International Journal of Computer and Information Sciences, Vol. 12, No. 1, 1983

Regular-Like Tree Expressions[1]

A. Barrero,[2] M. G. Thomason,[3] and R. C. Gonzalez[2]

Received November 1982; revised January 1983

Regular-like expressions provide a compact notation for the explicit description of context-free languages. In this paper we extend these concepts by establishing a one-to-one correspondence between expansive tree languages and regular-like sets of trees. Algorithms are developed for constructing an expansive tree grammar whose language is defined by a given regular-like tree (RLT) expression, and vice versa. These results are then used to obtain RLT equations that describe sets of tree resulting from regularity-preserving transformations associated with three types of errors commonly found in practice.

KEY WORDS: Trees; regular expressions; expansive tree grammars; regularity-preserving transformations; context-free languages.

1. INTRODUCTION

Regular and regular-like expressions are used as compact representations of string languages that are of types three and two, respectively, in the Chomsky hierarchy.[1] The requirements of certain areas, such as syntactic pattern recognition, for the representation of structures that cannot be conveniently handled by string languages has motivated investigations into the properties and applications of tree grammars and languages.[2–6] In this paper we extend the concept of regular-like expressions by establishing a one-to-one correspondence between expansive tree languages and regular-like sets of trees.

In the following sections, regular-like tree (RLT) expressions are defined and it is shown that the set of trees defined by an RLT expression is

[1] Work supported by the Office of Naval Research, Arlington, VA, under Contract N00014-75-C-0545.

[2] Department of Electrical Engineering, University of Tennessee, Ferris Hall, Knoxville, Tennessee 37916.

[3] Department of Computer Science, University of Tennessee, 8 Ayres Hall, Knoxville, Tennessee 37916.

0091-7036/83/0200-0001$03.00/0

828/12/1-1

generated by an expansive tree grammar. It is also shown that an RLT expression can be constructed for the language of an expansive tree grammar. In addition, three types of errors in trees that are commonly encountered in practice are studied in terms of their representation as regularity-preserving transforms. The development is based on operations of RLT expressions that yield descriptions of the transformed tree structures.

2. CONSTRUCTION OF EXPANSIVE TREE GRAMMARS THAT GENERATE REGULAR-LIKE SETS

We use the standard definition of an *expansive tree grammar* $G = (V_T, V_N, P, S)$ as finite sets of terminals, nonterminals, expansive tree productions, and starting trees.[2,4,5] A *V-tree* is a tree with node labels from a finite set V.

Definition 1. Let E and F be two sets of V-trees and let k be in V. Define the following operations with E and F:

$E \cdot kF =$ the set of V-trees obtained by taking trees of F and replacing each node in the frontier with label k by a tree from E.

$E^{0;k} = \{k\}$, the tree with a single node labeled k.

$E^{n+1;k} = E \cdot kE^{n;k}$, $n \geqslant 0$.

$$E^{+k} = \bigcup_{n=1}^{\infty} E^{n;k}.$$

$$E^{*k} = \bigcup_{n=0}^{\infty} E^{n;k}.$$

Lemma 1. $\cdot k$ is an associative operation with identity $\{k\}$.

Proof. It follows from the previous definition that $E \cdot k \cdot (F \cdot kG) = (E \cdot kF) \cdot kG$, and $E \cdot k\{k\} = \{k\} \cdot kE$. ∎

Definition 2. *Regular-like tree* (RLT) *expressions* over V and the sets of trees they denote are defined recursively as follows: (1) $\varnothing$ is an RLT expression over V denoting the null set $\varnothing$. (2) a is an RLT expression denoting the set $\{a\}$. (3) $\begin{array}{c} a \\ / \quad \backslash \\ x_1 \cdots x_n \end{array}$ is an RLT expression denoting the set $\left\{ \begin{array}{c} a \\ / \quad \backslash \\ x_1 \cdots x_n \end{array} \right\}$. (4) If p and q are RLT expressions denoting sets P and Q, respectively, then: (a) $p + q$ is an RLT expression denoting the set $P \cup Q$; (b)

$p \cdot kq$ is an RLT expression denoting the set $P \cdot kQ$; (c) p^{*k} is an RLT expression denoting the set P^{*k}. (5) Nothing else is an RLT expression.

