have:

DEFINITION 5

A set of trees U is *structurally ambiguous* if the projection $\bar{\beta}$ is not one-to-one.

In $T_{\Sigma \times \Sigma}$ define W to be the set of trees with properties: (1) there is at least one label $\langle \sigma, \sigma' \rangle$ on an internal node with $\sigma \neq \sigma'$; and (2) the frontiers are labeled with pairs $\langle \sigma, \sigma \rangle$. Clearly W is recognizable (in fact, local). Now let π_1 and π_2 be the natural projections from $T_{\Sigma \times \Sigma}$ onto T_Σ.

LEMMA 5

U is structurally ambiguous iff $\pi_1^{-1}(U) \cap \pi_2^{-1}(U) \cap W$ is nonempty.

Applying Theorems 5, 2, and 3, we have:

THEOREM
(*Paull and Unger*)

There exists a decision procedure for structural ambiguity.

Example 4

Gray's result on invertible grammars (Gray and Harrison, 1969)

A context-free grammar is *invertible* if the right-hand side of a production uniquely determines the nonterminal on the left-hand side.

THEOREM
(Gray)

For any context-free grammar G, there exists a structurally equivalent invertible grammar G'.

Note, of course, that structural equivalence implies weak equivalence; i.e., the languages are the same. Given G, $\bar{\beta}T(G)$ is recognizable by a deterministic FRA $\mathfrak{A}$ (Example 2). Now $CS(\mathfrak{A})$ is $T(G')$ for some G' (Theorem 7). All rules of G' are of the form $\langle *, M(*, s_1, s_2)\rangle \longrightarrow \delta_1\delta_2$ where either $\delta_i = \langle *, s_i\rangle$ and $s_i \in S$ (states of $\mathfrak{A}$) or $\delta_i \in \Sigma$ and $M(\delta_i, s_0, s_0) = s_i$. Thus G' is invertible and structurally equivalent to G.

Example 5

The "xu^kwv^ky" theorem (Bar-Hillel, Perles, and Shamir, 1961)

THEOREM
(Bar-Hillel, Perles, and Shamir).

For any context-free grammar, there exists an integer p such that if the length of a word $z \in L(G)$ is greater than p, then one can find x, u, w, v, and y such that $z = xuwvy$, and $xu^kwv^ky \in L(G)$ for all k.

One may have had the feeling that this theorem is a generalization of the corresponding theorem in finite automata theory. Indeed, looking at Lemma 4, let $\mathrm{fr}(t_1) = xsy$, $\mathrm{fr}(t_2) = usv$, and $\mathrm{fr}(t_3) = w$. Then if the automaton involved is the one accepting $T(G)$, we have $t_1t_2^nt_3 \in T(G)$ for all n so that $xu^nwv^ny \in L(G)$ for all n. The integer p from this proof is 2^r, where r is the number of states of the automaton recognizing $T(G)$.

It seems fairly clear that other results in context-free language theory which involve the periodicity of the derivation trees may be more easily handled using the finite automata approach.

4-5. TREE AUTOMATA WITH OUTPUT

The generalization of the concept of finite automata with output is as direct and simple as that of recognizer with one exception, the way in which successive outputs are "composed." In the conventional theory, the composition operation is to concatenate the outputs together in sequence. This composition on strings is so natural that it hardly merits mention; however, in the case of trees, we must state explicitly what we view as an appropriate "composition." That explicit statement is at once the complication in the generalization and, I believe, the power of the concept.

First we consider the case where the inputs as well as outputs are trees. The output function A for an automaton (either RFA or FRA) has the same domain of definition as the transition function. (Thus we are dealing with the "Mealy model.") The values of the output function are trees over a possibly different alphabet, say Ω, plus two (since we are considering only *binary* trees at present) special symbols, x_1 and x_2, which can occur on the frontiers of the output trees. As in the definition of $T_{\Sigma,S}$ in Section 4-4, we write T_{Ω,X_2} to denote the set of Ω trees which also allow x_1, x_2 to label frontier nodes.

