CIS-TR-80-9

A Syntactic Representation of Two-Dimensional Patterns

by

Patrick Shen-pei Wang

Department of Computer and Information Science
University of Oregon
Eugene, Oregon 97403

Abstract - This paper describes some technique for two-dimensional pattern representation. The concept of "regular-like expressions" is introduced and then this structure is used for representing two-dimensional patterns generated by "array grammars". This technique is both sequential and parallel, which provides a compromise between purely sequential methods, which takes too much time for large arrays and purely parallel methods, which take too much hardware for large arrays.

Keywords: regular-like expressions, array grammars, two-dimensional patterns, sequential and parallel technique

1. Introduction

A simple property-specifying device for a class of languages has the definite advantage that each language in the class can be defined explicitly by the device, in contrast to the implicit representation of languages by grammars. For instance, a regular language can be represented by a regular expression. However, no such representation is known for two-dimensional patterns. In this paper, we introduce array grammars and regular-like expressions that describe sets of well-formed formula, which can be interpreted as rectangular arrays. This method employs a sequential/parallel technique, which provides a compromise between purely sequential method, which take too much time for large arrays and purely parallel method, which usually take too much hardware for large arrays. We adapt the definitions and notations from Salomaa [1] and Wang [2,3].

2. Definitions and Notations

Definition 2.1 Let a be a letter, and let L and L_1 be string languages. The a-substitution of L_1 into L, in symbols, LS_aL_1, is defined by

$$LS_aL_1 = \{ P \mid P = P_1Q_1P_2 \ldots P_kQ_kP_{k+1},\ k \geq 0,$$
$$P_1aP_2a \ldots P_ka \in L,\ a \text{ is not a subword of } P_1P_2 \ldots P_{k+1}$$
$$\text{and } Q_j \in L_1,\ \text{for } 1 \leq j \leq k \}.$$

Notice that for each a, S_a is an associative operation. Therefore we may write "products" associatively, for instance, $LS_aL_1S_aL_2S_aL_1$.

The notation $S_a(L^i)$ means the S_a-product with $i \geq 2$ factors L. By definition, $S_a(L^1) = L$. The iterated a-substitution closure of L, in symbols, L^{+a}, consists of all words P which satisfy the following two conditions: (1) For some $i \geq 1$, $P \in S_a(L^i)$ and (2) a is not a subword of P.

The iterated a-substitution closure is closely related to the operations superscript asterisk (*) and plus(+). It is an immediate consequence of the definition that if L is a language over an alphabet not containing

a, then $L^{*a} = (La \cup \Lambda)^{+a}$, $L^{+a} = (La \cup L)^{+a}$.

Definition 2.2 Assume that V_1 and $V_2 = \{\Theta, \textcircled{0}, \cup, +, \emptyset, \Lambda, (,)\}$ are disjoint alphabets, where Θ and $\textcircled{0}$ are row catenation and column catenation operators defined as in [2,3]., $\cup$ means union, $+$ is a specialrepetition operator, $\emptyset$ means empty set, Λ means empty word, (means left parenthesis,) means right parenthesis.

A regular-like expression over $V_1 \cup V_2$ can be defined recursively as follows:

(i) $\emptyset, \Lambda$ and a letter of V_1 are regular-like expressions,

(ii) If $a \in V_1$, Q and R are regular-like expressions, so are $(Q \textcircled{0} R)$, $(Q \Theta R)$, (QUR) and $(Q)^{+a}$,

(iii) Nothing else is a regular-like expression.

Example 2.1 $((a \textcircled{0} c \textcircled{0} b \cup \Lambda)^{+c} \Theta d \cup \Lambda)^{+d}$ is a regular-like expression. (Notice that we have ommitted unnecessary parenthesis). This regular-like expression denotes the language $\{(a^i b^i)_j \mid i, j \geq 0\}$.

Now we are in a position to define array grammars and array languages.

Definition 2.3 An array grammar (AG) is a quadruple $G = (V, T, P, S)$, where $V = V_1 \cup V_2$, where

V_1 is a finite set of nonterminals,

V_2 is a finite set of intermediates,

T is a finite set of terminals,

$P = P_1 \cup P_2$, where

P_1 is a finite set of intermediate rules,

P_2 is a finite set of terminal rules, and

$S \in V_1$ is the start symbol, $(,), \Theta, \textcircled{0} \in V_2$. Detailed explanations are given as follows:

P_1 is a set of ordered pairs (u,v) written as $u \to v$, u,v in $(V_1 \cup V_2)^+$,

v must be a well-formed formula (wff) which is defined as follows :(i) each member in $V_1 \cup V_2 - \{(,),\Theta,\mathbb{0}\}$ is a wff, (ii) if a,b are wffs so are $(a\ \Theta\ b)$, $(a\ \mathbb{0}\ b)$, (iii) nothing else is a wff.

The strings generated by P_1 is $\{x \mid S \overset{*}{\Rightarrow} x \in V_2^*\}$. The intermediate languages generated by P_2 are matrix languages as defined in [2,3] :. For convenience we denote the language generated by P_2 with initial symbol $A \in V_2 - \{(,),\ \Theta,\ \mathbb{0}\}$ by $L_A = \{x \mid A \overset{*}{\Rightarrow} \overset{*}{\Downarrow}\ x \in T^{++}\}$.

