AN ABSTRACT OF THE THESIS OF

DAVID LEO VOMOCIL for the degree MASTER OF SCIENCE
(Name) (Degree)

in COMPUTER SCIENCE presented on May 22, 1975
(Major Department) (Date)

Title: A GENERALIZATION OF TREE AUTOMATA AND ITS RELATION TO MATRIX GRAMMARS

Abstract approved: Redacted for Privacy
Curtis R. Cook

In this thesis we introduce alpha and beta tree acceptors, generalizations of tree automata. The alpha tree acceptors recognize a tree by final symbol and the beta tree acceptors by final state. We show that alpha and beta tree acceptors recognize the same sets of Gorn trees and demonstrate that there are sets of Gorn trees not recognizably by any alpha tree acceptor. The tree automata are equivalent to the one state alpha tree acceptors. We show that the languages of alpha tree acceptors are exactly those of matrix grammars and the derivation trees of matrix grammars are recognized by alpha tree acceptors. Finally we show that nondeterministic alpha tree acceptors are no more powerful than deterministic alpha tree acceptors.

A Generalization of Tree Automata and Its Relation to Matrix Grammars

by

David Leo Vomocil

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

June 1976

APPROVED:

Redacted for Privacy

Assistant Professor of Computer Science

in charge of major

Redacted for Privacy

Chairman of Department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented May 22, 1975

Typed by Clover Redfern for David Leo Vomocil

TABLE OF CONTENTS

A GENERALIZATION OF TREE AUTOMATA AND ITS RELATION TO MATRIX GRAMMARS

I. INTRODUCTION

Brainerd [1], Doner [3], and Thatcher [11] have studied tree automata. Brainerd introduced regular generating systems and used them to establish a strong relationship between the sets of derivation trees of context-free grammars and the sets of trees recognized by tree automata.

Others [8] have studied syntactic methods of pattern recognition. Fu and Bhargava [4] have used context-free grammars and tree automata to build pattern recognizers for bubble chamber patterns. It was this application of tree automata that led us to generalize tree automata.

The input for tree automata are Gorn trees. A tree automata accepts a Gorn tree by constructing a bottom-up parse tree called an accepting G-run [11]. The parse tree has the same structure as the input tree. The labels of the parse tree are determined by the labels of the input tree and the next state function of the tree automata. The tree is accepted if the label of the root of the parse tree is an accepting state of the tree automata.

In this thesis we introduce α and β tree acceptors, generalizations of tree automata. We remove the states of our

accepting devices, alpha and beta tree acceptors, from the input. Alpha or beta tree acceptors parse input trees by reducing subtrees of the frontier of the input trees to single symbols. These symbols used to replace the subtrees are determined by the present state of the device and the subtree being reduced. The alpha and beta tree acceptors also change their state as a function of the present state and the subtree being reduced. Acceptance of a Gorn tree is determined by considering the final symbol written in the case of alpha tree acceptors and determined by considering the final state entered in the case of beta tree acceptors. We show that these two types of acceptance are equivalent. We show further that alpha tree acceptors with only one state have the same power as tree automata. We then give a diagonal argument demonstrating that some sets of Gorn trees are not recognized by any alpha tree acceptor. Boolean operators are considered, but we are only able to obtain results for the union operation.

In our last two sections we demonstrate that matrix grammars [9] are related to alpha or beta tree acceptors in much the same manner that context-free grammars are related to tree automata. In particular, we show that for every λ-free context-free matrix grammar there is an alpha (beta) tree acceptor such that the languages of both are equal and such that the derivation trees of the matrix grammar form a set of Gorn trees that is nearly structurally equivalent to the set of Gorn trees accepted by the alpha (beta) tree acceptor. The

proof of this last result is constructive and is used to show that nondeterministic alpha (beta) tree acceptors are no more powerful than deterministic alpha (beta) tree acceptors.

II. DEFINITIONS

The input for our alpha tree acceptors and beta tree acceptors are Gorn trees. Gorn trees are defined below and in Gorn [5], Doner [3], Brainers [1], Thatcher [11] and Korfhage [7].

Let N^+ be the set of positive integers. Let U be the free monoid generated by N^+ with the operation $\cdot$, and let 0 be the identity of U.

Definition 2.1. The depth of $a \in U$ is denoted $d(a)$ and is defined by $d(0) = 0$ and $d(a \cdot i) = d(a) + 1$, $i \in N^+$.

For $a, b \in U$, $a \leq b$ if and only if there exists an $x \in U$ such that $a \cdot x = b$. Two elements of U, a and b, are incomparable if and only if $a \not\leq b$ and $b \not\leq a$.

Figure 2.1 illustrates this partial ordering on U.

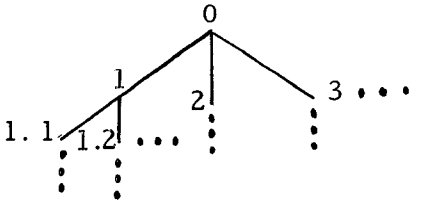


Figure 2.1.

Definition 2.2. D, a subset of U, is a tree domain if and only if 1) $b \in D$ and $a \leq b$ implies that $a \in D$ and 2) $a\,i \in D$

and $j < i$ in N^+ implies that $a \cdot j \in D$.

The set $A = \{0, 1, 2, 1\cdot1, 1\cdot2, 2\cdot1\}$ is a tree domain and is graphically represented by Figure 2.2.

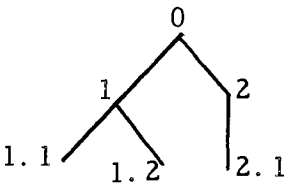


Figure 2.2.

The set $B = \{0, 1, 2, 1\cdot2, 2\cdot1, 2\cdot3, 3\cdot1\}$ is not a tree domain since $1\cdot1$, $2\cdot2$, and 3 are not elements of B.

Definition 2.3. A stratified alphabet is a pair $<\Sigma, r>$, where Σ is a finite set of symbols and $r : \Sigma \rightarrow N = N^+ \cup \{0\}$. For $x \in \Sigma$, $r(x)$ is called the stratification of x. A stratified alphabet is also called a ranked alphabet.

Definition 2.4. A k-tree domain is a tree domain over the set $\{0, 1, 2, \ldots, k\}$ instead of $\{0, 1, 2, \ldots\}$; that is, we allow each node to have at most k branches.

Definition 2.5. A (k, Σ) value tree is a pair (D, t) where D is a k-tree domain and t is a mapping from D into Σ. This provides a method for labeling nodes of D using elements of an

alphabet Σ. When the tree domain D and the mapping t are not important we sometimes refer to the (k, Σ) value tree (D, t) as the tree α.

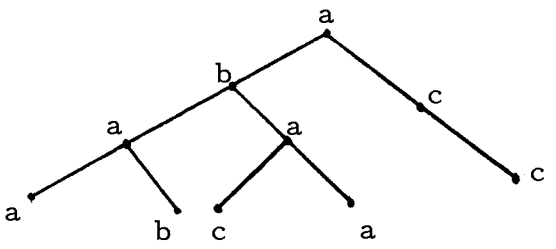


Figure 2.4.

Example 2.6. $\{0, 1, 2, 1\cdot1, 1\cdot2, 2\cdot1, 1\cdot1\cdot1, 1\cdot1\cdot2, 1\cdot2\cdot1, 1\cdot2\cdot2\}$ is a 2-tree domain. Defining the mapping t by $\{(0,a), (1,b), (2,c), (1\cdot1,a), (1\cdot2,a), (2\cdot1,c), (1\cdot1\cdot1,a), (1\cdot1\cdot2,b), (1\cdot2\cdot1,c), (1\cdot2\cdot2,a)\}$ where $\Sigma = \{a, b, c\}$ we obtain the $(2, \Sigma)$ value tree of Figure 2.4.

Definition 2.7. Let D be a tree domain. $x \in D$ is maximal if and only if there does not exist $y \in D$ such that $x \leq y$ and $x \neq y$. That is, x is maximal if and only if x is a leaf of D.

Definition 2.8. $x <\cdot y$ where x and y are elements of a tree domain D if and only if there exists a $z \in D$ such that $z\cdot i \leq y$ for some $i \in N^+$, $z\cdot j \leq x$ for some j, and $j < i$. That is, $x <\cdot y$ if and only if x is to the left of y.

Definition 2.9. $Fr(D) = s_1 s_2 \cdots s_n$ where 1) if s is a maximal element of D then there exists an i, $1 \leq i \leq n$ such that $s = s_i$ and 2) $s_i <\bullet s_{i+1}$. $Fr(D)$ is called the frontier of D.

Definition 2.10. $S_D(x) = (x\bullet 1, x\bullet 2, \ldots, x\bullet n)$ where 1) $x\bullet i \in D$ for $1 \leq i \leq n$ and 2) $x\bullet i \notin D$ for $i > n$. $S_D(x)$ is called the immediate successors of x in D.

Definition 2.11. $fr(D,t) = t(Fr(D))$

$$= t(x_1\ x_2\ \cdots\ x_n)$$

$$= t(x_1)t(x_2)\cdots t(x_n)$$

where $Fr(D) = x_1, x_2, \ldots, x_n$.

In Example 2.6, $fr(d,t) = t(Fr(D))$

$$= t(1\bullet 1\bullet 1, 1\bullet 1\bullet 2, 1\bullet 2\bullet 1, 1\bullet 2\bullet 2, 2\bullet 1)$$

$$= abcac$$

We extend the notion of depth to trees in the following manner. The depth of (D,t), denoted $d((D,t)) = \max\{d(x) : x \text{ is maximal in } D\}$.

Definition 2.12. $T_{k,\Sigma}$ is the set of all (k,Σ) value trees.

Different types of Polish notation have been used to represent trees using strings of elements from the alphabet. For example, the tree of Figure 2.5 can be represented by the string $+*abc$ using

Polish prefix notation and the string ab*c+ using Polish postfix notation.

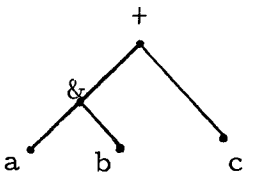


Figure 2.5.

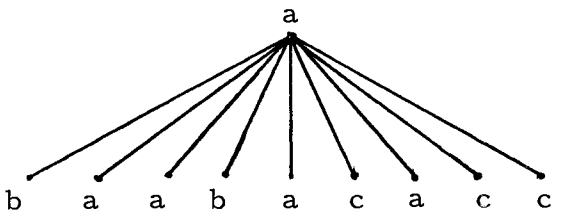


Figure 2.5.

These types of Polish notation are convenient when a distinction is made between terminal symbols and nonterminals. If we let $<\Sigma, r>$ be a stratified alphabet, and we define Σ_i, for $i \geq 0$, to be the set $\{x : x \in \Sigma \text{ and } r(x) = i\}$, then the Polish notations are convenient to use when it is the case that:

$$\Sigma_0 \cap \{\bigcup_{i>0} \Sigma_i\} = \phi \tag{2.1}$$

But in general we will not assume that 2.1 is true. When 2.1 is not true the Polish notations become ambiguous. For example, the Polish notation for the tree of Figure 2.4 is abaabacacc; but that is also the Polish prefix notation for the tree of Figure 2.6. To resolve this difficulty we define tree expressions. Similar notations are discussed in Thatcher [11] and Doner [3].

Definition 2.13. A tree expression over an alphabet Σ is

defined recursively as:

1) if $x \in \Sigma$, then x is a tree expression and

2) if $t_1, t_2, \ldots, t_n$ are tree expressions and $x \in \Sigma$, then $x(t_1 t_2 \cdots t_n)$ is also a tree expression where we assume that parentheses are not elements of Σ.

Thus the tree expression for the tree of Figure 2.4 is a(b(a(ab)a(ca))c(c)), and the tree expression for the tree of Figure 2.6 is a(baabacacc).

We now generalize the notion of tree automata. The input for generalized tree automata will be (k, Σ) value trees. To facilitate these definitions we will refer to the (k, Σ) value tree (D, t) by the single letter α. The domain of α, symbolized by $D(\alpha)$, will be the implied tree domain D. The following two definitions are from Brainerd [1] and, hopefully, will make the definitions of our generalized tree automata easier to follow.

Definition 2.14. If $\alpha \in T_{k,\Sigma}$ and $a \in D(\alpha)$, then $\alpha/a = \{(b, x): (a \cdot b, x) \in \alpha\}$. That is, α/a is the subtree of α at a. Note that here the x's refer to elements of Σ and the pairs (b, x) refer only to nodes of labeled trees.

Figure 2.7.

Definition 2.15. Let $a \in U$, the universal tree domain, and $\beta \in T_{k,\Sigma}$ $a \bullet \beta = \{(a \bullet b, x) : (b, x) \in \beta\}$.

Definition 2.16. Let $\alpha, \beta \in T_{k,\Sigma}$ and $a \in D(\alpha)$, then $\alpha(a \leftarrow \beta) = \{(b,x) : (b,x) \in \alpha \text{ and } a \not\leq b\} \cup a \bullet \beta$. That is, the subtree of α at a is replaced by β.

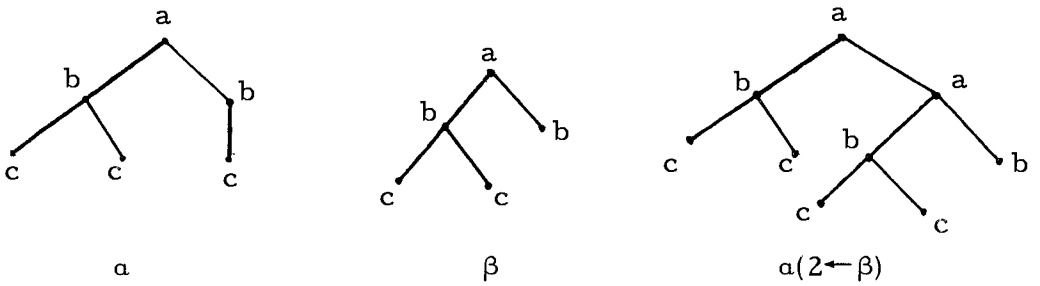


Figure 2.8.

Definition 2.17. An Alpha Tree Acceptor is a system $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ where

Q is a finite set of states,

$q_0 \in Q$, the initial state of the acceptor,

Ω is a finite alphabet,

δ is a finite set of state transition-tree reductions,

$\Sigma \subseteq \Omega$ (R will accept (k, Σ) value trees for some k), and

$\theta \subseteq \Omega$ is a set of final or accepting symbols.

The δ-rules define the manner in which R can affect a bottom up parse on a (k, Σ) value tree. These rules are of the form $\delta(q, \beta) = (p, \omega)$ where $q, p \in Q$, $\omega \in \Omega$, and β is a (k, Ω) value tree. When R is in state q it can apply the reduction $\delta(q, \beta) = (p, \omega)$ to the (k, Ω) value tree α if there exists an $a \in D(\alpha)$ such that $\alpha/a = \beta$. In this case R reduces the tree α to the tree α' where $\alpha' = \alpha(a \leftarrow \omega)$. When the reduction is completed R moves into state p.

In Definitions 2.18 thru 2.23 we will be assuming that R is the alpha tree acceptor $(Q, \Omega, \delta, q_0, \Sigma, \theta)$.

Definition 2.18. A configuration of R is an ordered pair (q, β) where $q \in Q$ and β is a (k, Ω) value tree.

As R parses a tree it moves from one configuration to another. This action will be symbolized by a $\vdash$ as per the following definition.