The special symbols x_1, x_2 will specify the composition of outputs in a natural way, which can be illustrated as follows.

Consider an arbitrary subtree of an input tree as shown diagrammatically in Figure 11. If the finite automaton determines (via the functions A and M) the one-step output for the root node of that subtree to be that shown in Figure 12 (where the alphabet is $\{c, d\}$), then, again diagrammatically, the output for the whole subtree would be that shown in Figure 13, where t_1' and t_2' are the outputs for the whole subtrees t_1 and t_2, respectivley.

If we denote by t the tree in Figure 13 and by t' that in Figure 12, then we use the terminology "t results from simultaneously substituting t_i' for x_i in t'" to refer to the operation performed to obtain t from t'. This operation will be denoted $t = t' \leftarrow [t_1', t_2']$.

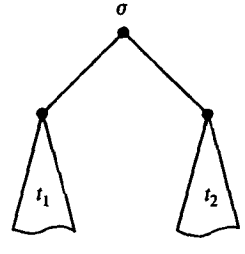

Fig. 11

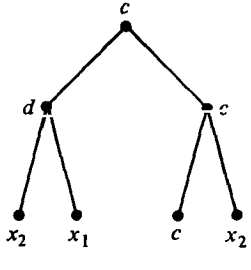

Fig. 12

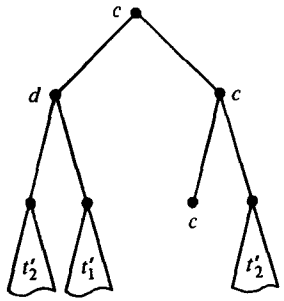

Fig. 13

DEFINITION 6

The tree $t \leftarrow [t_1, \ldots, t_n]$ is the result of simultaneously substituting t_i for x_i in t, $1 \leq i \leq n$.

We now give the definitions of tree automata with output (both FRA and RFA) and then some examples that exemplify how they operate. The emphasis will be on root-to-frontier automata because these have been studied (Rounds, 1970; Thatcher, 1970) and seem to be most applicable to questions in language theory. The definition of FRA with output is new, although it is a natural generalization deserving further study.

There is, in the following definitions, a complication that might not have been expected considering the informal discussion above. The automaton with output has two ouput functions, A and A_0. The former is the one discussed above; A_0 works, as in the definition of projection in Section 4-3, in the special case of processing frontier nodes.

DEFINITION 7

A *root-to-frontier automaton with output* (RFAO) consists of S, s_0, and M as in Definition 1 and two maps A and A_0 which have the same domain as M, A_0 taking values in T_Ω and A taking values in T_{Ω, X_2}. Σ is the *input alphabet* and Ω is the *output alphabet.*

The RFAO determines a map $\bar{A}: T_\Sigma \times S \longrightarrow T_\Omega$ by:

a. If t is a single node labeled σ, then $\bar{A}(t, s) = A_0(\sigma, s)$.
b. Generally, if t has the form of Figure 11, then
$\bar{A}(t, s) = A(\sigma, s) \leftarrow [\bar{A}(t_1, s_1), \bar{A}(t_2, s_2)]$, where $M(\sigma, s) = \langle s_1, s_2 \rangle$.

Finally, the map $\bar{A}: T_\Sigma \longrightarrow T_\Omega$ is given by $\bar{A}(t) = \bar{A}(t, s_0)$.

To aid in interpretation, the general step, b, can be read: The output produced from t with root state s is the result of substituting the output produced from t_i with root state s_i for x_i in $A(\sigma, s)$, where $\langle s_1, s_2 \rangle = M(\sigma, s)$.