Theorem 1. Let E be a regular-like tree set; then E is generated by an expansive tree grammar.

Proof by construction of the grammars. It is evident that $\varnothing$, $\{a\}$, and $\{\ \overset{a}{\underset{X_1 \cdots X_n}{\diagup\ \diagdown}}\ \}$ can be generated by expansive tree grammars. Let P and Q be sets of V-trees generated by the expansive tree grammars $G_P = (V, V_P, P_P, S_P)$ and $G_Q = (V, V_Q, P_Q, S_Q)$ with $V_P \cap V_Q = \varnothing$. Then the sets, (1) $P \cup Q$, (2) $P \cdot kQ$, k in V, and (3) P^{*k}, k in V, can be generated by the following expansive tree grammars: (1) $G_1 = (V,\ V_P \cup V_Q,\ P_P \cup P_Q,\ S_P \cup S_Q)$. (2) $G_2 = (V, V_Q \cup V_P - S_P, P_2, S_Q)$, where P_2 is $(P_Q - \{X \to k \mid X$ is in $V_Q\})$ joined with $\{X \to \overset{x}{\underset{X_1 \cdots X_n}{\diagup\ \diagdown}} \mid X \to k$ is in P_Q, $Y \to \overset{x}{\underset{X_1 \cdots X_n}{\diagup\ \diagdown}}$ is in P_P, Y is in $S_P\}$ joined with $(P_P - \{Y \to \overset{x}{\underset{X_1 \cdots X_n}{\diagup\ \diagdown}} \mid Y$ is in $S_P\})$. (3) Let K be a symbol not in V_P, and let $G_3 = (V,\ V_P \cup \{K\},\ P_3,\ S_P \cup \{K\})$, where $P_3 = \{K \to k\} \cup P_P \cup \{X \to \overset{x}{\underset{X_1 \cdots X_n}{\diagup\ \diagdown}} \mid X \to k$ is in P_P, $Y \to \overset{x}{\underset{X_1 \cdots X_n}{\diagup\ \diagdown}}$ is in P_P, Y is in $S_P\}$. ∎

3. CONSTRUCTION OF RLT EXPRESSIONS FOR LANGUAGES OF EXPANSIVE TREE GRAMMARS

The following definitions and lemmas establishes the properties of RLT expressions which are similar to properties of conventional regular expressions. In this section, Theorem 3 establishes that expansive tree languages are regular-like tree sets by using the RLT equations.

Definition 3. Two RLT expressions are *equal*, denoted $=$, iff they represent the same set of trees.

Lemma 2. Let α, β, and γ be RLT expressions. Then

$$\alpha + \beta = \beta + \gamma \tag{1}$$

$$\varnothing^{*k} = k \tag{2}$$

$$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \tag{3}$$

$$\alpha \cdot k(\beta \cdot k\gamma) = (\alpha \cdot k\beta) \cdot k\gamma \tag{4}$$

$$\alpha \cdot k(\beta + \gamma) = \alpha \cdot k\beta + \alpha \cdot k\gamma \tag{5}$$

$$(\alpha + \beta) \cdot k\gamma = \alpha \cdot k\gamma + \beta \cdot k\gamma \tag{6}$$