Thus we may think of P_2 as a set of dipoles $P_2 = \underset{A \in V_1}{\cup} P(A)$, each dipole cosisting of two sets of rules, the first one of which is for horizontal, sequential productions; the second of which is for vertical, parallel productions.

The language generated by an array grammar is explained as follows:

Starting with S, the rules of P_1 are applied until all nonterminals are replaced with intermediates. Thus we obtain a horizontal string over V_2 (wff). Now each symbol $A \in V_2 - \{(,),\Theta,\mathbb{0}\}$ will be replaced according to to P_2 until every symbol is replaced by a terminal. If each rule in P_2 is (CF:CF), P_1 is CF, then, according to Theorem 11.4 of [1] we see that the collection of all the wffs generated by an array grammar G can be represented by a regular-like expression defined in Definition 2.1 | This expression constitutes the language generated by the array grammar G. It can be interpreted as a set of rectangular arrays (matrix) by properly processing operators such as Θ, $\mathbb{0}$, U and the repetition operator +, subject to the condition imposed by the row and column catenation. (This means that we take only those matrices which are defined under Θ and $\mathbb{0}$.)

3. An Example : Digitized English Letter "R" With Fixed Proportions

In this section we are going to show an example of array grammar which generates a set of digitized English letter "R" with fixed proportion, say 1:1.

Example 3.1 Let $G = (V, T, P, S)$ be an array grammar,

where $V = \{S, S_1\} \cup \{A, B, C, D, E, F, G, H, K, (,), \theta, \emptyset\}$

$T = \{., x\}$

$P_1 = \{ S \to (F \emptyset ((D \theta S_1) \theta E) \emptyset G),$

$S_1 \to (C \emptyset ((A \theta S_1) \emptyset B) \emptyset C),$

$S \to H, \quad S_1 \to K \}$,

$$L_K = \left\{ \begin{matrix} . & . & . \\ x & x & x \\ . & x & . \end{matrix} \right\} = (.\emptyset.\emptyset.) \theta (x\emptyset x\emptyset x) \theta (.\emptyset.\emptyset.)$$

$$L_A = \{ (.)^n.(.)^n \mid n \geq 1 \} = ((.\emptyset a\emptyset.) \cup (.\emptyset.\emptyset.))^{+a}$$

$$L_B = \{ (.)^n x (.)^n \mid n \geq 1 \} = ((.\emptyset a\emptyset.) \cup (.\emptyset x\emptyset.))^{+a}$$

$$L_C = \left\{ \begin{matrix} (.)^n \\ x \\ (.)^n \end{matrix} \mid n \geq 1 \right\} = ((. \theta a \theta .) \cup (. \theta x \theta .))^{+a}$$

$$L_D = \{ x^n \mid n \geq 1 \} = ((x \emptyset a) \cup x)^{+a}$$

$$L_E = \{ (.)^{n-1} x^n \mid n \geq 1 \} = ((. \emptyset a \emptyset x) \cup x)^{+a}$$

$$L_F = \{ (x)_n \mid n \geq 5 \} = ((x \theta a) \cup (x \theta x \theta x \theta x \theta x))^{+a}$$

$$L_G = \left\{ \begin{matrix} (x)_n \\ (.)_{n-2} \\ x \end{matrix} \mid n \geq 2 \right\} = ((x \theta a \theta .) \cup (x \theta x))^{+a} \theta x$$

$$L_H = \left\{ \begin{matrix} x & x & x \\ x & x & x \\ x & x & x \end{matrix} \right\} = (x \emptyset x \emptyset x) \theta (x \emptyset x \emptyset x) \theta (x \emptyset x \emptyset x).$$

Again we omit all unnecessary parenthesis.

It can be seen that

$$L(G)=(L_F \circledcirc ((L_D \ominus ((L_C \circledcirc ((L_A \ominus a) \circledcirc L_B) \circledcirc L_C) \cup L_K)^{+} {}^{a} \ominus L_E) \circledcirc L_G)) \cup L_H$$

which can be interpreted as a set of all digitized English letter "R" with fixed proportion 1:1. The first four members of L(G) are listed in Figure 3.1 . A derivation process is illustrated in Figure 3.2 .

We can also find array grammars that generate two-dimensional patterns involving diagonal strokes as follows:

Example 3.2 . An array grammar which generates a language containing non-vertical, non-horizontal line is shown as follow.

Consider the following (CF:(R:R))AG

$G = (V,I,P,S)$, with $V = \{S,B,C\}$ $I = \{.,X\}$ and

$$P_1 = \left\{ S \rightarrow C \ominus (B \circledcirc S\ B) \ominus C\ , \quad S \rightarrow \begin{matrix} X\ .\ X \\ .\ X\ . \\ X\ .\ X \end{matrix} \right\}$$

$$L_B = \left\{ (.)_n \mid n \geq 3 \right\}$$

$$L_C = \left\{ X(.)^n X \mid n \geq 3 \right\}$$

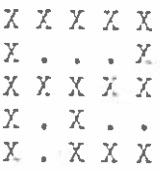
It is seen that $L(G) = \{$set of all digitized "X" with 1:1$\}$.