Definition 2.19. Let $q, p \in Q$ and β, γ be (k, Ω) value

trees. $(q, \beta) \vdash (p, \gamma)$ in R if and only if there exists a δ-rule for R of the form $\delta(q, a) = (p, \omega)$ and $a, b \in D(\beta)$ such that $\beta/b = a$ and $\gamma = \beta(b \leftarrow \omega)$.

Definition 2.20. Let $q, p \in Q$ and β, γ be (k, Ω) value trees. $(q, \beta) \vdash^{*} (p, \gamma)$ in R if and only if there exists sequences $\{q_i\}_{i=1,2,\ldots,m}$ and $\{\alpha_i\}_{i=1,2,\ldots,m}$ where $q_i \in Q$ and the α_i are all (k, Ω) value trees for $i = 1, 2, \ldots, m$, $q_1 = q$, $q_m = p$, $\alpha_1 = \beta$, $\alpha_m = \gamma$, and $(q_i, \alpha_i) \vdash (q_{i+1}, \alpha_{i+1})$ for $i = 1, 2, \ldots, m-1$. That is, $\vdash^{*}$ is the reflexive transitive closure of $\vdash$. If $m = 0$ then we mean $(q, \beta) = (p, \gamma)$; that is, R makes no moves.

Definition 2.21. Let α be a (k, Ω) value tree. The alpha tree acceptor R accepts α if and only if $(q_0, \alpha) \vdash^{*} (p, \beta)$ where $\beta = \{(0, \omega)\}$ and $\omega \in \theta$.

Definition 2.22. $A(R) = \{\alpha : \alpha$ is a (k, Σ) value tree and R accepts $\alpha\}$.

Definition 2.23. Γ, a set of (k, Σ) value trees, is said to be α-recognizable if and only if there exists an alpha tree acceptor R such that $\Gamma = A(R)$.

Definition 2.24. An alpha tree acceptor R is said to be simple if and only if all δ-rules for R are of the form

$\delta(q, \beta) = (p, \omega)$ and $d(\beta) \leq 1$.

Lemma 2.25. Let $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ be an alpha tree acceptor. There exists a simple alpha tree acceptor $R' = (Q', \Omega', \delta', q_0, \Sigma, \theta)$ such that $A(R) = A(R')$.

Proof: The proof offered here is similar to the proof used by Brainerd [1] to demonstrate that he need only consider simple regular systems. For that reason we only outline the construction here. The construction involves forcing a level by level reduction of the tree. For each offending δ-rule of R we add a sequence of δ'-rules to R'. The sequences of δ'-rules in R' are used to simulate the respective δ-rules of R.

For each δ-rule of R, $\delta(q, \beta) = (p, \omega)$, define the sequence of δ'-rules for R'

$$\{\delta'(q'_i, \beta'_i) = (q'_{i+1}, \omega'_{i+1})\}_{i=0, 1, \ldots, m-1}$$

where $q'_0 = q$, $q'_m = p$, $q'_i \in Q'$ and $q'_i \notin Q$ for $i = 1, 2, \ldots, m-1$, the β'_i are all (k, Ω') value trees such that $d(\beta'_i) \leq 1$ for $i = 0, 1, \ldots, m-1$, $\omega'_m = \omega$, $\omega'_i \in \Omega'$, and $\omega'_i \notin \Omega$ for $i = 1, 2, \ldots, m$. In this case $(q, \alpha) \vdash (p, \gamma)$ in R using $\delta(q, \beta) = (p, \omega)$ if and only if $(q, \alpha) \vdash^* (p, \gamma)$ in R' using the sequence $\{\delta'(q'_i, \beta'_i) = (q'_{i+1}, \omega'_{i+1})\}_{i=0, 1, \ldots, m-1}$. QED

As a result of Lemma 2.25 we assume from now on that any alpha tree acceptor is a simple alpha tree acceptor unless otherwise stated.

Definition 2.26. A Beta Tree Acceptor is a system $S = (Q, \Omega, \delta, q_0, \Sigma, \Phi)$ where Q, Ω, δ, q_0, and Σ are as for alpha tree acceptors. The difference being that for beta tree acceptors we have a set of final states instead of a set of final symbols; i.e., $\Phi \subseteq Q$.

Definitions 2.18, 2.19, and 2.20 concerning the operators $\vdash$ and $\vdash^*$ apply also to beta tree acceptors.

Definition 2.27. Let α be a (k, Σ) value tree. The beta tree acceptor $S = (Q, \Omega, \delta, q_0, \Sigma, \Phi)$ accepts α if and only if $(q_0, \alpha) \vdash^* (p, \beta)$ where $p \in \Phi$ and $\beta = \{(0, \omega)\}$. Note that for S to accept α, S must still be able to reduce α to a single symbol.

Definition 2.28. Let $S = (Q, \Omega, \delta, q_0, \Sigma, \Phi)$ be a beta tree acceptor. $B(S) = \{\alpha : \alpha$ is a (k, Σ) value tree and S accepts $\alpha\}$.

Definition 2.29. Let Γ be a set of (k, Σ) value trees for some alphabet Σ and some k. Γ is said to be β-recognizable if and only if there exists a beta tree acceptor S such that $\Gamma = B(S)$.

Definition 2.30. A beta tree acceptor S is said to be simple

if an only if all δ-rules are of the form $\delta(q, \beta) = (p, \omega)$ and $d(\beta) \leq 1$.

Corollary 2.31. Let $S = (Q, \Omega, \delta, q_0, \Sigma, \Phi)$ be a beta tree acceptor. There exists a simple beta tree acceptor $S' = (Q', \Omega', \delta', q_0, \Sigma, \Phi)$ such that $B(S) = B(S')$.

Proof: Merely apply the construction of Lemma 2.25. QED

Again as a result of the above corollary we assume from now on that any beta tree acceptor is a simple beta tree acceptor unless otherwise stated.

In both Definitions 2.24 and 2.30 we stipulated that $d(\beta) \leq 1$; that is, we allow δ-rules of the form $\delta(q, \beta) = (p, \omega)$ where $\beta = \{(0, \nu)\}$. We refer to these δ-rules as relabeling rules, since the automaton merely relabels a leaf of a tree and changes its state when such a rule is executed. These rules appear to add power to the respective automaton; therefore, we conjecture that acceptors that have relabeling rules accept more subsets of $T_{k, \Sigma}$ than acceptors that do not have such rules.

One last remark concerning notation. Frequently we will be working with trees of depth zero. In many such cases we will refer to these trees as merely elements of their alphabet. In particular if $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ is an alpha tree acceptor and α is a (k, Σ) value tree in $A(R)$ we have $(q, \alpha) \vdash^* (p, \beta)$. If $\beta = \{(0, \omega)\}$, then we often refer to β as ω.

III. α-RECOGNIZABLE IFF β-RECOGNIZABLE

In this section we show that a set of (k, Σ) value trees is α-recognizable if and only if it is β-recognizable. The proof is conceptually simple but tedious to write down. Heuristically, to go from α-recognizable to β-recognizable we merely build a beta tree acceptor with states that remember what symbol was just written. Similarly, to show that a β-recognizable set is α-recognizable we construct an alpha tree acceptor that writes symbols that indicate what state the automaton entered when the symbol was written.

Lemma 3.1. If the set Γ of $(k\Sigma)$ value trees is α-recognizable, then Γ is β-recognizable.

Proof: Let $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ be an alpha tree acceptor such that $\Gamma = A(R)$. We construct the betta tree acceptor $S = (Q', \Omega, \delta', q_0, \Sigma, \Phi)$ where

$$Q' = q_0 \cup \{(q, \omega): (q, \omega) \text{ is the right hand side of some } \delta\text{-rule of } R\},$$

and

$$\Phi = \{(q, \omega): \omega \in \theta \text{ in } R, \text{ and } q \in Q\}.$$

The δ'-rules for S include $\delta'(q_0, \beta) = ((q, \omega), \omega)$ for all β such that $\delta(q_0, \beta) = (q, \omega)$ is in δ of R. We also need δ'-rules of the form $\delta'(q_0, \sigma) = ((q_0, \sigma), \sigma)$ for all $\sigma \in \Sigma \cap \theta$. The later rules are

for the case when R accepts trees of the form σ where $\sigma \in \Sigma$. Note that in such cases R makes no moves.

Finally, we need to include for each δ-rule, $\delta(q, \beta) = (p, \omega)$, of R and for each $\tau \in \Omega$ such that $(q, \tau) \in Q'$ a δ'-rule for S of the form:

$$\delta'((q, \tau), \beta) = ((p, \omega), \omega).$$

We claim that $A(R) = B(S)$. To see this we first assume that the (k, Σ) value tree $\alpha \in A(R)$. Then $(q_0, \alpha) \vdash^* (q, \omega)$ in R and $\omega \in \theta$. This insures that we can find the following sequences:

$$(3.1.1) \qquad \{(q'_i, \beta_i) \vdash (q'_{i+1}, \beta_{i+1})\}_{i=0, 1, \ldots, m-1}, \quad \text{for some } m.$$

Here $q'_0 = q_0$ of R, $\beta_0 = \alpha$, and $\beta_m = \omega$. This sequence implies the following sequence of δ-rules for R exists.

$$(3.1.2) \qquad \{\delta(q'_i, \gamma_i) = (q'_{i+1}, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}$$

In (3.1.1) and (3.1.2) the same sequence of states $\{q'_i\}_i$ is used. Furthermore, a sequence of elements $\{b_i\}_i$, $b_i \in D(\beta_i)$, can be found such that $\beta_i / b_i = \gamma_i$ and $\beta_{i+1} = \beta_i(b_i \leftarrow \omega_{i+1})$ for $i = 0, 1, \ldots, m-1$.

For $i = 1, 2, \ldots, m$ let $q''_i = (q'_i, \omega_i) \in Q'$, $q''_0 = q$. Then for S we can write:

$$\{\delta'(q_i'', \gamma_i) = (q_{i+1}'', \omega_{i+1})\}_{i=0, 1, \ldots, m-1}$$

This insures $\{(q_i'', \beta_i) \vdash (q_{i+1}'', \beta_{i+1})\}_{i=0, 1, \ldots, m-1}$ holds for S. Hence for S we can write $(q_0, \alpha) \vdash^{*} (q_m'', \omega_m) = (q_m'', \omega)$. $q_m'' = (q_m', \omega_m)$ and $\omega_m = \omega \in \theta$; therefore, $q_m'' \in \Phi$ and $\alpha \in B(S)$. That is, we have $A(R) \subseteq B(S)$.

Similarly we show that $B(S) \subseteq A(R)$. If the (k, Σ) value tree $\alpha \in B(S)$, then we can write:

(3.1.3) $$(q_0, \alpha) \vdash^{*} (q, \omega) \quad \text{where} \quad q \in \Phi.$$

From the definition of the δ'-rules we know that $q = (q', \omega)$ and $\omega \in \theta$. (3.1.3) insures the existence of states $q_i'' \in Q'$ and trees β_i, for $i = 0, 1, \ldots, m$, such that

$$\{(q_i'', \beta_i) \vdash (q_{i+1}'', \beta_{i+1})\}_{i=0, 1, \ldots, m-1}$$

holds in S for some number m, where $q_0'' = q_0$, $\beta_0 = \alpha$, $q_m'' = q$, and $\beta_m = \omega$. If $q_j'' = (q_j', \omega')$, then $p_i = q_i'$ for $i = 1, 2, \ldots, m$ and $p_0 = q_0$ of R. Then using the definition of the δ'-rules we can write:

$$\{(p_i, \beta_i) \vdash (p_{i+1}, \beta_{i+1})\}_{i=0, 1, \ldots, m-1}$$

for R. Thus $(q_0, \alpha) \vdash^{*} (p_m, \beta_m)$ in R. $\beta_m = \omega \in \theta$ insures $\alpha \in A(R)$ and $B(S) \subseteq A(R)$.

Thus we have $A(R) = B(S)$ as desired. QED

Lemma 3.2. Let Γ be a set of (k, Σ) value trees. If Γ is β-recognizable, then Γ is α-recognizable.

Proof: Γ is β-recognizable implies that there exists a beta tree acceptor $S = (Q, \Omega, \delta, q_0, \Sigma, \Phi)$ such that $\Gamma = B(S)$. We construct the alpha tree acceptor $R = (Q, \Omega', \delta, q_0, \Sigma, \theta)$ where $\Omega' = (Q \times \Omega) \cup \Sigma$.

The construction of the δ'-rules will be more involved here than for Lemma 3.1. First, for all $q \in Q$ and $\sigma \in \Sigma$ we define the δ'-rule $\delta'(q, \sigma) = (q, (q, \sigma))$. These rules will allow S to introduce meaningful symbols on the frontier of a tree.

We now consider all δ-rules of S. They are of the form $\delta(q, \beta) = (p, \omega)$. For all such δ-rules introduce δ'-rules for R defined by the following. If $fr(\beta) = \omega_1\omega_2 \ldots \omega_n$ then form all ordered sets of size n using elements from Q. There are, of course, $|Q|^n$ such sets where $|Q|$ = the number of elements in Q. We will denote the ith such ordered pair by $(q_1^i, q_2^i, \ldots, q_n^i)$ where $q_j^i \in Q$ for $j = 1, 2, \ldots, n$. Using these n-tuples and $fr(\beta)$ we form the n-tuples

$$\psi_i = ((q_1^i\omega_1)(q_2^i\omega_2) \ldots (q_n^i, \omega_n)) \quad \text{for } i = 1, 2, \ldots, |Q|^n .$$

For each $i \in \{1, 2, \ldots, |Q|^n\}$ we take q, p, ω, and β from the δ-rule mentioned above. We let $\beta_i = \beta$ except $fr(\beta_i) = \psi_i$. Then we add to δ' the δ'-rule $\delta'(q, \beta_i) = (p, \omega)$.

Finally we set $\theta = \{(p, \omega): p \in \Phi\}$ and we again claim $B(S) = A(R)$.

If the (k, Σ) value tree $\alpha \in B(S)$, then $(q_0, \alpha) \vdash^* (q, \omega)$ in S where $q \in \Phi$ and $\omega \in \Omega$. Hence we know that a sequence of trees $\{\beta_i\}_i$ and a sequence of states $\{q_i\}_i$ exists such that

$$(3.2.1) \qquad \{(q'_i, \beta_i) \vdash (q'_{i+1}, \beta_{i+1})\}_{i=0, 1, \ldots, m-1}$$

for some m, $q'_0 = q_0$ of S, $\beta_0 = \alpha$, $q'_m = q$, and $\beta_m = \omega$. This sequence implies the following sequence of applicable δ-rules also exists.

$$(3.2.2) \qquad \{\delta(q'_i, \gamma_i) = (q'_{i+1}, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}.$$

Associated with sequences (3.2.1) and (3.2.2) is the sequence $\{b_i\}_i$ such that $b_i \in D(\beta_i)$, $\beta_i/b_i = \gamma_i$, and $\beta_{i+1} = \beta_i(b_i \leftarrow \omega_{i+1})$.

To see that $\alpha \in A(R)$ first if $fr(\alpha) = \omega_1\omega_2 \ldots \omega_n$ set $\delta(q_0, \omega_i) = (q_0, (q_0, \omega_i))$ for $i = 1, 2, \ldots, n$. Using this sequence and sequences (3.2.1) and (3.2.2) we can form the sequence $\{\gamma'_i\}_i$ where $\gamma'_i = \gamma_i$ except $fr(\gamma'_i) \in (Q \times \Omega)^{\ell}$ for some ℓ. Using this sequence and (3.2.2) we can write

(3.2.3) $\{\delta'(q'_i, \gamma'_i) = (q'_{i+1}, (q'_{i+1}, \omega_{i+1}))\}_{i=0, 1, \ldots m-1}$ for R.