$$\alpha \cdot kk = \alpha = k \cdot k\alpha \tag{7}$$

$$\varnothing \cdot k\alpha = \alpha \cdot k\varnothing = \varnothing \tag{8}$$

$$\alpha^{*k} = \alpha + \alpha^{*k} \tag{9}$$

$$(\alpha^{*k})^{*k} = \alpha^{*k} \tag{10}$$

$$\alpha + \alpha = \alpha \tag{11}$$

$$\alpha + \varnothing = \alpha \tag{12}$$

$$\alpha \cdot k\alpha^{*k} = \alpha^{*k} \cdot k\alpha = \alpha^{+k} \tag{13}$$

$$\alpha^{+k} = \alpha^{*k} \text{ iff } k \subseteq \alpha. \tag{14}$$

Proof. Equations (1), (3), (4), (5), (6), (7), (8), (11), and (12) follow immediately from the definitions. Equation (2): $\alpha^{*k} = k + \alpha \cdot kk + \alpha \cdot k(\alpha \cdot kk) + \cdots$; then from (8), $\varnothing^{*k} = k$. Equation (9): $\alpha^{*k} = k + \alpha \cdot kk + \cdots$; but from (7), $\alpha \cdot kk = \alpha$, so $\alpha \subset \alpha^*$. Equation (10): follows from $\alpha^{*k} \cdot k\alpha^{*k} = \alpha^{*k}$. Equation (13): follows by expansion of α^{*k} and the associativity of $\cdot k$. Equation (14): follows by expansion of α^{+k} and α^{*k}. ∎

Solution of an RLT equation such as

$$X = X \cdot k(\ \overset{a}{\underset{b}{\diagup}}\ \overset{}{\underset{k}{\diagdown}}\ ) + c$$

can be found by methods similar to the solution of conventional regular expression equations (cf. Ref. 1). By direct substitution, we have the following lemma.

Lemma 3. Given the RLT equation $X = X \cdot k\alpha + \beta$, where α and β are RLT expressions, then $X = \beta \cdot k\alpha^{*k}$ is a solution to the equation. ∎

Lemma 4. Given the RLT equation $X = X \cdot k\alpha + \beta$, and a solution χ, then $\beta \cdot k\alpha^{*k} \subset \chi$.

Proof. Since χ is a solution, $\chi = \chi \cdot k\alpha + \beta$, so that $\beta \subset \chi$, $\beta \cdot k\alpha \subset \chi$, $(\beta \cdot k\alpha) \cdot k\alpha \subset \chi$,..., $\beta + \beta \cdot k\alpha + \beta \cdot k(\alpha \cdot k\alpha) + \cdots \subset \chi$. But $\beta = \beta \cdot kk = \beta \cdot k\alpha^{0;k}$, $\beta \cdot k\alpha = \beta \cdot k\alpha^{1;k}$, $\beta \cdot k(\alpha \cdot k\alpha) = \beta \cdot k\alpha^{2;k}$,.... Thus $\beta \cdot k(\alpha^{0;k} + \alpha^{1;k} + \alpha^{2;k} + \cdots) \subset \chi$, and $\beta \cdot k\alpha^{*k} \subset \chi$. ∎

Lemma 5. $X=\beta \cdot k\alpha^{*k}$ is the unique solution to $X=X \cdot k\alpha+\beta$ iff the tree k is not in α.

Proof. If: Assume $\beta \cdot k\alpha^{*k}$ is the unique solution. If k is in α, it follows by direct substitution that $(\beta+\gamma) \cdot k\alpha^{*k}$ is a solution for any γ. Since this contradicts the original assumption, it follows that k is not in α.

Only if: Assume k is not in α. Then by Lemma 6 any solution to the equation can be written in the form $X=\beta \cdot k\alpha^{*k}+\gamma$, where γ is not contained in $\beta \cdot k\alpha^{*k}$. But

$$\begin{aligned}\beta \cdot k\alpha^{*k}+\gamma &= (\beta \cdot k\alpha^{*k}+\gamma) \cdot k\alpha+\beta \\ &= \beta \cdot k\alpha^{*k} \cdot k\alpha+\beta+\gamma \cdot k\alpha \\ &= \beta \cdot k\alpha^{*k}+\gamma \cdot k\alpha\end{aligned}$$

which implies that $\gamma=\gamma \cdot k\alpha$. Now, if $\gamma \neq \varnothing$, then k must be in α since this is the only way in which a tree of minimum depth in γ can be in $\gamma \cdot k\alpha$. But k is not in α, by assumption; therefore $\gamma=\varnothing$, and $\beta \cdot k\alpha^{*k}$ is the unique solution. ∎

Given an expansive tree grammar, the productions in the grammar can be written as the RLT equations with the nonterminals as the unknowns. Each unknown will represent the set of tree that can be generated by the corresponding nonterminal, and solutions to the equations will be unique.