DEFINITION 8

A *frontier-to-root automaton with output* (FRAO) consists of S, s_0, M as in Definition 2 and two functions, A and A_0, with the same domain as M The range of A_0 is T_Ω and the range of A is T_{Ω, X_1}. Σ is the *input alphabet* and Ω is the *output alphabet.*

The FRAO determines a map $\bar{A}: T_\Sigma \longrightarrow T_\Omega$ by:

a. If t is a single node labeled σ, then $\bar{A}(t) = A_0(\sigma, s_0, s_0)$.
b. Generally, if t has the form of Figure 11, then

$$\bar{A}(t) = A(\sigma, s_1, s_2) \leftarrow [\bar{A}(t_1), \bar{A}(t_2)], \text{ where } \bar{M}(t_i) = s_i,\ i = 1, 2.$$

We give two admittedly contrived examples (one RFAO and one RFAO) with the hope that these will clarify the relevant notions.

Let $t \in T_\Sigma$. An occurrence of a word w as a subword of fr(t) is called a σ phrase if that occurrence is fr(t') for some subtree t' of t with root labeled σ. Such an occurrence is called maximal if w is not properly contained in another σ phrase, For example, from Figure 1, *ba* is a b phrase, but it is not maximal because it is contained in the (maximal) b phrase *aba.*

The mapping $t \longrightarrow g_b(t)$, which reverses and puts parentheses around all maximal b phrases, is an RFAO mapping. The root-to-frontier automaton that performs this map is described in Table 1. Successive steps of the operation of this automaton on the input of Figure 1 are shown in Figures 14, 15,

$\Sigma = \{a, b\}$ $\Omega = \{a, b, (,)\}$ $S = \{s_0, s_1\}$

$M(a, s_0) = \langle s_0, s_0 \rangle$

$M(\sigma, s) = \langle s_1, s_1 \rangle$ otherwise

$A_0(\sigma, s) = \sigma$

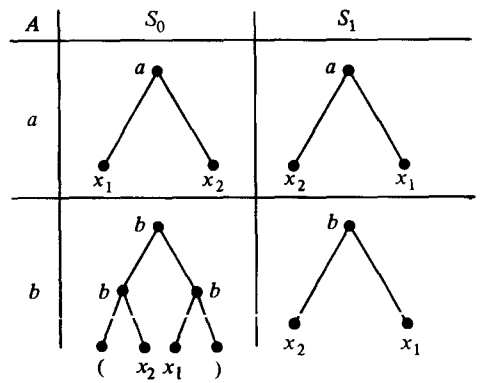

Table 1

and 16. The last figure actually comprises two steps in the process; the bottom two nodes should be labeled x_2, x_1 in that order and a final application of A_0 substitutes a for x_2 and b for x_1.

Let R be a regular set. Consider the mapping $t \longrightarrow h_R(t)$, which substitutes the single node labeled c for every maximal subtree t' of t for which $\mathrm{fr}(t') \in R$. This mapping can be performed by a frontier-to-root automaton with outputs