Furthermore, if $\beta'_i = \beta_i$ except $fr(\beta'_i) \in (\Omega x \Omega)^{\ell}$ for some ℓ we can use (3.2.2) to obtain

(3.2.4) $\{(q'_i, \beta'_i) \vdash (q'_{i+1}, \beta'_{i+1})\}_{i=0, 1, \ldots, m-1}$.

The β'_i are such that using $\{b_i\}_i$ we have $\beta'_i/b_i = \gamma'_i$ and $\beta'_{i+1} = \beta_i(b_i \leftarrow \omega_{i+1})$.

Concatenating sequences (3.2.3) and (3.2.4) we have $\alpha \in B(S)$ since $q_m = q \in \Phi$ insures that $(q_m, \omega_m) \in \theta$. Therefore $B(S) \subseteq A(R)$.

To obtain $A(R) \subseteq B(S)$ we assume that the (k, Σ) value tree $\alpha \in A(R)$. That is, $(q_0, \alpha) \vdash^{*} (q, \omega)$ is true in R where $\omega \in \theta$. $\omega \in \theta$ implies $\omega = (q', \omega')$ and $q' \in \Phi$. The above assures that we can find the sequence

(3.2.5) $\{(q'_i, \beta'_i) \vdash (q'_{i+1}, \beta'_{i+1})\}_{i=0, 1, \ldots, m-1}$, for R.

As before we can construct the corresponding set of δ'-rules of R,

$$\{\delta'(q'_i, \gamma'_i) = (q'_{i+1}, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}.$$

n of these δ'-rules will be of the form

$$\delta'(q'_i, \sigma) = (q'_i, (q'_i, \sigma))$$

where n = number of symbols in $fr(\alpha)$. Discarding these n rules we obtain the sequence

$$(3.2.6) \qquad \{\delta'(q'_i, \gamma'_i) = (q'_{i+1}, \omega_{i+1})\}_{i=0, 1, \ldots, m'-1}$$

where $m' = m - n$. Using this sequence and sequence (3.2.5) we can form, as before, sequences $\{\gamma_i\}_i$ and $\{\beta_i\}_i$ such that

$$\{\delta(q'_i, \gamma_i) = (q'_{i+1}, \omega_{i+1})\}_{i=0, 1, \ldots, m'-1},$$

and

$$\{(q'_i, \beta_i) \vdash (q'_{i+1}, \beta_{i+1})\}_{i=0, 1, \ldots, m'-1}$$

are both true for S. Hence, $(q_0, \alpha) \vdash^* (q'_{m'}, \beta_{m'})$ where $\beta_0 = \alpha$ and $q'_m = q'_m = q' \in \Phi$ holds for S. That is, $\alpha \in B(S)$ and we have $A(R) \subseteq B(S)$.

Therefore $A(R) = B(S)$ as claimed. QED

The two previous lemmas yield the following theorem.

Theorem 3.3. Let Γ be a set of (k, Σ) value trees. Γ is α-recognizable if and only if Γ is β-recognizable.

IV. ALPHA TREE ACCEPTORS AND TREE AUTOMATA

In this chapter we define tree automata and demonstrate that a set of (k, Σ) value trees Γ is recognizable by a tree atuomata if and only if Γ is α-recognizable by an alpha tree acceptor having a single state. We show further that, in general, alpha tree acceptors are more powerful than tree automata. Nondeterminism for one state alpha tree acceptors is introduced, and results from tree automata theory are applied.

The following definition was taken from Thatcher [11]. Similar definitions can be found in Brainerd [1] and Doner [3].

Definition 4.1. A finite tree automata is a system $G = (Q, \alpha, W, \psi)$ where Q is a set of states. For $\sigma \in \Sigma$, an alaphabet,

$\alpha_\sigma : \bigcup_{1 \leq n \leq k+1} Q^n \rightarrow Q$ are state transition functions,

$W_\sigma \subseteq W$ are initial states, and

$\psi \subseteq Q$ is a set of accepting or final states.

Definition 4.2. $(D, r) \in T_{k, Q}$ is a G-run over $(D, t) \in T_{k, \Sigma}$, denoted $r \in Rn(G, t)$, if and only if

1) $r(x) = W_{t(x)}$ for maximal $x \in D$, and

2) $r(x) = \alpha_{t(x)}(r(S(x)))$ for nonmaximal $x \in D$.

Definition 4.3. $r \in Rn(G, t)$ is an accepting run if $r(0) \in \psi$.

Definition 4.4. Let G be a tree automata. The set of trees accepted by G is $T(G) = \{(D, t)$: there exists a $r \in Rn(G, t)$ and r is an accepting run$\}$.

Definition 4.5. A set Γ of (k, Σ) value trees is TA recognizable if and only if there exists a tree automata G such that $\Gamma = T(G)$.

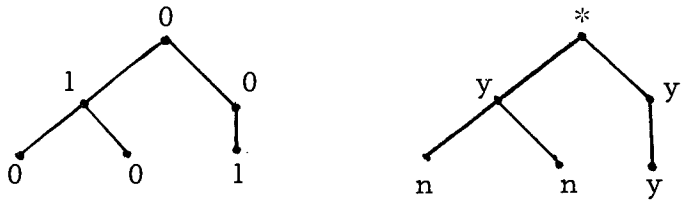
The following example is from Thatcher [11].

Example 4.6. Let $G = (Q, a, W, \psi)$ be a tree automata where $Q = \{y, n, *\}$, $W_0 = n$, $W_1 = y$, and $\psi = \{y\}$. Let the next state function of G be defined by the following table.

$r(S_D(x))$	a_0	a_1
*	*	*
yy	*	*
yn	y	*
ny	y	*
nn	n	y
n	n	y
y	y	*

The input for G is $(2, \{0, 1\})$ value trees. $T(G)$ is the set of $(2, \{0, 1\})$ value trees with exactly one node labeled 1.

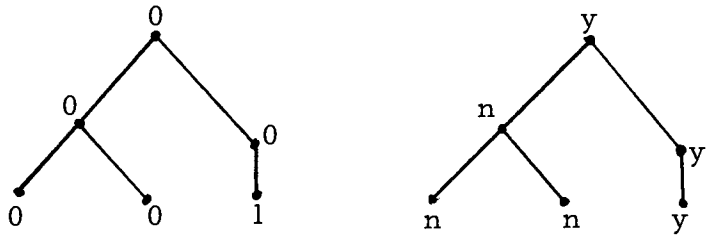
The following is a non-accepting run for G since $* \notin \psi$.



(D, t) (D, r)

Figure 4.1.

Figure 4.2 represents an accepting run for G.



(D, t) (D, r)

Figure 4.2.

Theorem 4.7. A set of (k, Σ) value trees is TA recognizable if and only if it can be recognized by an alpha tree acceptor R with only one state and all δ-rules for R are of the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) = 1$.

Proof: We assume that the set of (k, Σ) value trees, Γ, is TA recognizable. Then there exists a tree automaton $G = (Q, \alpha, W, \psi)$ such that $\Gamma = T(G)$. Construct the alpha tree acceptor $R = (\{q\}, \Omega, \delta, q, \Sigma, \theta)$ where $\Omega = \Sigma \cup Q$ and $\theta = \psi$. It is assumed

here that G accepts a subset of $T_{k,\Sigma}$.

The δ-rules for R are defined in two stages. First, for the α-rules of G we have the following:

$$\delta(q, x(s_1 s_2 \cdots s_n)) = (q, \alpha_x(s_1 s_2 \cdots s_n))$$

where $x \in \Sigma$ and $s_1, s_2, \ldots, s_n \in Q$. $x(s_1 s_2 \cdots s_n)$ is a tree expression for a tree of depth one.

Secondly we must take care of the initial state assignments of the tree automaton. We create δ-rules that by-pass this step. For all α-rules of G of the form $\alpha_x(s_1 s_2 \cdots s_n) = \omega$ where $x \in \Sigma$, $s_1, s_2, \ldots, s_n \in Q$ and at least one $s_j = W_\sigma \in W$ we add δ-rules of the following form:

$$\delta(q, x(x_1 x_2 \cdots x_n)) = (q, \omega) \quad \text{to the } \delta\text{-rules of } R.$$

Here we set $x_i = s_i$ if $s_i \notin W$ and if $s_i = W_{\sigma_i} \in W$, then we set $x_i = \sigma_i$.

We claim that $T(G) = A(R)$. First assume the (k, Σ) value tree $(D, t) \in T(G)$. Using the accepting run (D, r) we can form the sequence $\{\gamma_i\}_{i=0, 1, \ldots, m-1}$ of (k, Ω) value trees, where $\gamma_i = \sigma_{i_0}(x_{i_1} x_{i_2} \cdots x_{i_n})$. The corresponding subtree of (D, t) is $\sigma_{i_0}(\sigma_{i_1} \sigma_{i_2} \cdots \sigma_{i_n})$, and if we let $v = \alpha_{\sigma_{i_0}}(s_i s_i \cdots s_{i_n})$ the

corresponding subtree of (D, r) is $v(s_{i_1} s_{i_2} \ldots s_{i_n})$. If $s_{i_j} = r(x)$ and x is a nonmaximal element of D, then $x_{i_j} = s_{i_j}$. If $s_{i_j} = r(x)$ and x is a maximal element of D, then $x_{i_j} = \sigma_{i_j} = t(x)$.

We also need the sequence $\{\omega_i\}_{i=1, 2, \ldots, m}$ defined by $\omega_i = \alpha_{\sigma_{j_0}} (s_{j_1} s_{j_2} \ldots s_{j_n})$, $j = i-1$. The order of $\{\omega_i\}_i$ is arranged so that $\omega_m = r(0) \in \psi$, and the γ_i can be arranged so that we have

$$\{\delta(q, \gamma_i) = (q, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}.$$

Using the sequence $\{\gamma_i\}_i$ and the trees (D, t) and (D, r) we can define two sequences.

$\{b_i\}_{i=0, 1, \ldots, m-1}$, $b_i \in D$ for all i, and

$\{\beta_i\}_{i=0, 1, \ldots, m'}$ where $\gamma_i = \beta_i / b_i$ for all i, $\beta_0 = (D, t)$, and $\beta_m = \omega_m$.

The β_i's are chosen so that $\beta_{i+1} = \beta_i(b_i \leftarrow \omega_{i+1})$ From this consideration we see that the following sequence can be formed.

$$\{(q, \beta_i) \vdash (q, \beta_{i+1})\}_{i=0, 1, \ldots, m-1}$$

These configurations can be realized by R; hence, $(q, (D, t)) \vdash^* (q, \omega)$ is true for R. $\omega \in \psi$ implies that $\omega \in \theta$, and therefore $(D, t) \in A(R)$. We can conclude that $T(G) \subseteq A(R)$.

If we assume $(D, t) \in A(R)$ then $(q, (D, t)) \vdash^{*} (q, \omega)$ in R and $\omega \in \theta$. This implies that a sequence $\{\beta_i\}_{i=0, 1, \ldots, m}$ exists such that $\beta_0 = (D, t)$, $\beta_m = \omega$, and

$$\{(q, \beta_i) \vdash (q, \beta_{i+1})\}_{i=0, 1, \ldots, m-1} .$$

From this sequence and the δ-rules for R we can find sequences

$$\{b_i\}_{i=0, 1, \ldots m-1}, \quad b_i \in D \text{ for all } i,$$

$$\{\gamma_i\}_{i=0, 1, \ldots, m-1}, \quad \gamma_i = \beta_i / b_i \text{ for all } i,$$

$$\{\omega_i\}_{i=1, 2, \ldots, m}, \quad \omega_m = \omega .$$

Again we want $\{\delta(q, \gamma_i) = (q, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}$.

Using the definition for the δ-rules and the last sequence we can construct the tree (D, r) where for all maximal $x \in D$ $r(x) = W_{t(x)}$ and for all nonmaximal $x \in D$, $r(x) = \omega'$ where $\omega' \in \{\omega_1, \omega_2, \ldots, \omega_m\}$. Finally we note that $r(0) = \omega_m \in \theta$; therefore, $r(0) \in \psi$. $r(0) \in \psi$ implies that (D, r) is an accepting G-run for (D, t) insures that $(D, t) \in T(G)$. Hence $A(R) \subseteq T(G)$.

We have $A(R) = T(G)$. Moreover, $\Gamma = T(G)$ for some tree automata G implies that $\Gamma = A(R)$ for some one state alpha tree acceptor R and all of the δ-rules of R are of the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) = 1$.

On the other hand if we assume Γ is recognized by a one state alpha tree acceptor $R = (\{q\}, \Omega, \delta, q, \Sigma, \theta)$ and all the δ-rules are of the form $\delta(q, \beta) = (q, \omega)$ where $d(\beta) = 1$, then we can construct a tree automata $G = (Q, \alpha, W, \psi)$ where

$W = \{W_\sigma : W_\sigma = \sigma \text{ for all } \sigma \in \Sigma\}$,

$Q = \{\tau : R \text{ has a } \delta\text{-rule of the form } \delta(q, \beta) = (q, \tau)\} \cup W$,

$\psi = \theta$, and

α is defined by $\alpha_x(x_1 x_2 \cdots x_n) = \omega$ iff $\delta(q, x(x_1 \cdots x_n)) = (q, \omega)$

Note here that $n \geq 1$ in all cases since all δ-rules of R are of the form $\delta(q, \beta) = (q, \omega)$ where $d(\beta) = 1$. Again we claim $A(R) = T(G)$.

$(D, t) \in A(R)$ implies that $(q, (D, t)) \vdash^{*} (q, \omega)$ in R and $\omega \in \theta$.

Hence we again have the sequence $\{\beta_i\}_{i=0, 1, \ldots, m}$ where $\beta_0 = (D, t)$, $\beta_m = \omega$ and

$$\{(q, \beta_i) \vdash (q, \beta_{i+1})\}_{i=0, 1, \ldots, m-1} .$$

This sequence implies the existence of the sequences $\{b_i\}_i$, $b_i \in D(\beta_i)$, $\{\gamma_i\}_i$, $\gamma_i = \beta_i / b_i$, for $i = 0, 1, \ldots, m-1$, and the sequence $\{\omega_i\}_{i=1, 2, \ldots, m}$, such that $\omega_i \in \Omega$, $\omega_m = \omega$. For these sequences we have that

$\{\delta(q, \gamma_i) = (q, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}$, are all δ-rules for R.

We are now in a position to define an accepting G-run of (D, t). For all maximal $x \in D$ set $r(x) = W_{t(x)}$ and for all nonmaximal $x \in D$, set $r(x) = \omega_{i+1}$ when $\gamma_i = t(x)(u(x_1)u(x_2)\ldots u(x_n))$, u is r or t.

Since all δ-rules are of the form $\delta(q, \beta) = (q, \omega)$, $d(\beta) = 1$ we can define $r(x)$ for all necessary $x \in D$.

$r(0) = \omega_m = \omega \in \theta$; therefore, $r(0) \in \psi$. Hence (D, r) is an accepting G-run for (D, t). Therefore $(D, t) \in T(G)$, implying $A(R) \subseteq T(G)$.

Finally, we assume $(D, t) \in T(G)$. From the accepting G-run (D, r) we can build the sequence $\{\gamma_i\}_{i=0, 1, \ldots, m-1}$ and the sequence $\{\omega_i\}_{i=1, 2, \ldots, m}$. We get these sequences from the definition of α for G. That is, $\alpha_{t(x_i)}(r(S_D(x_i))) = \omega_{i+1}$, and if $S_D(x_i) = (x_{i_1}\ x_{i_2}\ \ldots x_{i_n})$, then $\gamma_i = t(x_i)(u(x_{i_1})u(x_{i_2})\ldots u(x_{i_n}))$ where $u = t$ when x_{i_j} is maximal in D and $u = r$ when x_{i_j} is nonmaximal.