Definition 4. A set of RLT equations with indeterminates $\{X_1,\ldots,X_n\}$ is in *standard form* if for each X_i there is an equation of the form

$$X_i=\sum_{j=1}^{m_1} a_j+\sum_{j=1}^{m_2} X_{l1} \cdot x_{l1}X_{l2} \cdot x_{l2} \cdots x_{lP_j} \cdot X_{lP_j}\left(\begin{array}{c} b_j \\ \diagup \quad \diagdown \\ x_{l1} \cdots x_{lP_j}\end{array}\right), \qquad \text{for } lP_j>0,$$

where $\{X_1,\ldots,X_n\}$, $\{a_1,\ldots,a_{m_1}\} \cup \{b_1,\ldots,b_{m_2}\}$, and $\{x_1,\ldots,x_n\}$ have no common elements and there is a homomorphism h: $\{X_1,\ldots,X_n\} \to \{x_1,\ldots,x_n\}$: $h(X_i)=x_i$.

Lemma 6. Given a set of RLT equations, it is possible to write each equation in normal form; that is, for each unknown X_i,

$$X_i=X_i \cdot x_i\alpha_i+\beta_i$$

Proof. It is only necessary to point out that in the preceding definition the operators $\cdot x_k$ can be commuted. Thus

$$X_i = X_i \cdot x_i \sum_{j=1}^{m_2} X_{l1} x_{l1} \cdots \left(\begin{array}{c} b_j \\ \diagup \quad \diagdown \\ x_{l1} \cdots x_{lPj} \end{array} \right) + \sum_{j=1}^{m_1} a_j \qquad \blacksquare$$

The preceding results are the basis for the following algorithm.

Algorithm 1. Solution of a set of RLT equations in standard form:

Input. A set of RLT equations in standard form over $\{X_1, \ldots, X_n\}$.

Output. A set of solutions of the form $X_i = \alpha_i$, $i = 1, \ldots, n$, where α_i is a RLT expression in which no X's appear.

Method. The method consists of successive elimination and back-substitution.

Step 1: Let $i = 1$.

Step 2: If $i = n$, go to Step 4. Otherwise, write the equation for X_i as $X_i = X_i \cdot x_i \alpha_i + \beta_i$. Then in the equations for $X_{i+1}, \ldots, X_n$, replace each occurence of X_i by $\beta_i \cdot x_i \alpha_i^{*x_i}$.

Step 3: Increment i by 1 and go to Step 2.

Step 4: Write the equation for X_i as $X_i = X_i \cdot x_i \gamma_i + \delta_i$. Let $X_i = \delta_i \cdot x_i \gamma_i^{*x_i}$.

Step 5: If $i = 1$, terminate. Otherwise, replace $\delta_i \cdot x_i \gamma_i^{*x_i}$ for X_i in the equations for $X_{i-1}, \ldots, X_1$. Decrement i by 1 and go to Step 4.

Lemma 7. Steps 2 and 4 of the Algorithm are always executable.

Proof. The proof follows from the commutativity of $\cdot x_{ki}$ and $\cdot x_{lj}$. $\blacksquare$

Theorem 2. Algorithm 1 gives the unique solution to a set of RLT equations in standard form. The proof follows from Lemmas 3 through 7. $\blacksquare$

Attention can now be given to the sets generated by expansive tree grammars. An important aspect is to be able to write an RLT equation in standard from for the set of trees generated by a nonterminal.

Lemma 8. Given an expansive tree grammar $G = (V_T, V_N, P, S)$, we can write an RLT equation in standard form for the set of trees generated by a nonterminal.

Proof. Let $V_N = \{X_1, ..., X_n\}$, consider V_N as a set of indeterminates, and let X_i denote the set of trees generated by a nonterminal X_i. Let

$$X_i = \sum_{j=1}^{m_1} a_j + \sum_{j=1}^{m_2} X_{l1} \cdot x_{l1} \cdots X_{lPj} \cdot x_{lPj}\left(\begin{array}{c} b_j \\ x_{l1} \cdots x_{lPj} \end{array}\right)$$

for all productions in P of the form $X_i \to a_j$, or

$$X_i \to \begin{array}{c} b_j \\ X_{lp1} \cdots X_{lPj} \end{array}$$

∎

Theorem 3. The set of trees generated by an expansive tree grammar is regular-like.