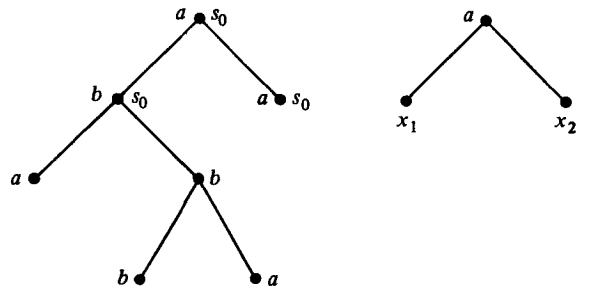

Fig. 14

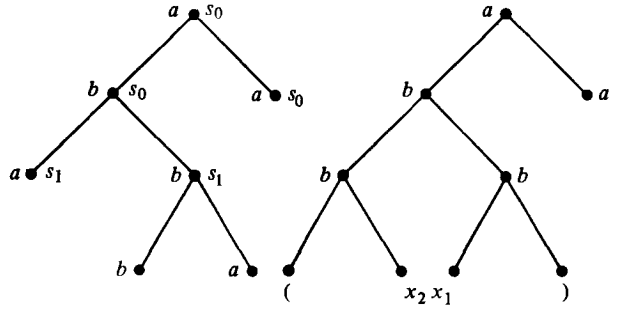

Fig. 15

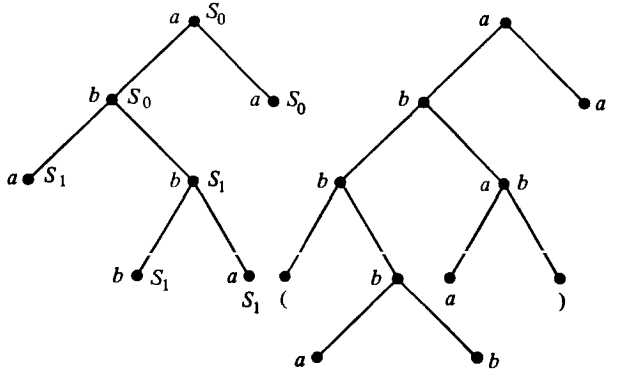

Fig. 16

as follows. Let S be a semigroup and μ a homomorphism from Σ^* into S such that $\mu^{-1}(S_F) = R$ (S_F is a designated subset of S). The states of the FRAC are $S \cup \{s_0\}$ ($s_0 \notin S$), the input alphabet is Σ and the output alphabet is $\Sigma \cup \{c\}$.

$$M(\sigma, s_0, s_0) = \mu(\sigma),\ M(\sigma, s_1, s_2) = s_1 s_2\ (s_1, s_2 \in S)$$

$$A_0(\sigma, s, s') = \begin{cases} c & \text{if } M(\sigma, s, s') \in S_F \\ \sigma & \text{otherwise.} \end{cases}$$

$$A(\sigma, s, s') = \begin{cases} c & \text{if } M(\sigma, s, s') \in S_F \\ \sigma(x_1, x_2) & \text{otherwise} \end{cases}$$

If $\Sigma = \{a, b, c,\}$ and $R = a^*b$, then the automaton constructed above would produce the output of Figure 17 with the input of Figure 10. We will not go through the detailed steps of the process.

There are two special cases of finite automata with output that are of particular importance. Under the situation when $|S| = 1$, the transition function is degenerate and the output functions depend only on the input symbols. It is easy to see that in this simplified case, Definitions 7 and 8 coincide. The maps A obtained in this way will be called *homomorphisms*.

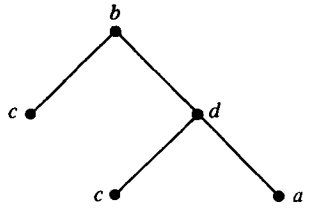

Fig. 17

DEFINITION 9

Maps $A_0: \Sigma \longrightarrow T_\Omega$ and $A: \Sigma \longrightarrow T_{\Omega,X}$, determine a *homomorphism* $\bar{A}$ by:

a. If t is a single node labeled σ, then $\bar{A}(t) = A_0(\sigma)$.
b. If t has the form of Figure 4-11, then $\bar{A}(t) = A(\sigma) \leftarrow [\bar{A}(t_1), \bar{A}(t_2)]$.

A simple example of a homomorphism is obtained by taking $A_0(\sigma) = \sigma$ and $A(\sigma) = \sigma(x_1, x_2)$. Then $\bar{A}(t)$ is the reversal or reflection of t. As a second example, let $A_0(\sigma) = a$ and $A(\sigma) = \sigma(x_1, x_2)$. $\bar{A}(t)$ is the balanced binary tree completely labeled with "a" of height the same as the length of the leftmost path of t. Every projection as defined in Section 4-3 is a homomorphism.

An automaton with output (and the map $\bar{A}$ associated with such an automaton) will be called *linear* if, in the values of the output functions, there are no repetitions of the special symbols, x_1 and x_2. Thus, in the two previous examples of homomorphisms, the first is linear and the second is not. The importance of linearity is given in the following theorem.

THEOREM 10

Maps from T_Σ to T_Ω determined by linear automata preserve recognizability. Conversely, if $\bar{A}$ is a map determined by a nonlinear automaton, there exists a recognizable set $U \subseteq T_\Sigma$ such that $\bar{A}(U)$ is not recognizable.