The sequences should be arranged so that $\omega_m = r(0)$. We can construct the usual sequences from $\delta(q, \gamma_i) = (q, \omega_{i+1})_{i=0, 1, \ldots, m-1}$ to get $(q, (D, t)) \vdash^* (q, \omega_m)$ in R. $\omega_m = r(0)$ and (D, r) an accepting G-run insures that $r(0) \in \psi$; therefore, $\omega_m \in \theta$ and $(D, t) \in A(R)$. Hence $T(G) \subseteq A(R)$ and finally we have $T(G) = A(R)$.

QED

Definition 4.8. A nondeterministic tree automaton is a system $G = (Q, \alpha, W, \psi)$ where Q and ψ are as for deterministic tree automata but α and W are defined below.

$\alpha_\sigma : \bigcup_{1 \leq n \leq k} Q^n \rightarrow P(Q)$ for $\sigma \in \Sigma$ and $P(Q)$ is the power set of Q, and

$W_\sigma \subseteq Q$ for $\sigma \in \Sigma$.

Definition 4.9. $(D, r) \in T_{k, Q}$ is a G-run over $(D, t) \in T_{k, \Sigma}$ denoted $r \in Rn(G, t)$, where G is a nondeterministic tree automata if and only if

1) $r(x) \in W_{t(x)}$ for maximal $x \in D$ and

2) $r(x) \in \alpha_{t(x)}$ for nonmaximal $x \in D$.

Definition 4.10. A set Γ of (k, Σ) value trees is nondeterministically TA recognizable if and only if there exists a nondeterministic tree automaton G such that $\Gamma = T(G)$.

Theorem 4.11 (Thatcher). Let Γ be a set of (k, Σ) value trees. Γ is TA recognizable if and only if Γ is nondeterminnstically TA recognizable.

The proof is a direct application of the subset construction used for ordinary automata [11].

Definition 4.12. An alphabet nondeterministic alpha tree

acceptor is a system $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ where Q, Ω, q_0, Σ, and θ are as for deterministic alpha tree acceptors. The δ-rules are of the form:

$$\delta: Q \times Q^n \rightarrow Q \times P(\Omega): (q, \beta) \rightarrow \{(p, \omega_1), (p, \omega_2), \ldots, (p, \omega_n)\}$$

We can use the methods of the proof of Theorem 4.7 to obtain the following result.

Theorem 4.13. Γ, a set of (k, Σ) value trees, is nondeterministically TA recognizable if and only if it can be recognized by some nondeterministic alpha tree acceptor R and R has only one state.

Theorems 4.7, 4.11, and 4.13 yield

Theorem 4.14. Γ, a set of (k, Σ) value trees, is α-recognizable by a one state alpha tree acceptor if and only if Γ is recognizable by an alphabet nondeterministic alpha tree acceptor with one internal state.

Other types of nondeterminism can be defined for alpha tree acceptors. We do this in Section 8. The reason that we delay further discussion of nondeterminism until Section 8 is that the proofs of the more general results involve techniques developed in Section 7 where

matrix grammars are discussed.

In designing alpha tree acceptors we merely considered tree automata and then separated the state of the device from the input. The hope was to build a device with more power than standard tree automata. The next theorem demonstrates that we have been successful.

For notational purposes we let $T_i = \{\Gamma : \Gamma$ is a set of (k, Σ) value trees, for some k and Σ, and there exists an i-state Alpha Tree Acceptor R such that $\Gamma = A(R)\}$. Clearly for $i \geq 1$, $T_i \subseteq T_{i+1}$.

Theorem 4.17. $T_1 \neq T_2$.

Proof: We consider the following set of tree domains $\Lambda = \{\{0\}, \{0, 1, 2\}, \{0, 1, 2, 1\cdot1, 2\cdot1\}, \{0, 1, 2, 1\cdot1, 2\cdot1, 1\cdot1\cdot1, 2\cdot1\cdot1\}, \ldots\}$, and the map $t: D \rightarrow \{a\} : x \rightarrow a$ where $D \in \Lambda$, and $x \in D$.

Let $\Gamma = \{(D, t) : D \in \Lambda\}$. We will build an alpha tree acceptor with two states that recognizes Γ.

Let $R = (Q, \Omega, \delta, q_0, \{a\}, \theta)$ where $Q = \{q_0, q_1\}$, $\Omega = \{a, X, Y, Z\}$, $\theta = \{Y\}$, and δ is defined by the following δ-rules.

$$\delta(q_0, a) = (q_1, X) \qquad \delta(q_1, a) = (q_0, Y)$$

$$\delta(q_0, a(X)) = (q_1, Z) \qquad \delta(q_1, a(Y)) = (q_0, Y)$$

$$\delta(q_0, a(Z)) = (q_1, A) \qquad \delta(q_1, X) = (q_0, Y)$$

$$\delta(q_1, Z(Y)) = (q_0, Y)$$

Here $A(R) = \Gamma$.

We assume that there exists a one state alpha tree acceptor, R, such that $A(R) = \Gamma$. Say $R = (\{q\}, \Omega, \delta, q, \Sigma, \theta)$ where $|\Omega| = n$.

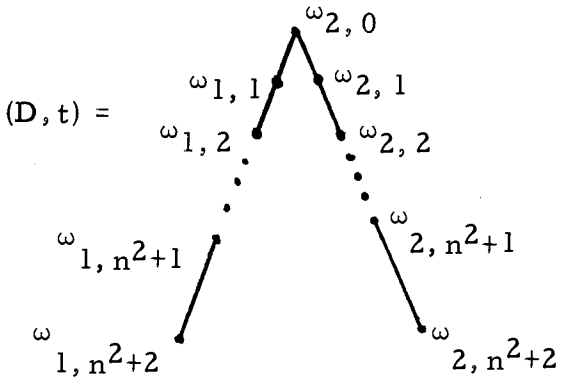
In the following paragraphs we will show that no such one state alpha tree acceptor exists. This proof will be independent of the map t. That is, not only is Γ not α-recognizable by one state alpha tree acceptors but also there is no map t defined on the tree domains of Λ such that the resulting set of trees is α-recognizable by any one state alpha tree acceptor.

The following proof is based on the proof frequently used to demonstrate that $\{0^n 1^n : n \geq 1\}$ is not regular.

Choose $D \in \Lambda$ as shown below.

$$D = \{0, 1, 2, 1\bullet 1, 2\bullet 1, 1\bullet 1\bullet 1, 2\bullet 1\bullet 1, \ldots, 1\bullet 1\bullet 1 \ldots 1, 2\bullet 1\bullet 1 \ldots 1\}.$$

The last two elements of D above have n^2+2 and n^2+1, 1's respectively. Clearly $D \in \Lambda$. $(D, t) \in A(R)$ implies that there exists a map t such that (D, t) is a $(2, \Sigma)$ value tree and R can parse (D, t) to a single element of θ. Say



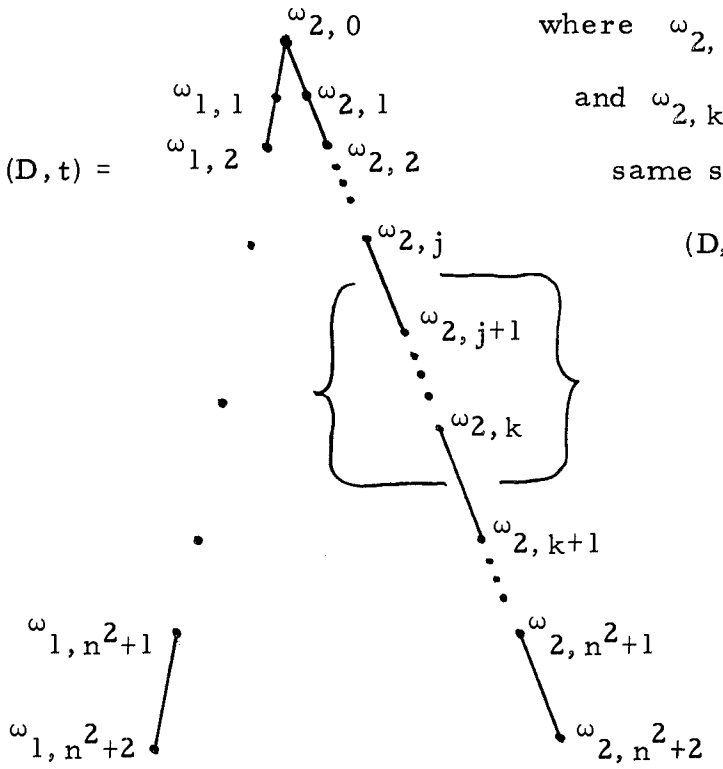
where $\omega_{2,0}, \omega_{1,i}, \omega_{2,i} \in \Sigma$

for $i = 1, 2, \ldots, n^2+2$.

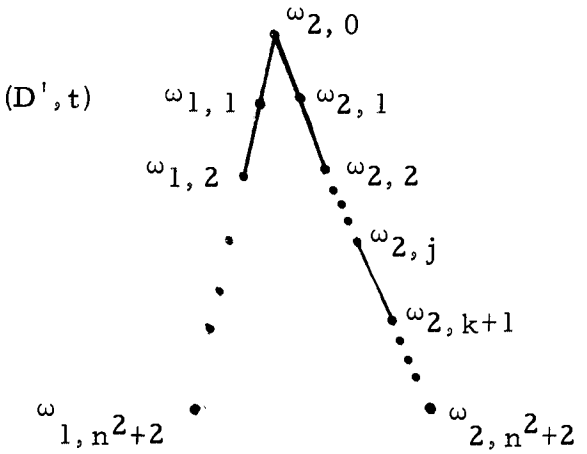
Since R has only one state it must be able to parse (D, t) one branch at a time except possibly for the last three nodes. As R parses the right branch of (D, t) the δ-rules involved must be of the form:

$$\delta(q, \omega_{2,k}) = (q, \gamma_{2,k}) \quad \text{or} \quad \delta(q, \omega_{2,k}(\gamma_{2,k+1})) = (q, \gamma_{2,k})$$

Hence in a most efficient parse R must rewrite n^2+1 $(1, \Omega)$ value trees of depth one. Since $|\Omega| = n$ there are only n^2 distinct $(1, \Omega)$ value trees of depth one. We can conclude that some $(1, \Omega)$ value tree appears twice as R parses the right branch of (D, t). That is, (D, t) must be as depicted below.



where $\omega_{2,j} = \omega_{2,k'}$ and $\omega_{2,j+1}$ and $\omega_{2,k+1}$ are replaced by the same symbol when R parses (D, t). But then we can remove the braced section of (D, t) to obtain (D', t).



(D, t) ϵ A(R) implies that (D', t) ϵ A(R), but D' $\notin \Lambda$. →←

We must conclude that R does not exist. QED

It is conjectured that in general $T_i \neq T_{i+1}$, but attempts to extend the above example have been unsuccessful.

V. DECIDABILITY FOR ALPHA TREE ACCEPTORS

The relation between alpha tree acceptors that have δ-rules of the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) = 0$, relabeling moves, and those that do not is not fully understood. In this chapter we prove a decidability result for both types. In both cases we appeal to a frequently used diagonal argument [6] and [9]. The following discussion will apply to any arbitrary but given alphabet, Σ.

We proceed by noting that we will be using tree expressions to describe (k, Σ) value trees. This maps the set of (k, Σ) value trees into the set of base $|\Sigma| + 2$ numbers. Hence we can refer to the ith (k, Σ) value tree.

We can also use a Gödel-like numbering system to order alpha (beta) tree acceptors; therefore, we can also refer to the ith alpha (beta) tree acceptor.

We now consider the following set.

$L = \{\alpha : \alpha$ is the ith (k, Σ) value tree and $\alpha \notin A(R_i)$ where R_i is the ith alpha (beta) tree acceptor$\}$.

If we limit our alpha (beta) tree acceptors to alpha (beta) tree acceptors that do not have relabeling moves, then L is recursive. Then if we apply the usual diagonal argument to L, we have the following theorem.

Theorem 5.1. There do exist recursive sets of (k, Σ) value trees that are not recognizable by alpha (beta) tree acceptors where all δ-rules are of the form $\delta(q, \beta) = (p, \delta)$ where $d(\beta) \geq 1$.

To remove the restriction on the δ-rules of the alpha (beta) tree acceptors considered, we apply a proof used by Salomaa [9] to prove a similar result for context-sensitive grammars.

Theorem 5.2. Given a (k, Σ) value tree, α, and an alpha [beta] tree acceptor R there is an algorithm to determine whether or not $\alpha \in A(R)$.

Proof: Given $R = (Q, \Omega, \delta, Q_0, \Sigma, \theta)[(Q, \Omega, \delta, Q_0, \Sigma, \Phi)]$ we consider sequences of ordered pairs:

$$(q'_0, \beta_0), (q'_1, \beta_1), \ldots, (q'_n, \beta_n)$$

where $d(\beta_0) = 0$, $\beta_n = \alpha$, $n \geq 0$, and $\beta_i \in T_{k, \Omega}$ and $q'_i \in Q$ for $i = 0, 1, \ldots, n$. We also make the following restrictions on the sequences considered above. If we use tree expressions over $\Omega \cup \{(,)\}$, $(,) \notin \Omega$, to describe each β_i, then we can refer to the length of β_i, denoted $|\beta_i|$, as the number of symbols in the tree expression for β_i. Here we consider sequences such that $|\beta_i| \leq |\beta_{i+1}|$ for $i = 0, 1, \ldots, n-1$, and each of the pairs (q'_i, β_i) are pairwise distinct. These last restrctions insure that the number

of sequences under consideration is finite. Finally, we note $\alpha \in A(R)$ if and only if for at least one sequence we have $(q'_{i+1}, \beta_{i+1}) \vdash^{*} (q_i, \beta_i)$ for $i = 0, 1, \ldots, n-1$, in R and $\beta_0 \in \theta[q'_0 \in \Phi]$. QED

Theorem 10.2 assures that L is recursive even if we allow alpha [beta] tree acceptors with relabeling moves. Hence we have the following theorem.

Theorem 5.3. There do exist sets of (k, Σ) value trees that are not recognizable.

VI. BOOLEAN PROPERTIES

In this section we give a constructive proof that shows that the union of two recognizable sets is recognizable. We have been unable to obtain similar results for intersection and complement.

Theorem 6.1. If Γ and ψ are recognizable sets of (k, Σ) value trees then $\Gamma \cup \psi$ is a recognizable set.

Proof: Since Γ and ψ are recognizable sets there exists alpha tree acceptors R_1 and R_2 such that $\Gamma = A(R_1)$ and $\psi = A(R_2)$.