Proof. The proof is by construction of an expression denoting the set generated by the grammar. Let $G = (V_T, V_N, P, S)$, write a set of RLT equations in standard form over V_N as indicated in Lemma 8, and use Algorithm 1 to solve them. Then let

$$L = \sum \delta_{li} \cdot x_{li} \gamma_{li}^* x_{li}$$

where the summation is over all lj such that X_{li} is in S. By Lemma 7 and Theorem 2, $L = L(G)$. ∎

Example. To illustrate ideas developed in this section, we consider an expansive tree grammar used in syntactic pattern recognition to describe inductor-capacitor electrical circuits with voltage source e, ground g, and repetitive $l-c$ sections.[4,5] The grammar is $G = (V_T, V_N, P, S)$ where $V_T = \{\$, e, g, l, c\}$; $V_N = \{A, E, G, L, C\}$; $S = \{A\}$, and there are six productions: (a) $A \to \begin{array}{c} \$ \\ E \quad L \end{array}$, (b) $E \to \begin{array}{c} e \\ | \\ G \end{array}$, (c) $G \to g$, (d) $L \to \begin{array}{c} l \\ C \quad L \end{array}$, (e) $C \to \begin{array}{c} c \\ | \\ G \end{array}$, (f) $L \to \begin{array}{c} l \\ | \\ C \end{array}$. The first step is to write a set of RLT equations in standard form. The construction of Lemma 8 yields

$$A = E \cdot w\left(L \cdot y\left(\begin{array}{c} \$ \\ w \quad y \end{array}\right)\right) \tag{15}$$

$$E = G \cdot x\left(\begin{array}{c} e \\ | \\ x \end{array}\right) \tag{16}$$

$$G = g \tag{17}$$

$$L = L \cdot y(C \cdot z(\underset{z}{\underset{|}{l}}\ \underset{y}{\underset{|}{}}\ )) + C \cdot z(\underset{z}{\underset{|}{l}}) \tag{18}$$

$$C = C \cdot x(\underset{x}{\underset{|}{c}}) \tag{19}$$

It is now possible to apply Algorithm 1 to this set of equations; however, in this particular instance, all that is required is to solve Eq. (18) and perform judicious substitutions. From Lemma 5 we know that

$$L = (C \cdot z(l)) \cdot y(C \cdot z(\underset{z}{\underset{|}{l}}\ \underset{y}{\underset{|}{}}\ ))^{*y}$$

is the unique solution to Eq. (18). After a series of substitutions, we obtain

$$A = (g \cdot x(\underset{x}{\underset{|}{e}})) \cdot w((g \cdot x(\underset{x}{\underset{|}{l}})) \cdot y((g \cdot x(\underset{x}{\underset{|}{l}})) \cdot z(\underset{z}{\underset{|}{l}}\ \underset{y}{\underset{|}{}}\ ))^{*y} \cdot y(\underset{w}{\underset{|}{\$}}\ \underset{y}{\underset{|}{}}\ ))$$

Further manipulations yield the equivalent expression,

$$A = (\underset{g}{\underset{|}{c}}) \cdot y(\underset{c}{\underset{|}{l}}\ \underset{y}{\underset{|}{}}\ )^{*y} \cdot y(\underset{e}{\underset{|}{\$}}\ \underset{y}{\underset{|}{}}\ )$$

which has a one-to-one correspondence with the equivalent regular tree grammar. The $*k$ operation only occurs where there are instances of repetitive trees.

4. REGULARITY-PRESERVING TRANSFORMATIONS

It is useful at times to describe certain regularity-preserving transformations via operations on RLT expressions. The three transformations in this section represent common kinds of errors that arise, for example, in automated analysis of patterns and in trees used to represent patterns.[4,5,7]

Definition 5. Given a V_T-tree t, let an α error denote the deletion of a proper subtree of t (a subtree other than the tree itself); a β error, the

insertion of a single node descendent at a node of t; and a γ error, the change of the label at a node of t. Let V_T^T denote the V_T-trees. Define the operators,

$\alpha(t) = \{t' \mid t' \text{ in } V_T^T,\ t' \text{ obtained from } t \text{ by the deletion of a proper subtree}\}$;

$\beta(t) = \{t' \mid t' \text{ in } V_T^T,\ t' \text{ obtained by the insertion of a single node descendent at any node of } t\}$, and

$\gamma(t) = \{t' \mid t' \text{ in } V_T^T,\ t' \text{ obtained from } t \text{ by changing the label of any node of } t\}$.