Having considered two specializations of finite tree automata with output, we now mention two generalizations. First, in the obvious way, nondeterminism can be introduced as in Definition 2′. That is, for a nondeterministic automaton with output, the three functions M, A, and A_0 all have sets as values and an initial set of states is allowed.

A far more important generalization is studied extensively in Thatcher (1970), where the transition and output functions are combined. Looking at the one-step output in Figure 12 (in which there are two occurrences of x_2), the root-to-frontier automaton would determine, via M, a pair of states $\langle s_1, s_2 \rangle$. Then the output of the automaton operating on t_2 with root state s_2 (which in Figure 13 is shown as t_2') is substituted for both occurrences of x_2. Mechanistically, one can view this process as if the automaton had duplicated at both nodes labeled x_2 and continued to process t_2 in state s_2. The generalization would allow distinct states, say s_2 and s_2', to be used as root states for the two distinct occurrences of x_2. The formulation of this generalization is in a sense simpler than the original. Instead of the special symbols $\{x_1, x_2\}$, we will allow special symbols from the set $\{x_1, x_2\} \times S$, where S is the set of states and an occurrence of the pair $\langle x_i, s \rangle$ on the frontier of an output will mean that the root-to-frontier automaton will substitute for the special symbol $\langle x_i, s \rangle$ the result of operating on t_i with root state s.

Since Definition 7 is really a special case of the generalization being discussed, we will use the same terminology.

DEFINITION 7′

A *root-to-frontier automaton with output* (RFAO) consists of S and s_0 as in Definition 7 and a pair of maps A from $\Sigma \times S$ into $T_{\Omega, X_2 \times S}$ and A_0 from $\Sigma \times S$ into T_Ω.

The RFAO determines a map $\bar{A}: T_\Sigma \times S \longrightarrow T_\Omega$ by:

a. If t is a single node labeled σ, then $\bar{A}(t, s) = A_0(\sigma, s)$.

b. Generally, if t has the form shown in Figure 11, then $\bar{A}(t, s)$ is the result of simultaneously substituting $\bar{A}(t_i, s')$ for all occurrences of $\langle x_i, s' \rangle$ in $A(\sigma, s)$.

Finally, the map $\bar{A}: T_\Sigma \longrightarrow T_\Omega$ is given by $\bar{A}(t) = \bar{A}(t, s_0)$.

Nondeterminism and linearity are extended to Definition 7′ just as stated above. Note that under the condition of linearity, Definitions 7 and 7′ coincide.

An example of an RFAO mapping which has appeared several times in the literature is the derivative of an arithmetic expression. This example requires the generalization of Definition 7′. To simplify matters, consider the map defined on the local subset of $T_{\{x, +, \cdot, c, 0, 1\}}$ consisting of those trees (arithmetic expressions) which have internal nodes labeled from $\{+, \cdot\}$ and frontier nodes labeled from $\{x, c, 0, 1\}$. The output trees are from the same set. The RFAO which produces the derivative of such an expression has two states, $\{D, I\}$, which correspond to "take the derivative" and "identity," respectively. The initial state is D and the combined transition-output functions are shown in Tables 2 and 3.

A_0	D	I
x	1	x
$\sigma \in \{c, 0, 1\}$	0	σ
$+, \cdot$	not applicable	

Table 2

A	D	I
$+$	$\left[+ \; [x_1, D] \; [x_2, D]\right]$	$\left[+ \; [x_1, I] \; [x_2, I]\right]$
$\{C, 0, 1, x\}$	$\left[+ \; [x_1, D] \; [x_2, I] \; [x_1, I] \; [x_2, D]\right]$	$\left[\bullet \; [x_1, I] \; [x_2, I]\right]$
	not applicable	not applicable

Table 3

4-6. TRANSLATIONS AND SEMANTICS: FINITE AUTOMATA REVISITED?

In order to begin to make comparisons between generalized finite automata with output and some of the formal treatments of translations and semantics, we must get rid of one technical headache at the outset. Although not so from the finite automata point of view, it seems to have been natural from the context-free language point of view, to have proposed definitions of translations and of semantics based on functions defined on the productions of a grammar.