If $R_1 = (Q_1, \Omega_1, \delta_1, q_{1_0}, \Sigma, \theta_1)$ and

$R_2 = (Q_2, \Omega_2, \delta_2, q_{2_0}, \Sigma, \theta_2)$, then construct

$$R = (Q_1 \times Q_2, \Omega'', \delta, (q_{1_0}, q_{2_0}), \Sigma, \theta)$$

where

$$\Omega'' = (\Omega'_1 \times \Omega'_2) \cup \Sigma, \quad \Omega'_1 = \Omega_1 \cup \{*\}$$

$$\Omega'_2 = \Omega_2 \cup \{*\}, \quad \text{and} \quad * \notin \Omega_1 \cup \Omega_2, \quad \text{and}$$

$$\theta = (\Omega'_1 \times \theta_2) \cup (\theta_1 \times \Omega'_2) .$$

The δ-rules of R are defined by $\delta((q, q'), \beta) = ((p, p'), (\omega, \omega'))$, where (q, q') and (p, p') are elements of $Q_1 \times Q_2$, (ω, ω') is an element of $\Omega'_1 \times \Omega'_2$, and β is a (k, Ω'') value tree. $((p, p'), (\omega, \omega'))$ is defined by making the following considerations.

First, associated with β is the (k, Ω_1) value tree β_1. If $\beta = (D, t)$, then $\beta_1 = (D, t_1)$ where if $b \in D(\beta)$ and $t(b) = (\tau', \tau'')$ is an element of $(\Omega'_1 \times \Omega'_2)$, then $t_1(b) = \tau'$; but if $t(b) = \sigma \in \Sigma$, then $t_1(b) = \sigma$. Then if R_1 has a δ-rule of the form $\delta_1(q, \beta_1) = (q', \eta)$, then $p = q'$ and $\omega = \eta$. If R_1 has no such rule, then $p = q_{1_0}$ and $\omega = *$. We define p' and ω' similarly. Associated with β is the (k, Ω_2) value tree β_2. If $\beta = (D, t)$, than $\beta_2 = (D, t_2)$ where if $b \in D(\beta)$ and $t(b) = (\tau', \tau'') \in (\Omega'_1 \times \Omega'_2)$, then $t_2(b) = \tau''$; but if $t(b) = \sigma \in \Sigma$, then $t_2(b) = \sigma$. Then if R_2 has a δ_2-rule of the form $\delta_2(q, \beta_2) = (q'', \eta')$, then $p' = q''$ and $\omega' = \eta'$. If R_2 has no such rule then $p' = q_{2_0}$ and $\omega' = *$.

Such δ-rules are created for R for all possible trees β_1 and β_2 and respective elements from Q_1 and Q_2. Heuristically, we see that R simulates R_1 and R_2 running in parallel. When either R_1 or R_2 is unable to continue, then R writes a $*$ in the respective component of the output symbol.

The definition of recognition insures that the (k, Σ) value tree α is an element of $A(R)$ if and only if

$$((q_{1_0}, q_{2_0}), \alpha) \vdash^{*} ((p, p'), (\omega, \omega')) \quad \text{where} \quad (\omega, \omega') \in \theta.$$

Hence $\alpha \in A(R)$ if and only if

$$(q_{1_0}, a) \vdash^{*} (p, \omega) \quad \text{in} \quad R_1 \quad \text{and} \quad \omega \in \theta_1,$$

and/or

$$(q_{2_0}, a) \vdash^{*} (p', \omega') \quad \text{in} \quad R_2 \quad \text{and} \quad \omega' \in \theta_2.$$

Hence, $a \in A(R)$ if and only if $a \in A(R_1)$ and/or $a \in A(R_2)$.

That is, $A(R) = A(R_1) \cup A(R_2) = \Omega \cup \psi$. QED

VII. ALPHA TREE ACCEPTORS AND MATRIX GRAMMARS

In this section we will define the language of an alpha tree acceptor, matrix grammars, and the notion of near structural equivalence. We will show that for every alpha tree acceptor there exists a nearly structurally equivalent matrix grammar; and conversely, for every matrix grammar there exists a nearly structurally equivalent alpha tree acceptor.

Definition 7.1. Let R be an alpha [beta] tree acceptor. The language of R is the set $L(R) = \{w:w \in \Sigma^* \text{ and there exists a } (k, \Sigma) \text{ value tree } \alpha \text{ such that } \alpha \in A(R)[B(R)] \text{ and } w = fr(\alpha)\}$.

Definition 7.2. A matrix grammar is an ordered quadruple $G(V_n, V_t, X, M)$ where V_n is a set of nonterminals, V_t is a set of terminals, $X \in V_n$, and M is a finite set of finite nonempty sequences whose elements are ordered pairs (P, Q) where $P \in V_n$ and $Q \in (V_n \cup V_t)^+$. The symbol X is sometimes referred to as the axiom of the matrix grammar.

We have actually only defined λ-free type 2 matrix grammars. Other matrix grammars are discussed in Salomaa [8]. Salomaa points out that

> . . . the generative capacity of type i matrix grammars is the same as that of type i grammars for $i \neq 2$, whereas the generative capacity of context free [type 2] matrix grammars is remarkably larger than that of context-free grammars. [1]

Furthermore, as we shall see, the generative power of type 2 matrix grammars is nearly the same as the recognition power of alpha [beta] tree acceptors. For this reason, we will use the term matrix grammar to mean λ-free type 2 matrix grammar.

The pairs (P, Q) are referred to as the productions and written $P \rightarrow Q$. The sequences of $M = \{m_1, m_2, \ldots, m_t\}$ are referred to as matrices of productions and written

$$m_i = [P_{i_1} \rightarrow Q_{i_1}, P_{i_2} \rightarrow Q_{i_2}, \ldots, P_{i_{r_i}} \rightarrow Q_{i_{r_i}}], \quad r_i \geq 1.$$

Each of the productions in the sequence is a context-free production.

Let $V = V_n \cup V_t$ and let $W(V) = V^+$ denote the nonempty words over V. For a matrix grammar G we define a binary relation $\Rightarrow_G$, or merely $\Rightarrow$ when the grammar is understood, on the set $W(V)$ as follows. For any strings $u, v \in W(V)$, $u \Rightarrow_G v$ holds if and only if there exists a matrix of M, say $m_i = \{P_{i_j} \rightarrow Q_{i_j}\}_{j=1, 2, \ldots, r_i}$, and sequences $\{R_j\}$ and $\{R'_j\}$,

---

[1] Arto Salomaa, Formal Languages, Academic Press, New York, 1973, pp 143-144.

$j = 1, 2, \ldots, r_i+1$, $R_j, R'_j \in W(V)$ for all j. We must be able to find these elements of $W(V)$ such that $u = R_1 P_{i_1} R'_1$, $v = R_{r_i+1} Q_i R'_{r_i+1}$, and for $j = 1, 2, \ldots, r_i-1$, Q_{i_j} is a substring of $R_{j+1} P_{i_{j+1}} R'_{j+1}$. Let $\Rightarrow^*_G$ or $\Rightarrow^*$ be the reflexive transitive closure of $\Rightarrow_G$.

Definition 7.3. Let G be a matrix grammar. The language of G is the set $L(G) = \{u \in V_t^+ : X \Rightarrow^*_G u\}$.

Definition 7.4. Two matrix grammars G_1 and G_2 are equivalent if and only if $L(G_1) = L(G_2)$.

Definition 7.5. Two matrix grammars G_1 and G_2 are structurally equivalent if and only if they are equivalent and they assign similar derivation trees, differing only in the labeling of the nodes, to each word of the language.

Definition 7.6. An elementary subdivision of a tree α is a tree α' that is obtained from α by replacing each of zero or more nodes of α with two nodes and a connecting branch. Two matrix grammars, G and G', are nearly structurally equivalent if $L(G) = L(G')$ and a tree, α, is a derivation tree of G if and only if there exists a derivation tree, α', of G' and a tree, γ, such that α and α' are both elementary subdivisions of γ.

Example 7.7. Let $G_1 = (V_n, V_t, X, M)$ where

$V_n = \{X, A, B, C\}$, $V_t = \{a, b, c\}$, and $M = \{m_1, m_2, m_3\}$, where

$$m_1 = [X \rightarrow ABC], \quad m_2 = [A \rightarrow aA, B \rightarrow bB, C \rightarrow cC],$$

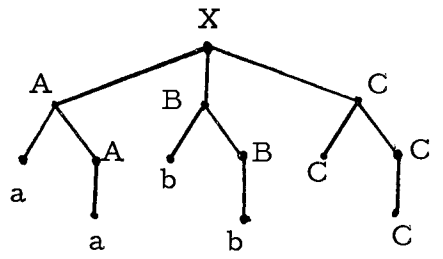
and

$$m_3 = [A \rightarrow a, B \rightarrow b, C \rightarrow c].$$

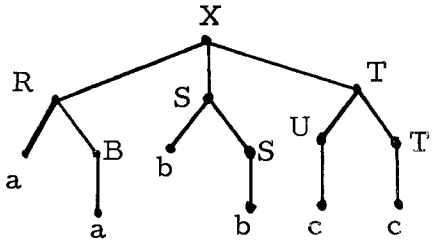
Let $G_2 = (V_n', V_t, X, M')$ where $V_n' = \{X, R, S, T, U\}$ and $M = \{m_1', m_2', m_3', m_4'\}$ where

$$m_1' = [X \rightarrow RST], \quad m_2' = [R \rightarrow aR, s \rightarrow bS, T \rightarrow UT],$$

$$m_3' = [R \rightarrow a, S \rightarrow b, T \rightarrow c], \quad \text{and} \quad m_4' = [U \rightarrow c].$$



Derivation tree for aabbcc in G_1



Derivation tree for aabbcc in G_2

Figure 7.1.

G_1 and G_2 are nearly structurally equivalent, but they are not structurally equivalent. We note further that

$L(G_1) = L(G_2) = \{a^n b^n c^n : n \geq 1\}$. This is not a context-free language, but all the involved productions are context-free.

The above notions of equivalence and structural equivalence are taken from Salomaa [9]. The next three definitions extend these concepts to alpha [beta] tree acceptors. Near structural equivalence will be used to relate matrix grammars and alpha [beta] tree acceptors.

Definition 7.8. An alpha [beta] tree acceptor R and a matrix grammar G are equivalent if and only if L(R) = L(G).

Definition 7.9. An alpha [beta] tree acceptor R and a matrix grammar G are structurally equivalent if and only if L(R) = L(G) and α is a derivation tree of G if and only if there exists a tree $\beta \in A(R)[B(R)]$ such that α and β are equal up to a relabeling of nodes.

Definition 7.10. An alpha [beta] tree acceptor, R, and a matrix grammar, G, are nearly structurally equivalent if and only if L(R) = L(G) and a tree α is a derivation tree of G if and only if there is a tree $\beta \in A(R)$ and a tree γ such that α and β are both elementary subdivisions of γ.

Our next step is to show that for every matrix grammar there is a nearly structurally equivalent alpha tree acceptor. The next two lemmas insure that we can always put the matrix grammar into a desirable form.

Lemma 7.11. If $G = (V_n, V_t, X, M)$ is a matrix grammar, then there exists a matrix grammar $G' = (V_n', V_t, X', M')$ such that X' appears in only one matrix, say m_0, of M', the number of productions in m_0 is one, X' does not appear on the right hand side of any production, and G and G' are nearly structurally equivalent.

Proof: Let $V_n' = V_n \cup \{X'\}$, $X' \notin X_n$, and $M' = M \cup [X' \rightarrow X]$.

QED

Definition 7.12. We say that a matrix grammar, $G = (V_n, V_t, X, M)$, is uniquely invertible if and only if m_i, $m_j \in M$, $i \neq j$, the last production of m_i is $P \rightarrow Q$, and the last production of m_j is $P' \rightarrow Q'$ insures that $P \neq P'$ and/or $Q \neq Q'$. That is, each matrix of M is identified by its last production.

Lemma 7.13. Let $G = (V_n, V_t, X, M)$ be a matrix grammar. There exists a matrix grammar, $G' = (V_n', V_t, X, M')$, such that G' is uniquely invertible and G and G' are nearly structurally equivalent.

Proof: Let $M = \{m_1, m_2, \ldots, m_t\}$ and define $M' = \{m_1', m_2', \ldots, m_t'\}$ where if

$$m_i = [P_1^i \rightarrow Q_1^i, P_2^i \rightarrow Q_2^i, \ldots, P_{r_i}^i \rightarrow Q_{r_i}^i],$$

then

$$m_i' = [P_1^i \rightarrow Q_1^i, P_2^i \rightarrow Q_2^i, \ldots, P_{r_i}^i \rightarrow R^i, R^i \rightarrow Q_{r_i}^i] .$$

For each i the R^i is a new nonterminal. QED

Applying first Lemma 7.13 and then 7.11 we obtain the following result.

Lemma 7.14. If $G = (V_n, V_t, X, M)$ is a matrix grammar, then there exists a matrix grammar, $G' = (V_n', V_t, X', M')$ such that X' appears in only one matrix of M' and this matrix is of the form $[X' \rightarrow X]$, G' is uniquely invertible, and G and G' are nearly structurally equivalent.

Theorem 7.15. If G is a matrix grammar then there exists an alpha tree acceptor, R, such that all the δ-rules of R are of the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) \leq 1$ and G and R are nearly structurally equivalent.

Proof: Let $G = (V_n, V_t, X, M)$ be a matrix grammar and construct the matrix grammar $G' = (V_n', V_t, X', M')$ such that G and G' are nearly structurally equivalent and G' satisfies the conditions of Lemma 7.14. Then we construct the alpha tree acceptor $R = (Q, V_n' \cup V_t, \delta, q_0, V_n' \cup V_t, X')$ where δ and Q are defined below. First we introduce some notation. For $m_i \in M'$, say

$$m_i = [P^i_1 \to Q^i_1, P^i_2 \to Q^i_2, \ldots, P^i_{r_i} \to Q^i_{r_i}],$$

where

$$Q^i_j = a^i_{j_1} a^i_{j_2} \ldots a^i_{j_{s^i_j}}, \quad a^i_{j_k} \in V'_n \cup V_t \text{ for } k = 1, 2, \ldots, s^i_j,$$

we define a sequence of trees by the following:

$$\beta^i_j = P^i_j(a^i_{j_1} a^i_{j_2} \ldots a^i_{j_{s^i_j}}) = P^i_j(Q^i_j) \quad \text{for } j = 1, 2, \ldots, r^i.$$

Note that all the β^i_j are $(k, V'_n \cup V_t)$ value trees.

To define the δ-rules for R we start with q_0 and for all matrices of M' we add a δ-rule of the following form:

$$\delta(q_0, \beta^i_{r^i}) = (q^i_{(r^i-1)}, P^i_{r^i}).$$

Hence if α is a derivation tree of G', then as R is attempting to parse α, q_0 is merely a state of R that looks for a subtree of depth one that could be produced by the last production of some matrix of M'. If such a subtree is found, it is reduced to the symbol on the left hand side of the respective production and control of R is turned over to a state, $q^i_{(r^i-1)}$, which will initiate the reverse of the next to last production of the appropriate matrix of M'. It is at this point that we are using the fact that G' is invertible. In particular, the moves from q_0 are deterministic.

We now define the remaining δ-rules and add the necessary states to Q. Basically we add a sequence of states $\{q^i_j\}_{j=1, 2, \ldots, r^i-1}$ for each $i = 1, 2, \ldots, t$. The sequence $\{q^i_j\}_{j=1}^{r^i-1}$ mimics the reverse of the productions $P^i_1 \rightarrow Q^i_1, P^i_2 \rightarrow Q^i_2, \ldots, P^i_{(r^i-1)} \rightarrow Q^i_{(r^i-1)}$ of m_i. That is, for all $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, r^i-1$ we set

$$\delta(q^i_j, \beta^i_j) = (q^i_{j-1}, P^i_j)$$

where the β^i_j are as defined above. The q^i_j are all new states for $j \neq 0$, and when $j = 0$, $q^i_j = q_0$. Hence control is returned to q_0 when R has completed imitating the reverse of an entire matrix of M'.