Definition 6. Let V_N and V_T be two finite nonempty sets of symbols which are disjoint. Let T be a nonempty set of trees with nodes labeled from $V = V_N \cup V_T$. Let t be a tree in T. Then

$\alpha_{V_T}(t) = \{t' \mid t' \text{ in } V^T,\ t' \text{ obtained from } t \text{ by the deletion of a proper subtree}\}$,

$\beta_{V_T}(t) = \{t' \mid t' \text{ in } V^T,\ t' \text{ obtained from } t \text{ by the insertion of a single node descendent labeled } x,\ x \text{ in } V_T, \text{ at any node of } t \text{ labeled } y,\ y \text{ in } V_T\}$,

$\gamma_{V_T}(t) = \{t' \mid t' \text{ in } V^T,\ t' \text{ obtained from } t \text{ by changing the label at node of } t \text{ from } x \text{ to } y,\ x \text{ and } y \text{ in } V_T,\ x \neq y\}$.

Finally, for transformations on sets of trees,

$\alpha_{V_T}(T) = \{t' \mid t' \text{ in } \alpha_{V_T}(t),\ t \text{ in } T\}$.

$\beta_{V_T}(T) = \{t' \mid t' \text{ in } \beta_{V_T}(t),\ t \text{ in } T\}$.

$\gamma_{V_T}(T) = \{t' \mid t' \text{ in } \gamma_{V_T}(t),\ t \text{ in } T\}$.

The next step is to determine the effect of the operators on the basic trees used to define the RLT sets.

Lemma 9. Let V_T be a finite nonempty set of symbols. Then

$\alpha_{V_T}(\varnothing) = \varnothing$,

$\alpha_{V_T}(a) = \varnothing$, for all a in V_T,

$\alpha_{V_T}($  $) =$  $+$  $+ \cdots +$  ;

$$\beta_{V_T}(\varnothing) = \varnothing,$$

$$\beta_{V_T}(a) = \sum_{x \text{ in } V_T} x \cdot k(a), \; k \text{ not in } V_T,$$

$$\beta_{V_T}\left(\begin{array}{c} a \\ X_1 \cdots X_n \end{array}\right) = \sum_{x \text{ in } V_T} x \cdot k\left(\begin{array}{c} a \\ k\ X_1 \cdots X_n \end{array} + \cdots + \begin{array}{c} a \\ X_1 \cdots X_n\ k \end{array}\right),$$

where $X_1, \ldots, X_n$ are not in V_T;

$$\gamma_{V_T}(\varnothing) = \varnothing,$$

$$\gamma_{V_T}(a) = \sum_{\substack{x \text{ in } V_T \\ x \neq a}} x$$

$$\gamma_{V_T}\left(\begin{array}{c} a \\ X_1 \cdots X_n \end{array}\right) = \sum_{\substack{x \text{ in } V_T \\ x \neq a}} \begin{array}{c} x \\ X_1 \cdots X_n \end{array},$$

where $X_1, \ldots, X_n$ are not in V_T.

Proof. The proof follows immediately from the definitions. Moreover, the resulting sets are regular-like. ∎

Lemma 10. Let P and Q be regular-like sets over $V_T \cup \{k\}$, k not in V_T. Then: (a) $\alpha_{V_T}(P \cup Q) = \alpha_{V_T}(P) \cup \alpha_{V_T}(Q)$, (b) $\beta_{V_T}(P \cup Q) = \beta_{V_T}(P) \cup \beta_{V_T}(Q)$, (c) $\gamma_{V_T}(P \cup Q) = \gamma_{V_T}(P) \cup \gamma_{V_T}(Q)$.