A homomorphism (Definition 9) is a mapping from trees to trees; thus a homomorphism might be looked upon as a mapping from the derivation trees of one grammar to those of another. The basis function A of that definition would, when looked upon in this way, have the set $\Sigma \cup N$ as its domain. This is the natural approach from the finite automata point of view. But it is clear from the literature that this is not the natural approach from the context-free language theorists' point of view. That outlook would have led one to have replaced Definition 9 with one which A was defined (in the binary case to which we were restricting ourselves) on $\Sigma \times \Sigma^2$, in effect, on the productions of a grammar.

As indicated in Thatcher (1970), the definitions given here (both acceptors and automata with output) can be generalized to two-level or even n-level automata. In the n-level generalization, the next-state and the output for a given node of an input tree are determined by the current-state together with the complete input subtree of height n rooted at that node. Comparing this with the conventional theory of finite automata, the analogue would be a finite state device with n permanently adjacent reading heads on its input tape which are required to move in unison, one tape square at a time. Now this isn't a very interesting generalization in the conventional theory; in fact, no new input–output functions can be obtained. For tree automata, new input–output functions are obtained with increasing n, but there are mathematical reasons to believe that this kind of generalization is the wrong approach for theoretical study. The fundamental reason is that the algebraic formulation naturally leads to the "one-level" definitions given here and secondary reasons are typified by the observation that "two-level homomorphisms" are not closed under composition.

This technical headache (and the extent to which it is a headache will be clear from the propositions to follow) is overcome with an alternative approach provided by the following definition.

DEFINITION 10

For $t \in T_\Sigma$, the *production tree* of t, denoted $\mathcal{P}(t)$, is the unique $(\Sigma \cup \Sigma^3)$ tree which is structurally equivalent to t (thus having the same frontier as t) and which for every internal node has the label $\langle \sigma, \sigma_1, \sigma_2 \rangle$ iff the corresponding node of t has label σ and successors labeled $\langle \sigma_1, \sigma_2 \rangle$. For $U \subseteq T_\Sigma$ we will, as usual, write $\mathcal{P}(U)$ for the set of production trees of elements of U and in the special case of a context-free grammar G, we will write $T_\mathcal{P}(G)$ for $\mathcal{P}(T(G))$.

This definition has an auxiliary benefit in giving an example of an FRAO realizable mapping which cannot be obtained with a root-to-frontier automaton. Indeed, it should be a very simple matter for the reader to check that for any Σ, $\mathcal{P}$ (which, like the structure projection $\bar{\beta}$, should be indexed by Σ)

is the output map $\bar{A}$ of a linear FRAO with state set Σ. Thus, by Theorem 10, if U is recognizable, so is $\mathcal{P}(U)$. For the special case of context-free grammars, it is a familiar construction to go from a grammar G to a structurally equivalent grammar G' such that $T(G') = T_\mathcal{P}(G)$.

In the pioneering paper on the subject of "syntax-directed translation" (Irons, 1961), Irons emphasized that the essential feature of his syntax-directed compiler for ALGOL 60 was the fact that it separated the function of defining a language (via a context-free grammar, for example) from that of translating it into another. This idea of separating language definition, recognition and parsing (generally, language syntax) from the translation into another language (generally assigning semantics to the source language) has certainly gained wide acceptance. Various versions, formalizations, modifications, and generalizations of Irons's idea of syntax-directed mapping have been investigated in Aho and Ullman (1969a, 1969b, 1968), Culik (1967), Knuth (1968b), Lewis and Stearns (1968), Petrone (1965), Rounds (1970), Thatcher (1969, 1970), and Younger (1967). Although these papers (including my own) are replete with references to earlier and concurrent works, statements such as "similar models have been studied by [x, y, z]" are symptomatic of the disarray in the theoretical development relating to the second half of Irons's dichotomy. The approach based on finite tree automata (with the associated algebraic formulation), as outlined in the last section, provides a unifying framework for that development.