In case $r^i = 1$, that is in the case that $m_i = [P^i \rightarrow Q^i]$, we will have $\delta(q_0, P^i(Q^i)) = (q_0, P^i)$. In particular, we will have the δ-rule $\delta(q_0, X'(X)) = (q_0, X')$.

We claim that G' and R are structurally equivalent.

Assume that $s \in L(G')$. Then there exists at least one sequence $\{m_i\}$, $i = 1, 2, \ldots, v$, $m_i \in M'$ and $X' \Rightarrow^*_{G'} s$ by $X' \underset{m_1}{\Rightarrow} \ldots \underset{m_v}{\Rightarrow} s$. Associated with this derivation is the $(k, V'_n \cup V_t)$ value tree α where the leaves of α spell s.

We can divide the sequence $\{m_i\}_{i=1, 2, \ldots, v}$ into a sequence

of individual productions, say $\{P_j \rightarrow Q_j\}_{j=1, 2, \ldots, u}$, where $u = \sum_{i=1}^{v} r^i$ and r^i is the number of productions in m_i. Using the sequence $\{P_j \rightarrow Q_j\}_{j=1, 2, \ldots, u}$, we can construct the sequences

$\{q'_i\}_{i=0, 1, \ldots, u}$, $q'_i \in Q$ for all i,

$\{\beta_i\}_{i=0, 1, \ldots, u-1}$, $\beta_i = P_{u-i}(Q_{u-i})$ are all $(k, V'_n \cup V_t)$ value trees, and

$\{\omega_i\}_{i=1, 2, \ldots, u}$, $\omega_i = P_{u-i+1}$, $\omega_i \in V'_n \cup V_t$

The sequence $\{q'_i\}_i$ is obtained from $\{P_j \rightarrow Q_j\}_{j=1, 2, \ldots, u}$ by $q'_i = q_{j_k}$ when $P_{u-i} \rightarrow Q_{u-i}$ is $P_{j_k} \rightarrow Q_{j_k}$ from m_j unless $k = r_j$ and then $q'_i = q_0$. This is true for $i = 1, 2, \ldots, u-1$. When $i = u$ or $i = 0$, then $q'_i = q_0$.

Thus using the sequence $\{\delta(q'_i, \beta_i) = (q'_{i+1}, \beta_{i+1})\}_i$ for $i = 0, 1, \ldots, u-1$, we have $(q_0, \alpha) \vdash^* (q_0, \beta_u)$ in R. $\beta_u = P_1$ and $P_1 = X'$. Therefore $\alpha \in A(R)$ and $L(G') \subseteq L(R)$ and derivation trees of G' are elements of $A(R)$.

To show that $L(R) \subseteq L(G')$ and $\alpha \in A(R)$ implies that α is a derivation tree for G' we assume that $s \in L(R)$ and α is a $(k, V'_n \cup V_t)$ value tree such that the leaves of α spell s and $(q_0, \alpha) \vdash^* (q_0, X')$ in R. Associated with this parse are the sequences

$\{q'_i\}_{i=0, 1, \ldots, u}$, $q'_i \in Q$, $q'_0 = q'_u = q_0$,

$\{\beta_i\}_{i=0, 1, \ldots, u-1}$, β_i are all $(k, V'_n \cup V_t)$ value trees,

in particular we can write

$$\beta_i = P_{u-i}(Q_{u-i}),$$

$\{\omega_i\}_{i=1, 2, \ldots, u}$, $\omega_i = P_{u-i+1}$.

As a result of the definition of the δ-rules for R we can divide the sequence $\{q'_k\}_k$ into a sequence of subsequences $\{\{q'_{i_j}\}_{j=1, 2, \ldots, r_i}\}_{i=1, 2, \ldots, v}$ where each subsequence $\{q'_{i_j}\}_{j=1, 2, \ldots, r_i}$ corresponds to a matrix m_i of M'. Note that here we are using Lemma 7.13 in that we are assuming that X' does not appear in the center of some matrix m_i. The sequence $\{\{q'_{i_j}\}_{j=1, 2, \ldots, r_i}\}_i$ indicates a sequence $\{m_i\}_{i=1, 2, \ldots, v}$ from M'. Using this sequence we have that $X' \underset{m_1}{\Rightarrow} \ldots \underset{m_v}{\Rightarrow} s$ and the resulting derivation tree is α. Hence $s \in L(G')$ and α is a derivation tree of G'. That is, $\alpha \in A(R)$ and $fr(\alpha) = s$ insures $s \in L(G')$ and α is a derivation tree of G'.

Combining this result with our previous result we have that G' and R are structurally equivalent. Since G' and G are nearly structurally equivalent, G and R are nearly structurally equivalent. We note further that the nature of the construction of R insures that all the δ-rules are of the form $\delta(q, \beta) = (p, \omega)$ where

$d(\beta) \geq 1$ as required. QED

In the following paragraphs we demonstrate that for alpha tree acceptors satisfying the depth requirements we can construct a nearly structurally equivalent matrix grammar. We note that the consequences of allowing δ-rules where $d(\beta) = 0$ are not fully understood.

In the next theorem we show that given an alpha tree acceptor with no relabeling moves we can find a nearly structurally equivalent matrix grammar. The constructive proof is motivated by viewing an alpha tree acceptor as a sequential machine that moves from one state to the next as a function of the present state and the subtree reduced. We draw a state diagram of the sequential machine. We use this state diagram to define matrices of a matrix grammar. Each matrix is designed to simulate the reverse of a single transition of the sequential machine. Special symbols are used to allow the matrix to remember what state it should be simulating.

Before we state the main theorem we offer the following corollary which shows we only need consider alpha tree acceptors with δ-rules of depth one.

Corollary 7.16. Let $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ be an alpha tree acceptor such that all δ-rules of R are of the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) \geq 1$. Then there exists an alpha tree acceptor $R' = (Q', \Omega', \delta', q_0', \Sigma, \theta)$ such that all δ-rules of R' are of the form

$\delta'(q, \beta) = (p, \omega)$ where $d(\beta) = 1$ and $A(R) = A(R')$.

Proof: The construction mentioned in the proof of Lemma 2.26 will suffice since it introduces no relabeling moves. That is, if R has no δ-rules with $d(\beta) = 0$, then if R' is defined by the mentioned construction, then R' will have no δ-rules with $d(\beta) = 0$. Since all δ-rules of R' are such that $d(\beta) \leq 1$, R' is as desired.

QED

To facilitate the reading of the next proof we present an example alpha tree acceptor and its state diagram.

Example 7.17. Let $R = (\{q_0, q_1\}, \{A, B, a\}, \delta, q_0, \{a\}, \{B\})$ be an alpha tree acceptor where the δ-rules are defined by the following.

$$\delta(q_0, a(a)) = (q_0, a) \qquad \delta(q_0, a(aaa)) = (q_1, A)$$

$$\delta(q_1, a(A)) = (q_1, B) \qquad \delta(q_1, a(B)) = (q_1, B)$$

$A(R)$ is the set of all $(3, a)$ value trees such that exactly one node has three descendents, all other nodes have one descendent, and at least one node is above the node having three descendents. The state diagram for R is below. We are using tree expressions in the diagram. To aid in the understanding of the diagram we remark that the symbol $a(aaa)/A$ associated with a particular arrow indicates that when R makes the state transition indicated by the arrow it must

reduce a subtree of the tree being parsed, in particular a(aaa), to an A.

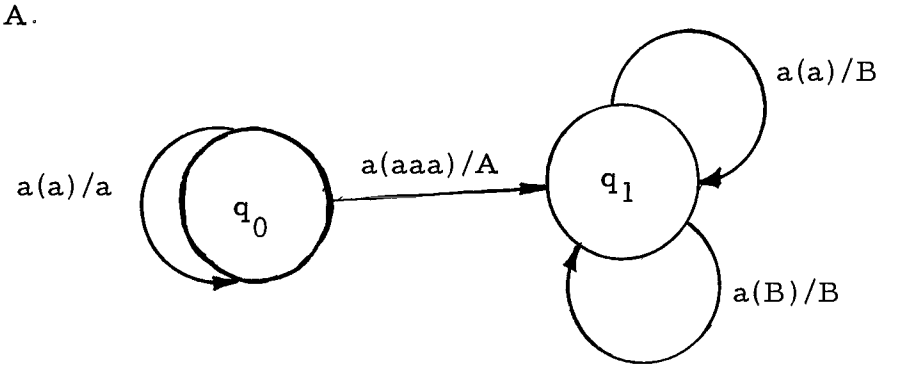


Figure 7.2.

Theorem 7.18. If $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ is an alpha tree acceptor with no relabeling moves, then there exists a matrix grammar, G, such that G and R are nearly structurally equivalent.

Proof: From Corollary 7.16 we can assume that all the δ-rules of R are of depth one. We let $G = (V_n, V_t, X, M)$ where $V_t = \Sigma$, and $V_n = \{A_\omega : \omega \in \Omega\} \cup \{(q, \omega) : q \in Q \text{ and } \omega \in Q\}$.

To define the matrices of M we first consider the cases where R accepts trees that are a single node, that is, (k, Σ) value trees of the form σ and $\sigma \in \theta$. For all such elements $\sigma \in \Sigma$ add the matrix $[X \rightarrow \sigma]$ to M.

To determine the other matrices of M first draw a state diagram for R' as in Figure 7.3.

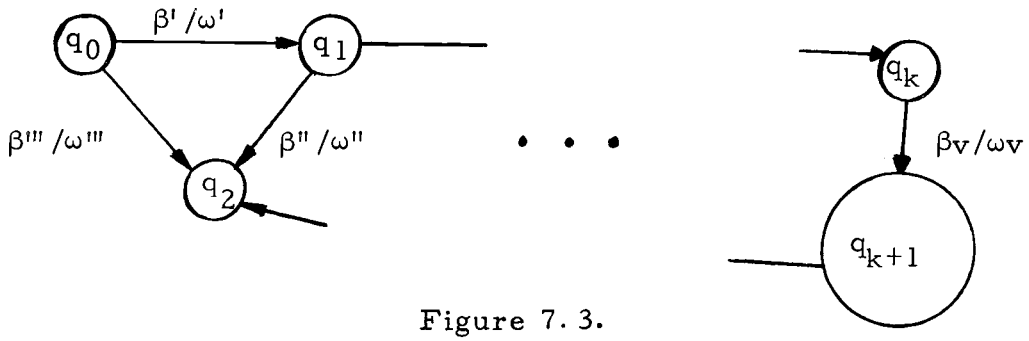
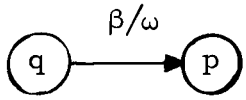


Figure 7.3.

Figure 7.3 is interpreted as $\delta(q, \beta) = (p, \omega)$ in R' if and only if



From the state diagram of R we should locate all states, p, with at least one entering arrow of the form $q \overset{\beta/\sigma}{\longrightarrow} p$ where $\sigma \in \theta$. For all such states and for each entering arrow if $\beta = a_0(a_1\, a_2 \cdots a_n)$ where $a_i \in \Omega$ for $i = 0, 1, \ldots, n$, then we should add the matrix

(7.1.1) $$[X \rightarrow (q, a_1)A_{a_1} A_{a_2} \cdots A_{a_n}] \text{ to } M.$$

We must also add a matrix for each section of the state diagram as depicted by Figure 7.4.

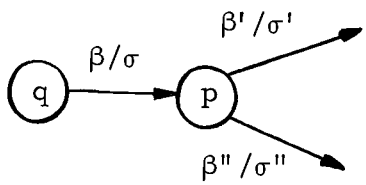


Figure 7.4.

Actually in this case we need to add two matrices to M. The number of matrices that we need to add is equal to the number of egressing arrows attached to state p. If

$$\beta = a_0(a_1\, a_2 \ldots a_n) \qquad a_i \in \Omega,\ i = 0, 1, \ldots, n,$$

$$\beta' = a'_0(a'_1\, a'_2 \ldots a'_{n'}) \qquad a'_i \in \Omega,\ i = 0, 1, \ldots, n',$$

and

$$\beta'' = a''_0(a''_1\, a''_2 \ldots a''_{n''}) \qquad a''_i \in \Omega,\ i = 0, 1, \ldots, n'',$$

then we need to add the following two matrices to M.

(7.1.2)
$$\begin{bmatrix} (p, a'_1) \rightarrow A_{a'_1} \\ A_\sigma \rightarrow (q, a_1)A_{a_2} A_{a_3} \ldots A_{a_n} \end{bmatrix}$$

and

$$\begin{bmatrix} (p, a''_1) \rightarrow A_{a''_1} \\ A_\sigma \rightarrow (q, a_1)A_{a_2} A_{a_3} \ldots A_{a_n} \end{bmatrix}$$

In all cases depicted by Figure 7.4 where $q = q_0$ of R' we should also add matrices of the form:

(7.1.3)
$$\begin{bmatrix} (p, a'_1) \rightarrow A_{a'_1} \\ A_\sigma \rightarrow A_{a_1} A_{a_2} \ldots A_{a_n} \end{bmatrix}$$

and

$$\left[\begin{array}{l} (p, a_1'') \rightarrow A_{a_1''} \\ \\ A_\sigma \rightarrow A_{a_1} A_{a_2} \ldots A_{a_n} \end{array}\right].$$

Also for matrices of the form: $[X \rightarrow (q, a_1)A_{a_2} A_{a_3} \ldots A_{a_n}]$ if $q = q_0$ of R, then we should add the matrix $[X \rightarrow A_{a_1} A_{a_2} \ldots A_{a_n}]$ to M. Finally, we add matrices of type (7.1.4) for all $\omega \in \Sigma$.

(7.1.4) $$[A_\omega \rightarrow \omega]$$

Application of one matrix of type (7.1.1) allows the matrix grammar to initiate a derivation. Matrices of type (7.1.2) allow the matrix grammar to simulate the reverse of what is done by R as it perambulates through its states. G uses symbols of the form (q, a) where $q \in Q$ and $a \in \Omega$, to remember what state should be simulated next. Matrices of the type (7.1.3) allow G to halt the simulation, and finally matrices of type (7.1.4) allow G to place terminals on the frontier of the derivation tree. In the following paragraphs we formalize this discussion.

We claim that R and G are nearly structurally equivalent.

$s \in L(R)$ implies that there exists a tree α such that $fr(\alpha) = s$ and $\alpha \in A(R)$. That is, $(q_0, \alpha) \vdash^* (q, \omega)$ in R and $\omega \in \theta$. As before this guarantees the existence of a sequence of states $\{q_i'\}$, a sequence of trees $\{\beta_i\}_i$, and a sequence of symbols $\{\omega_i\}$

such that as R parses α we can write

$$\{(q'_i, \beta_i) \vdash (q'_{i+1}, \beta_{i+1})\}_{i=0, 1, \ldots, m-1}$$

where $q'_0 = q_0$ of R, $\beta_0 = \alpha$, and $\omega_m = \omega$. This implies that we can form the sequence of subtrees $\{\gamma_i\}$ and the sequence $\{b_i\}$ such that

$$\{\delta'(q'_i, \gamma_i) = (q'_{i+1}, \omega_{i+1})\}_{i=0, 1, \ldots, m-1}$$

where $\gamma_i = \beta_i / b_i$ and $\beta_{i+1} = \beta_i(b_i \leftarrow \omega_{i+1})$.

Since $\omega_m = \omega \in \theta$, $m'_0 = [X \rightarrow (q_{m-1}, a_1)A_{a_2} A_{a_3} \ldots A_{a_n}] \in M$ where $\beta_{m-1} = a_0(a_1 a_2 \ldots a_n)$.