Proof. The result follows immediately by applying the operators to one tree at a time. ∎

Lemma 11. Let P and Q be regular-like sets over $V_T \cup \{k\}$, k not in V_T. Then: (a) $\alpha_{V_T}(P \cdot kQ) = \alpha_{V_T}(P) \cdot kQ + P \cdot k\alpha_{V_T}(Q)$, (b) $\beta_{V_T}(P \cdot kQ) = \beta_{V_T}(P) \cdot kQ + P \cdot K\beta_{V_T}(Q)$, (c) $\gamma_{V_T}(P \cdot kQ) = \gamma_{V_T}(P) \cdot kQ + P \cdot k\gamma_{V_T}(Q)$.

Proof. $P \cdot kQ$ is obtained by replacing nodes with label k at the frontier of trees in Q with trees from P. Modifications at a node not labeled with k can be done before or after the replacement. ∎

Lemma 12. Let P be a regular-like set over $V_T \cup \{k\}$, k not in V_T, and the tree k not in P. Then: (a) $\alpha_{V_T}(P^{*k}) = P^{*k} \cdot kV_T(P)\alpha k \cdot P^{*k}$, (b) $\beta_{V_T}(P^{*k}) = P^{*k} \cdot kV_T(P)\beta k \cdot P^{*k}$, (c) $\gamma_{V_T}(P^{*k}) = P^{*k} \cdot kV_T(P)\gamma k \cdot P^{*k}$.

Proof. For part (a):

$$P^{*k} = \{k\} + P^{*k} \cdot kP$$

Then

$$\alpha_{V_T}(P^{*k}) = \gamma_{V_T}(\{k\}) + \alpha_{V_T}(P^{*k} \cdot kP)$$

But

$$\alpha_{V_T}(P^{*k} \cdot kP) = \alpha_{V_T}(P^{*k}) \cdot kP + P^{*k} \cdot k\alpha_{V_T}(P)$$

Thus

$$\alpha_{V_T}(P^{*k}) = \alpha_{V_T}(P^{*k}) \cdot kP + P^{*k} \cdot k\alpha_{V_T}(P)$$

and since the tree k is not in P, by Lemma 7 the unique solution to this equation is

$$\alpha_{V_T}(P^{*k}) = P^{*k} \cdot k\alpha_{V_T}(P) \cdot kP^{*k}$$

The proofs for parts (b) and (c) are similar. ∎

Theorem 4. Let p be a RLT expression over $V_N \cup V_T$, denoting the regular-like set P, and $V_N = \{k \mid \cdot\, k$ or $*k$ are operators in $p\}$. Then $\alpha_{V_T}(P)$, $\beta_{V_T}(P)$, and $\gamma_{V_T}(P)$ are regular-like.

Proof. The proof is based on Lemmas 9, 10, 11, and 12; and follows by induction on the number of applications of the rules, in Definition 2, used to construct p. ∎

Corollary. Let p be a RLT expression over $V_N \cup V_T$, denoting the regular-like set P of V_1-trees, and $V_N = \{k \mid \cdot\, k$ or $*k$ are operators in $p\}$. Then $\alpha(P)$, $\beta(P)$, and $\gamma(P)$ are regular-like. ∎

REFERENCES

1. A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).
2. W. S. Brainerd, "Tree Generating Regular Systems," *Infor. and Contr.* **14**:217–231 (1969).
3. J. W. Thatcher and J. B. Wright, "Generalized Finite Automata Theory with Application to a Decision Problem of Second Order Logic," *J. Math. System Theory* **2**:57–82 (1968).
4. R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition: An Introduction* (Addison-Wesley, Reading, Mass., 1978).
5. K. S. Fu and B. K. Bhargava, "Tree Systems for Syntactic Pattern Recognition," *IEEE Trans. Comput.* **22**:1087–1099 (1973).
6. M. A. Arbib and Y. Give'On, "Algebra Automata I: Parallel Programming as a Prolegomena to the Categorical Approach," *Infor. and Contr.* **12**:331–345 (1968).
7. M. G. Thomason and R. C. Gonzalez, "Syntactic Recognition of Imperfectly Specified Patterns," *IEEE Trans. Comput.* **C24**:93–95 (1975).