Of the list above, Petrone's "Syntax Mappings of Context-Free Languages" (Petrone, 1965) is the earliest, most general, and least frequently referenced treatment of Irons's ideas.

Because of the generality, the lack of algebraic formalism here (and in Petrone's paper) is especially limiting. It requires our discussion to be even more imprecise than before.

In Section 4-5 we said that we would "first" consider the case of generalized finite automata where both inputs and outputs were trees. There are many other possibltities for output domains. Maintaining the restriction to binary trees as inputs, the form of Definitions 7, 8, 9, and 7′ suggests that the operation "$\leftarrow$" of substitution could be replaced by any collection of binary operations indexed by the values of the output function A. Thus, in general, the output universe of discourse could be any algebraic system with associated binary functions. A would take those binary functions as values and A_0 would take values in the carrier of the algebraic system. (It should be emphasized that all this generality is automatic with the algebraic formulation.) Looking at Definition 7, we see that this is exactly the situation. The carrier of the algebraic system is T_Ω and corresponding to any $t \in T_{\Omega, X_2}$ (which might be the value of A) is the binary operation on T_Ω; $t(t_1, t_2) = t \leftarrow [t_1, t_2]$.

This generalized situation can be phrased in the following definition.

DEFINITION 9′

Let $\mathfrak{B} = \langle B, \beta_1, \ldots, \beta_k, \ldots \rangle$ be an algebra where B (the carrier) is a set and the β_i are binary operations on B. A pair of maps, $A: \Sigma \longrightarrow \{\beta_1, \ldots, \beta_k, \ldots\}$ and $A_0: \Sigma \longrightarrow B$ determine a homomorphism $\bar{A}$ from T_Σ into $\mathfrak{B}$ by:

a. If t is a single node labeled σ, then $\bar{A}(t) = A_0(\sigma)$.
b. If t has the form of Figure 11, then $\bar{A}(t) = A(\sigma)(\bar{A}(t_1), \bar{A}(t_2))$.

Although the notation and terminology differ slightly from Petrone's, the following proposition provides the connection between his ideas and the definitions presented here.

PROPOSITION 1

The concept of $\mathfrak{B}$-semantics (Petrone, 1965) of a context-free language $V = L(G)$ coincides with the concept of homomorphism defined on $T_\wp(G)$ taking values in $\mathfrak{B}$.

Within the general algebraic setting of Definition 9′, it is possible to prove a generalization of Theorem 10, but it reads, roughly speaking, that the linear maps preserve equational sets in the sense of Mezei and Wright (1967). As proved in that paper, under the conditions of Theorem 10 the equational and recognizable sets coincide.

A specific application, more like those in Section 4-4, is obtained when $\mathfrak{B}$ is taken to be the structure $\langle N^k, + \rangle$ (N is the set of nonnegative integers) and the input alphabet is $\{\sigma_1, \ldots, \sigma_k\}$. The homomorphism determined by $A_0(\sigma_i) = \langle 0 \ldots 010 \ldots 0 \rangle$ (1 in the ith position) and $A(\sigma_i) = +$ $(1 \leq i \leq k)$ maps a tree t into the k-tuple $(n_1, \ldots, n_k)$, where n_i is the number of occurrences of σ_i in $\mathrm{fr}(t)$. The general theorem referred to above gives a neat algebraic proof of Parikh's theorem (Parikh, 1961) because the semilinear sets are equational.

Similar considerations with Σ^* and concatenation replacing N^k and $+$ yield an alternative proof of the second half of Theorem 6. It should be emphasized again that these examples are special cases of one general theorem and that theorem is proved using essentially the ideas of finite automata theory.

Attention in later work has focused on the study of Irons's specific idea of syntax-directed translation. Using the terminology of Thatcher (1970), any map μ from trees to trees determines a *translation* τ_μ which is the set of ordered pairs $\langle \mathrm{fr}(t), \mathrm{fr}(\mu(t)) \rangle$ for t in the domain of μ. For an arbitrary set U of trees in the domain of μ, τ_μ restricted to $\mathrm{fr}(U)$ should in general be called a syntax-directed translation of $\mathrm{fr}(U)$ because the translation (or translations) of a string in $\mathrm{fr}(U)$ is determined by a mapping of the "structure" of that string, (i.e., the syntax of that string).