In case $m = 1$ we only need the matrix

$m'_0 = [X \rightarrow A_{a_1} A_{a_2} \ldots A_{a_n}]$ where $\beta_0 = a_0(a_1 a_2 \ldots a_n)$.

Also for $j = 1, 2, \ldots, m-2$ we let

$$m'_j = \begin{bmatrix} (q'_{m-j}, a_{(m-j)_1} \rightarrow A_{a_{(m-j)_1}} \\ A_{\omega_{(m-j)}} \rightarrow (q'_{m-j-1}, a_{(m-j-1)_1})A_{a_{(m-j-1)_2}} \ldots A_{a_{(m-j-1)_{n'}}} \end{bmatrix}$$

where $\delta(q'_{m-j}, \gamma_{m-j}) = (q'_{m-j+1}, \omega_{m-j+1})$,

where $\gamma_{m-j} = a_{(m-j)_0}(a_{(m-j)_1} \ldots a_{(m-j)_n})$,

and $\delta(q'_{m-j-1}, \gamma_{m-j-1}) = (q'_{m-j}, \omega_{m-j})$,

where $\gamma_{m-j-1} = a_{(m-j-1)_0}(a_{(m-j-1)_1} \cdots a_{(m-j-1)_{n'}})$.

The matrix m'_j merely mimics the reverse of what R' does when R' moves from q'_{m-j-1} to q'_{m-j}. For the m-1'st matrix in our sequence we choose

$$m'_{m-1} = \begin{bmatrix} (q'_1, a_{1_1}) \rightarrow A_{a_{1_1}} \\ A_{\omega_1} \rightarrow A_{a_{0_1}} A_{a_{0_2}} \cdots A_{a_{0_{n''}}} \end{bmatrix}$$

where $\delta(q'_1, \gamma_1) = (q'_2, \omega_2)$, $\delta(q'_0, \gamma_0) = (q'_1, \omega_1)$, $\gamma_1 = a_{1_0}(a_{1_1} \cdots a_{1_{n_1}})$, and $\gamma_0 = a_{0_0}(a_{0_1} a_{0_2} \cdots a_{0_{n''}})$.

If α has k leaves, that is if the length of s is k, say $s = \sigma_1\sigma_2 \cdots \sigma_k$ where $\sigma_i \in \Sigma$ for $i = 1, 2, \ldots, k$, then

$$m'_{m-1+i} = [A_{\sigma_i} \rightarrow \sigma_i] \quad \text{for} \quad i = 1, 2, \ldots, k.$$

From the definition of the elements of M we know $m'_i \in M$ for $i = 0, 1, \ldots, m+k-1$. Also $x \underset{m'_0}{\Longrightarrow} \cdots \underset{m'_{m+k-1}}{\Longrightarrow} s$ in G', and the derivation tree corresponding to this derivation is nearly structurally equivalent to α. Hence, $s \in L(R)$ and $\alpha \in A(R)$ such that $fr(\alpha) = s$ implies that $s \in L(G)$ and there is a derivation tree of s in G that is nearly structurally equivalent to α.

Finally, we need to assume $s \in L(G)$ and α is the derivation tree of s in G corresponding to the sequence $\{m_i\}_{i=1, 2, \ldots, v}$ of elements from M. If the length of $s = k$ we may assume without loss of generality that the last k matrices of $\{m_i\}$ are of the form $[A_\sigma \rightarrow \sigma]$. Hence to generate a parse of a tree α', nearly structurally equivalent to α, in R' we need only consider the sequence $\{m_i\}_{i=1, 2, \ldots, v-k}$. It must also be the case that

$$m_{v-k} = \begin{bmatrix} (q, a) \rightarrow A_a & \\ A_\omega \rightarrow A_{a_1} A_{a_2} \ldots A_{a_n} & \end{bmatrix}$$

which means that there must also be a matrix of M, say m'_{v-k}, such that

$$m'_{v-k} = \begin{bmatrix} (q, a) \rightarrow A_a & \\ A_\omega \rightarrow (q_0, a_1)a_2\ a_3 \ldots a_n) & \end{bmatrix}$$

Note that our purpose here is to construct a sequence of δ-rules that can be used to parse a tree nearly structurally equivalent to α. The matrix m'_{v-k} tells us what the first reduction should be and insures that we can have R start in q_0. To obtain the other δ-rules we note that to derive s we must have used matrices m_i for

$i = 1, 2, \ldots, v-k-1$ where these matrices were of the form

$$m_i = \begin{bmatrix} (q'_{v-k-i-1}, a_{(v-k-i-1)_1} \rightarrow A_{a_{(v-k-i-1)_1}} & & \\ & A_{\omega_{v-k-i}} \rightarrow (q_{v-k-i}, a_{v-k-i})A_{a_{(v-k-i)_2}} & \\ & & \ldots A_{a_{(v-k-i)_{n_{(v-k-i)}}}} \end{bmatrix}$$

Finally we have also that

$$m_0 = [X \rightarrow (q_{(v-k}, a_{(v-k)_1})A_{a_{(v-k)_2}} \ldots A_{a_{(v-k)_{n_0}}}];$$

or in the special case that $v = k$, then we have

$m_0 = [X \rightarrow A_{a_1} A_{a_2} \ldots A_{a_k}]$, where $s = a_1 a_2 \ldots a_k$.

Thus the sequence $\{m_i\}_{i=0, 1, \ldots, v-k}$ defines the sequence

$$\{\delta(q'_j, \gamma_j) = (q'_{j+1}, \gamma_{j+1})\}_{j=0, 1, \ldots, v-k-2}$$

where $\gamma_j = a_{j_0}(a_{j_1} a_{j_2} \ldots a_{j_{n_j}})$. And since,

$$m_0 = [X \rightarrow (q'_{v-k}, a_{(v-k)_1} A_{a_{(v-k)_2}} \ldots A_{a_{(v-k)_{n_0}}}]$$

we know there exists a δ-rule in R' of the form

$$\delta(q'_{v-k}, \gamma) = (q', \omega_{v-k}) \quad \text{where} \quad \omega_{v-k} \in \theta.$$

Using this δ-rule, the above sequence of δ-rules, and the definition of the matrices for M we know that the sequence of trees $\{\beta_i\}_{i=0, \ldots, v-k}$ exists such that $\beta_0 = \alpha'$, $\beta_{v-k-1} = \gamma$, $\beta_{v-k} = \omega_{v-k}$, and

$$\{(q'_i, \beta_i) \vdash (q'_{i+1}, \beta_{i+1})\}_{i=0, 1, \ldots, v-k-1}.$$

This means $(q_0, \alpha') \vdash^* (q'_{v-k}, \omega_{v-k})$ in R. That is, $\alpha' \in A(R)$ and $s \in L(R)$. Therefore, $s \in L(G)$ and α a derivation tree of s in G insures that $s \in L(R')$ and there exists a (k, Σ) value tree α' such that $s = fr(\alpha')$, α and α' are nearly structurally equivalent, and $\alpha' \in A(R')$.

Hence, combining the above results we have that G and R' are nearly structurally equivalent. This insures that G and R are nearly structurally equivalent. QED

Theorem 7.19 summarizes the last two theorems.

Theorem 7.19. For every matrix grammar G there exists an alpha tree acceptor R such that G and R are nearly structurally equivalent and for every alpha tree acceptor R there exists a matrix grammar G such that G and R are nearly structurally equivalent.

As a corollary to the last theorem and the proof of Lemma 3.1 we have the following corollary.

Corollary 7.20. For every matrix grammar G there exists a beta tree acceptor S such that G and S are nearly structurally equivalent and for every beta tree acceptor S there exists a matrix grammar G such that G and S are nearly structurally equivalent.

Corollary 7.21. For every matrix grammar $G = (V_n, V_t, X, M)$ there exists an equivalent matrix grammar $G' = (V'_n, V_t, X', M')$ such that M' consists of matrices m'_i each containing at most two productions.

Proof: First we use the construction of Theorem 7.15 to obtain an alpha tree acceptor nearly structurally equivalent to G. Then we apply Theorem 7.18 to obtain a matrix grammar G' nearly structurally equivalent to the alpha tree acceptor. Clearly $L(G) = L(G')$ and as a result of the second construction all the matrices of G' will have at most two productions. QED

Adding two more definitions we get a weaker result.

Definition 7.22. $\mathcal{M} = \{L(R): R \text{ is an alpha tree acceptor}\}$.

Definition 7.23. $\mathcal{N}$ = {L(G): G is a matrix grammar}.

Corollary 7.24. $\mathcal{M} = \mathcal{N}$.

VIII. NONDETERMINISM FOR MULTI-STATE ALPHA TREE ACCEPTORS

In this section we recall the definition of alphabet nondeterminism and give definitions of two more types of nondeterminism for alpha tree acceptors. We show that the nondeterministic devices are no more powerful than the deterministic alpha tree acceptors. Moreover, we show that for every nondeterministic alpha tree acceptor we can find a deterministic alpha tree acceptor such that the two alpha tree acceptors are nearly structurally equivalent.

Definition 8.1. A State Nondeterministic Alpha Tree Acceptor, a SNTA, is a system $R_s = (Q, \Omega, \delta, Q_0, \Sigma, \theta)$ where Q, Ω, δ, and θ are as for deterministic alpha tree acceptors, $Q_0 \subseteq Q$ is a set of one or more initial states, and the δ-rules are of the form

$$\delta(q, \beta) = \{(p_1, \omega), (p_2, \omega), \ldots, (p_n, \omega)\} \quad \text{and} \quad d(\beta) \geq 1$$

That is, a SNTA in a particular state can reduce a given subtree β to a particular symbol ω but the device has a choice of one or more possible next states.

Definition 8.2. (Very similar to Definition 4.12). An Alphabet Nondeterministic Alpha Tree Acceptor, a ANTA, is a system $R = (Q, \Omega, \delta, q_0, \Sigma, \theta)$ where Q, Ω, q_0, Σ, and θ are as for

deterministic alpha tree acceptors. The δ-rules for R are of the form

$$\delta(q, \beta) = \{(p, \omega_1), (p, \omega_2), \ldots, (p, \omega_n)\} \quad \text{where} \quad d(\beta) \geq 1 .$$

Here each ANTA can reduce a subtree β to one of a set of symbols but the next state is identified by the present state of the ANTA and the subtree reduced.

Definition 8.3. A State-Alphabet Nondeterministic Alpha Tree Acceptor, a SANTA, is a system $R_{sa} = (Q, \Omega, \delta, Q_0, \Sigma, \theta)$ where again the sets Q, Ω, Σ, and θ are as for deterministic alpha tree acceptors. $Q_0 \subseteq Q$ is a set of initial states and all the δ-rules are of the form

$$\delta(q, \beta) = \{(p_1, \omega_1), (p_2, \omega_2), \ldots, (p_n, \omega_n)\} \quad \text{where} \quad d(\beta) \geq 1 .$$

Here a SANTA has a choice for both the next state for a move and the symbol to which the subtree β is reduced.

In all cases we stipulated that $d(\beta) \geq 1$. We do this to facilitate the use of the methods of the previous chapter. Further, we stipulate that all δ-rules of deterministic alpha tree acceptors of this chapter be the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) \geq 1$. This stipulation, Corollary 7.15, and its proof insure that we may assume without

loss of generality that all the δ-rules of each deterministic alpha tree acceptor of this chapter are of the form $\delta(q, \beta) = (p, \omega)$ where $d(\beta) = 1$. Reviewing briefly, Corollary 7.15 insures that for each alpha tree acceptor R we can find an alpha tree acceptor R' such that $A(R) = A(R')$ and R' is of the desired type. The proof of Corollary 7.15 insures that R and R' are structurally equivalent, where the structural equivalence of two alpha tree acceptors is defined below. It is the structural equivalence of alpha tree acceptors that concerns us in this chapter.

Definition 8.4. Two alpha tree acceptors R and R' are structurally equivalent if and only if $L(R) = L(R')$ and a (k, Σ) value tree $\alpha \in A(R)$ if and only if there exists a (k, Σ') value tree $\alpha' \in A(R')$ and α and α' are equal up to a relabeling of nodes.

Definition 8.5. Two alpha tree acceptors, R and R', are nearly structurally equivalent if and only if $L(R) = L(R')$ and a tree $\alpha \in A(R)$ if and only if there is a tree $\beta \in A(R')$ and a tree γ such that α and β are elementary subdivisions of γ.

The following definitions formalize the notion of acceptance by nondeterministic alpha tree acceptors. Since the definitions apply to all types of nondeterministic alpha tree acceptors we let $g \in \{s, \alpha, s\alpha\}$ and $R_g = (Q, \Omega, \delta, Q_0, \Sigma, \theta)$ be a nondeterministic

alpha tree acceptor of the respective type. In the case $g = \alpha$, $|Q_0| = 1$. Just as for deterministic alpha tree acceptors, a configuration of a nondeterministic alpha tree acceptor R_g is a pair (q, β) where $q \in Q$ and β is a (k, Ω) value tree.

Definition 8.6. We say that $(q, \beta') \vdash (p, \beta'')$ in R_g if and only if $q, p \in Q$, β' and β'' are (k, Ω) value trees, $(p, \omega) \in \delta(q, \beta)$, and there is a $b \in D(\beta')$ such that $\beta = \beta'/b$ and $\beta'' = \beta'(b \leftarrow \omega)$. Again we let $\vdash^*$ be the reflexive transitive closure of $\vdash$.

Definition 8.7. $A(R_g) = \{\alpha : \alpha$ is a (k, Σ) value tree and for some $q_0 \in Q_0$, $(q_0, \alpha) \vdash^* (p, \omega)$ in R_g and $\omega \in \theta\}$.

Using Definition 8.7 we stipulate that the terms structurally equivalent and nearly structurally equivalent refer, in a manner analogous to Definition 8.4 and 8.5, to any two alpha tree acceptors R_g and R'_g, where g and g' are elements of $\{\phi, s, \alpha, s\alpha\}$. When $g = \phi$ R_g is a deterministic alpha tree acceptor. Definitions 7.9 and 7.10 concerning the structural equivalence of alpha tree acceptors and matrix grammars are also extended in a natural manner to nondeterministic alpha tree acceptors.

The following Corollary generalizes Corollary 7.16 to all alpha tree acceptors.

Corollary 8.8. Let $g = \{\phi, s, a, sa\}$ and $R_g = (Q, \Omega, \delta, Q_0, \Sigma, \theta)$ be an alpha tree acceptor of the respective type such that all δ-rules are of the form $\delta(q, \beta) \subseteq P(Q \times \Omega)$ and $d(\beta) \geq 1$. Then there exists a SANTA, $R'_{sa} = (Q', \Omega', \delta', Q'_0, \Sigma, \theta)$, such that $A(R_g) = A(R'_{sa})$ and all δ'-rules of R'_{sa} are of the form $\delta'(q, \beta) \subset P(Q' \times \Omega')$ and $d(\beta) = 1$.

Proof: Again the construction of Brainerd [1] will do. To extend the construction to the nondeterministic cases we add a sequence of moves to δ' for each element of the set $(q, \beta) \times \delta(q, \beta)$. This must be done for each pair (q, β) on which δ is defined. It is interesting to note that all of the states we add are "deterministic" states. QED

Definition 8.9. For $g \in \{\phi, s, a, sa\}$, $T_g = \{\Gamma : \Gamma \in T_{\Sigma, k}$ and for some alpha tree acceptor of the respective type, R_g, we have $\Gamma = A(R_g)\}$.

Definition 8.10. For g and g' elements of $\{\phi, s, a, sa\}$, $g \neq g'$ we say $T_g \doteq T_{g'}$ if and only if 1) for each $\Gamma \in T_g$ there exists a $\Gamma' \in T_{g'}$ such that for some nearly structurally equivalent alpha tree acceptors R_g and R'_g, $\Gamma = A(R_g)$ and $\Gamma' = A(R'_g)$; and 2) conversely we stipulate that for each $\Gamma' \in T_g$, there exists a $\Gamma \in T_g$ such that for some nearly structurally equivalent alpha

tree acceptors $R'_{g'}$ and R_g $\Gamma' = A(R'_{g'})$ and $\Gamma = A(R_g)$.

Theorem 8.11. $T \doteq T_s \doteq T_a \doteq T_{sa}$.

Proof: From the definition it is clear that $T \subseteq T_s \subseteq T_{sa}$ and $T \subseteq T_a \subseteq T_{sa}$. Therefore we need only show that for $\Gamma \in T_{sa}$ we can find a $\Gamma' \in T$ and alpha tree acceptors R_{sa} and R of the respective types, such that $\Gamma = A(R_{sa})$, $\Gamma' = A(R)$, and R_{sa} and R are nearly structurally equivalent. This will be done by implementing the methods of the proof of Theorem 7.18 and then applying Theorem 7.15 to the resulting matrix grammar.

$\Gamma \in T_{sa}$ implies there exists a SANTA $R_{sa} = (Q, \Omega, \delta, Q_0, \Sigma, \theta)$ such that $\Gamma = A(R_{sa})$. As a result of Corollary 8.8 and its proof we may assume without loss of generality that all δ-rules of R_{sa} are of the form $\delta(q, \beta) \subseteq P(Q \times \Omega)$ where $d(\beta) = 1$. As in the proof of Theorem 7.18 we draw the state diagram of R_s. This state diagram will be the same as the state diagram in Theorem 7.18 except that situations as depicted by Figure 8.1 can occur. These new situations can arise as a result of allowing nondeterminism.

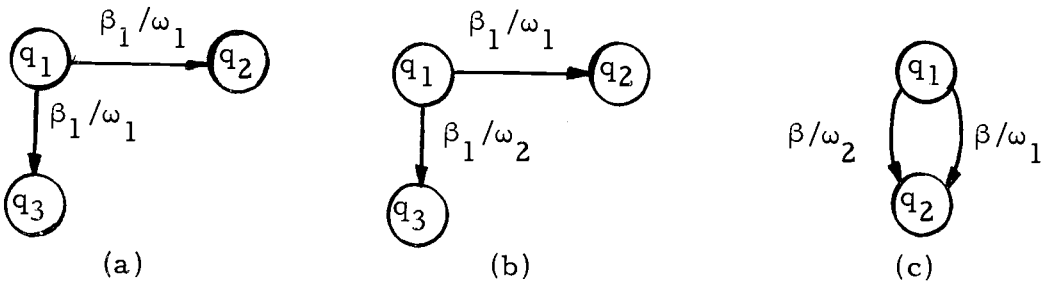


Figure 8.1.

In the following proof we will apply the proof of Theorem 7.18 to obtain a matrix grammar G such that G and $R_{s\alpha}$ are nearly structurally equivalent; then we apply Theorem 7.15 to obtain a deterministic alpha tree acceptor R such that G and R are structurally equivalent. We get that G and R will be structurally equivalent as a result of the way that we construct G. That is, G is constructed in such a manner that Lemma 7.14 can be bypassed when applying Theorem 7.15. Since R will be structurally equivalent to G and G and $R_{s\alpha}$ will be nearly structurally equivalent, R and $R_{s\alpha}$ will be nearly structurally equivalent which is what we need. To that end we add some symbols to the arrows of the state diagram of $R_{s\alpha}$. We index the arrows egressing from each node as per Figure 8.2. In Figure 8.2 it might be the case that for some $i \neq j$, $p_i = p_j$, $\beta_i = \beta_j$, or $\omega_i = \omega_j$.

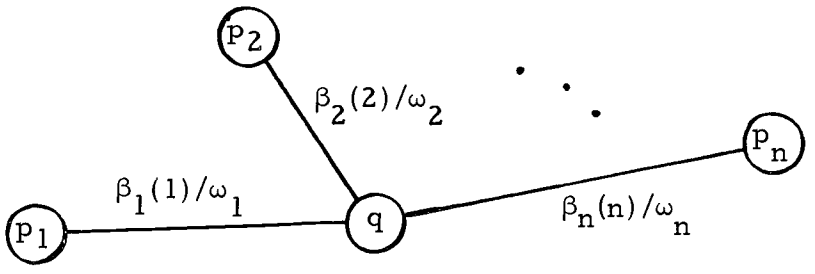


Figure 8.2.

Let $h = \max_{q \in Q} \{k: q \text{ has } k \text{ egressing arrows}\}$.

Now we proceed with a construction parallel to that of the proof of Theorem 7.18. Our goal here is a matrix grammar $G = (V_n, V_t, X, M)$ such that G and R_{sa} are nearly structurally equivalent. As in Theorem 7.17, we let $V_t = \Sigma$, but here

$$V_n = A_\omega : \omega \in \Omega \cup \{(q(i), \omega) : q \in Q,\ i \leq k,\ \text{and}\ \omega \in \Omega\}$$
$$\cup \{(\omega, i, q) : q \in Q,\ i \leq k,\ \text{and}\ \omega \in \Omega\}.$$

Again the set M for G is constructed in five steps.

1) For all trees of the form $\bullet\sigma$ such that $\sigma \in \Sigma \cap \theta$, we add $[X \rightarrow \sigma]$ to M. If $\Sigma \cap \theta = \phi$, then no matrices are added to M at this step.

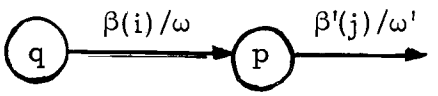


Figure 8.3.

2) We locate all states p on the state diagram such that the situation is as depicted by Figure 8.3 and $\omega \in \theta$. For each such state p and each such entering arrow if $\beta = a_0(a_1\, a_2 \cdots a_n)$ where $a_i \in \Omega$, $i = 0, 1, \ldots, n$, then we add the matrix

$$[X \rightarrow (q(i), a_1)A_{a_2} A_{a_3} \cdots A_{a_n}] \text{ to } M.$$

In case $q \in Q_0$, we should also add

$$[X \rightarrow (a_1, i, q)A_{a_2} A_{a_3} \ldots A_{a_n}] \text{ to } M.$$

3) For all situations as depicted by Figure 8.3 where β is as for 2) and $\beta' = a'_0(a'_1\ a'_2 \ldots a'_{n'})$, $a'_i \in \Omega$ for $i = 0, 1, \ldots, n'$, we add the matrix

$$\left[\begin{array}{l} (p(j), a'_1) \rightarrow A_{a'_1} \\ \\ \qquad A_\omega \rightarrow (q(i), a_1)A_{a_2} A_{a_3} \ldots A_{a_n} \end{array}\right] \text{ to } M.$$

Note that this must be done for each arrow egressing from state p.

4) In all situations where step 2) applies and $q \in Q_0$ of $R_{a\alpha}$ we add the following matrix to M.

$$\left[\begin{array}{l} (p(j), a'_1) \rightarrow A_{a'_1} \\ \\ \qquad A_\omega \rightarrow (a_1, i, q)A_{a_2} A_{a_3} \ldots A_{a_n} \end{array}\right]$$

Again β and β' are as defined above.

5) For all $\sigma \in \Sigma$ we add the matrix $[A_\sigma \rightarrow \sigma]$ to M, for all $\sigma \in \Sigma$ and $(\sigma, i, q) \in V_n$ we add $[(\sigma, i, q) \rightarrow \sigma]$ to M.

We claim that R_s and G are nearly structurally equivalent. This follows immediately from the second half of the proof of Theorem 7.18. That is, the argument is exactly the same and all the necessary

sequences are obtained in the same manner.

From the construction of M we can see that each matrix is identified by the right hand side of its last production. Furthermore, X appears in matrices that have only a single production. It is also important to note that X does not appear on the right hand side of any production and that X when it appears, is only in the first production of a matrix. From inspection of the proof of Theorem 7.15 we can see that these conditions are sufficient to allow us to apply the construction in the proof directly to G and obtain a deterministic alpha tree acceptor $R' = (Q', \Omega', \delta', q_0', \Sigma, \theta')$ such that G and R' are structurally equivalent. Since $R_{s\alpha}$ and G are nearly structurally equivalent we can conclude that $R_{s\alpha}$ and R' are nearly structurally equivalent. Thus $T_{s\alpha} \subseteq T$. QED

Definitions 8.1, 8.2, and 8.3 can be extended to nondeterministic beta tree acceptors, nondeterministic automata that accept by final state instead of final symbol. In a manner parallel to Definitions 8.6 and 8.7 we can define acceptance by nondeterministic beta tree acceptors, and we can extend the notions of structural equivalence and near structural equivalence to nondeterministic beta tree acceptors. Again a corollary similar to 8.8 applies; that is, we need only consider nondeterministic beta tree acceptors with δ-rules such that $d(\beta) = 1$.

Definition 8.12. Let $g \in \{\phi, s, a, sa\}$,

$V_g = \{\Gamma : \Gamma$ is a set of (k, Σ) value trees for some k and alphabet Σ and there is a beta tree acceptor S_g of the respective type such that $\Gamma = B(S_g)\}$.

We can extend Definition 8.10 to apply to relation $\doteq$ to the sets V_g for $g \in \{\phi, s, a, sa\}$.

Corollary 8.13. $V \doteq V_s \doteq V_a \doteq V_{sa}$.

Proof: We can either extend the ideas of Chapter 3 to show that the corresponding types of α-recognition and β-recognition carry over to nondeterministic alpha tree acceptors and nondeterministic beta tree acceptors, or we can apply the methods of the previous proof to nondeterministic beta tree acceptors. The only changes that we need make are in the area of constructing the matrices for M. The first two steps are changed as indicated below.

1) If $Q_0 \in \Phi$, then $[X \rightarrow \sigma]$ is in M for all $\sigma \in \Sigma$.

2) We must locate all states p on the state diagram such that the situation is as depicted by Figure 8.3 and $p \in \Phi$.

The remainder of the proof requires no change. QED

IX. CONCLUSION

In this thesis we have generalized the standard tree automata. We have done this by defining tree acceptors that operate in a manner very similar to standard finite automata; that is, our tree acceptors have an input, a finite memory, and an output. We have seen that the set of one state alpha tree acceptors is equivalent to the set of standard tree automata It has been shown that tree automata can be used to characterize the sets of derivation trees of extended context-free grammars. We have obtained a similar result relating alpha (beta) tree acceptors and λ-free context-free matrix grammars. Our result is that alpha (beta) tree acceptors and λ-free context-free matrix grammars are nearly structurally equivalent. The reasons that we are able to only obtain near structural equivalence are 1) the problem of unique invertibility for matrix grammars is not solved and 2) the construction used to obtain a matrix grammar from an alpha (beta) tree acceptor alters slightly the involved set of trees.

With respect to the above mentioned problems we note that work is being done with generalizations of Brainerd's regular systems [1]. The generalization that we are refering to is the matrix tree generating systems (MTGS) [2]. A MTGS is a matrix grammar except that the productions of a MTGS involve rewriting labeled trees instead of strings. Possibly results from this work will yield stronger results

concerning relations between alpha (beta) tree acceptors and λ-free context-free matrix grammars.

Recent work in syntactic techniques of pattern recognition [8] and [10], led to our interest in generalizing standard tree automata. Tree automata have been used to drive pattern recognizers. Hopefully, alpha and beta tree acceptors can be used to drive pattern recognizers. Example 7.17 demonstrates that it is easy to specify the nature of patterns (trees) to be recognized and then build an alpha tree acceptor that recognizes the described set of trees. Theorem 4.17 demonstrates that alpha or beta tree acceptors are more powerful than standard tree automata.

The above paragraph is related to the applications of alpha or beta tree acceptors. There are many related theoretical questions that remain unanswered. Of these are the questions concerning Boolean closure. After obtaining the results of Chapter 7 it was hoped that we might be able to use the techniques of standard automata to help resolve these questions for alpha or beta tree acceptors and, hence, for matrix grammars. However, we have been unable to generalize the usual techniques. The main problem appears to be the inherent nondeterminism of even the deterministic alpha and beta tree acceptors Specifically, for standard string automata the next input is either the symbol under the read head or the symbol to its immediate left or right. With alpha or beta tree acceptors we are not able to

identify the next input symbol with such reliability. The next input could be any subtree on the frontier of the tree being parsed.

To exemplify this problem we consider a technique used for proving that the complement of a regular set is regular. First one builds a regular automaton that accepts the regular set. If the automaton does not accept some strings by halting in the middle of the strings, then we construct a second automaton from the first by adding a nonaccepting state to the first that is entered when the first automaton would have halted in the middle of a string In this nonaccepting state the second is then able to finish reading the input string. We conclude the proof by building a third automaton. The third automaton is a copy of the second except that all nonaccepting states of the second become accepting states of the third and vice-versa. The set of strings accepted by the third automaton is the complement of the set accepted by the first. The important step is that we were able to force the second to read its entire input in every case. We have been unable to apply this technique to alpha tree acceptors. Attempts have always yielded a "third" alpha (or beta) tree acceptor that accepts every input. The reason appears to be the inherent nondeterminism of the devices.

Another direction to attack the problem might be to characterize sets of trees that are not accepted by alpha or beta tree acceptors. An easy proof that context-free languages are not closed under union

is based on the fact that $\{a^n b^n c^n : n \geq 1\}$ is not a context-free language. We have been unable to make any such characterization of sets not accepted by alpha or beta tree acceptors.

Finally, we mention again the problem of whether or not to allow relabeling moves adds power to alpha or beta tree acceptors. We have not allowed such moves in most of the alpha or beta tree acceptors considered here since by their very definition tree automata are not allowed relabeling moves and relabeling moves have no meaning in the framework of matrix grammars. In the setting of matrix tree generating systems relabeling moves are well defined and an answer has been found [2]. There we have developed a near structurally equivalent relationship again.

LIST OF REFERENCES

1. Brainerd, W.S. Tree generating regular systems. Information and Control. 14:217-231. 1969.

2. Cook, C.R. and D.L. Vomocil. Matrix tree generating systems. Paper in preparation.

3. Doner, J.E. Tree acceptors and some of their applications. Journal of Computers and Systems Science. 4:406-451. 1970.

4. Fu, K.S. and B.K. Bhargava. Tree systems for syntactic pattern recognition. IEEE Transactions on Computers. 22:1087-1099. 1973.

5. Gorn, S. Explicit definitions and linguistic dominoes. Systems and Computer Science. ed.s J.F. Hart and S. Takasu. Toronto: University of Toronto Press, 1967.

6. Hopcroft, J.E. and J.D. Ullman. Formal Languages and Their Relation to Automata. Reading, Mass.: Addison-Wesley Pub. Co., 1969.

7. Korfhage, R. Discrete Computational Structures. New York: Academic Press, 1974.

8. Miller, W.F. and A.C. Shaw. Linguistic methods in picture processing - a survey. Proceedings of the Fall Joint Computer Conference. 33:279-290. 1968.

9. Salomaa, A. Formal Languages. New York: Academic Press, 1973.

10. Shaw, A.C. A formal picture description scheme as a basis for picture processing systems. Information and Control. 14:9-52. 1969.

11. Thatcher, J.W. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. Journal of Computers and Systems Science. 1:317-322. 1967.