Of course, that idea of syntax-directed translation is far more general

than Irons's original idea or any of the formalizations in the literature as is indicated by the following comparisons.

PROPOSITION 2

Any "syntax correspondence" with source grammar G as defined by Younger (1967) is a nondeterministic homomorphism of $T_\wp(G)$.

PROPOSITION 3

Any "syntax-directed translation" with source grammar G as defined by Lewis and Stearns (1968) is a linear homomorphism of $T_\wp(G)$ as is any "syntax-directed translation scheme" with source grammar G as defined by Aho and Ullman (1969a, 1969b).

PROPOSITION 4

Any "generalized syntax-directed translation scheme" with source grammar G as defined by Aho and Ullman (1968) corresponds to an RFAO map A defined on $T_\wp(G)$.

PROPOSITION 5

A "nondeterministic transformation" on T_Σ as defined by Rounds (1970) is equivalent to a nondeterministic RFAO map $\bar{A}$ on $\wp(T_\Sigma)$.

The five propositions above relate various formalizations in the area of translations and semantics to the finite automata definitions of Section 4-4. Propositions 2 and 3 do not present equivalences because additional restrictions must be placed on the (very simple) notion of homomorphism to get exact equivalence. Taking only one of the cases for the concept of syntax-directed translation as defined by Irons, Lewis and Stearns, and Aho and Ullman, one has to add the following condition to the basis function A of the homomorphism. For $\sigma, \sigma_1, \sigma_2$, let $t = A(\sigma, \sigma_1, \sigma_2)$. Then t must be of height 2, must have root σ, and the special symbol x_i can occur on the frontier of t if and only if σ_i is a nonterminal of G. Thus we might introduce the appropriate definitions of terminology to say that a syntax-directed translation is a linear, internal-label-preserving, primitive (for height 2), modified-rank-preserving, homomorphism of $T_\wp(G)$. It is interesting to note that when reflected back to the conventional (monadic) finite automata theory through the algebraic formulation, the only syntax-directed translation is the identity!

There is one notable omission in our list of comparisons (and I apologize to others whose definitions I have not taken note of) and that is the treatment of semantics by Knuth (1968b). It should be fairly clear that his idea of "synthesized attributes" corresponds to frontier-to-root automata with appropriate output domains and that the calculation of "inherited attributes"

corresponds to root-to-frontier automata with appropriate output. It must certainly occur to the automata theorist that Knuth's semantics may find a formulation in terms of two-way tree automata.

The view presented here is simply the thesis that finite automata theory, generalized and algebraically formulated, has the potential for significant application in theoretical computer science.

We have presented arguments in support of that thesis on both sides of the dichotomy (if it should be called that) between recognition and translation, or for lack of better words, between syntax and semantics. The difference between the arguments on the two sides should be striking. On the one hand, we listed examples of several theorems in context-free language theory which can be proved from the finite automata theory point of view. Besides simplifying proofs, I believe that one obtains significantly more insight into the theorems with that approach.

On the other side, with the exception of Parikh's theorem, the arguments in support of the thesis rest solely on showing that certain definitions can be neatly (i.e., algebraically) reformulated using generalized finite automata with output.

One reason for this striking difference is obvious; the overwhelming amount of related work in theoretical computer science has been on the recognition-syntax side of the dichotomy, the formalism is fairly settled, the concepts are well known, and there is a great wealth of theorems from which to choose. But there are theorems on the other side of the dichotomy. Some of those in the papers cited in Section 4-5 can be eliminated or simplified through the algebraic approach.

The number one priority in the area is a careful assessment of the significant problems concerning natural language and programming language semantics and translation. If such problems can be found and formulated, I am convinced that the approach informally surveyed here can provide a unifying framework within which to study them.