For instance

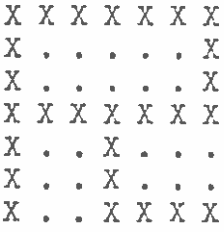
$$\begin{matrix} X\ .\ X \\ .\ X\ . \\ X\ .\ X \end{matrix} \qquad \begin{matrix} X\ .\ .\ .\ X \\ .\ X\ .\ X\ . \\ .\ .\ X\ .\ . \\ .\ X\ .\ X\ . \\ X\ .\ .\ .\ X \end{matrix} \qquad \begin{matrix} X\ .\ .\ .\ .\ .\ X \\ .\ X\ .\ .\ .\ X\ . \\ .\ .\ X\ .\ X\ .\ . \\ .\ .\ .\ X\ .\ .\ . \\ .\ .\ X\ .\ X\ .\ . \\ .\ X\ .\ .\ .\ X\ . \\ X\ .\ .\ .\ .\ .\ X \end{matrix} \qquad \text{etc} \in L(G)$$

It can be seen that $L(G) = ((L_C \ominus (L_B \circledcirc a \circledcirc L_B) \ominus L_C) \cup \begin{matrix} X\ .\ X \\ .\ X\ . \\ X\ .\ X \end{matrix})^{+a}$ |

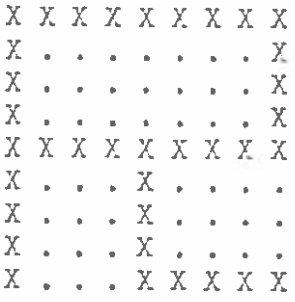
M_1



M_2



M_3



M_4

Figure 3.1 The first four members of L(G).

(1) Replace a by L_K, we obtain

```
. . .
x x x
. x .
```

(2) Replace L_A by . . ., combine with L_K, we have

```
. . .
. . .
x x x
. x .
```

(3) Replace L_B by . x ., combine with above, we have

```
. . .
. . .
x x x
. x .
. x .
```

(4) Replace L_C by $\begin{matrix}\cdot\\ x\\ \cdot\end{matrix}$, combine with the above, we have

```
. . . . .
. . . . .
x x x x x
. . x . .
. . x . .
```

(5) Replace L_D by x x x x x, L_E by $\begin{matrix}x\\ \vdots\\ x\\ \vdots\end{matrix}$, combine with the above result, we have

```
x x x x x
. . . . .
. . . . .
. . x . .
. . x . .
```

(6) Replace L_F by $\begin{matrix}x\\ x\\ x\\ x\\ x\\ x\end{matrix}$ and replace L_G by $\begin{matrix}x\\ \vdots\\ x\\ \vdots\end{matrix}$, combine with above, we have

```
x x x x x x x
x . . . . . x
x . . . . . x
x x x x x x x
x . . x . . .
x . . x . . .
x . . x x x x
```

$\in L(G)$

Figure 3.2 The derivation process of M_3.

4. Discussion

We have just introduced the concepts of "regular-like expressions" and "array grammars". This method for describing two-dimensional patterns employs sequential/parallel techinque, which provides a compromise between purely sequential method, which take too much time for large arrays and purely parallel method, which usually take too much hardware for large arrays. Futhermore, it can generate patterns involving diagonal strokes and patterns with fixed proportions, which is beyond the capabilities of matrix grammars [2,3]. It is the author's hope and belief that in the future more interesting properties of array grammars such as hierarchical structures and closure properties will be found and characterizations of array languages such as recognizers will be established.

References

[1] A. Solomaa, "Formal Languages", Academic Press, 1973.

[2] P.S.P. Wang, "Sequential/Parallel Matrix Array Languages", Journal of Cybernetics, vol. 5.4 (1975), pp 19-36.

[3] P.S.P. Wang "Recognition of Two-Dimensional Patterns", Proceedings of ACM'77 National Conference, Seattle, Washington (1977), pp484-489.

Appendix

For the convenience of the readers, the definition of "matrix grammar" is given as follows.

A matrix over an alphabet I is an mxn rectangular array of symbols from I(m,n≥o). The set of all matrices over I is denoted by I^{**} and $I^{++}=I^{**}-\{\wedge\}$ where $\wedge$ is the empty matrix.

Definition A1 Let $X=\begin{pmatrix}a_{11}\cdots a_{1n}\\ \cdots\\ a_{m1}\cdots a_{mn}\end{pmatrix}$ and $Y=\begin{pmatrix}b_{11}\cdots b_{1n'}\\ \cdots\\ b_{m'1}\cdots b_{m'n'}\end{pmatrix}$

then the column catenation ⊙ is defined only when m=m' and is given by

$$X\odot Y=\begin{pmatrix}a_{11}\cdots a_{1n}b_{11}\cdots b_{1n'}\\ \cdots\\ a_{m1}\cdots a_{mn}b_{m'1}\cdots b_{m'n'}\end{pmatrix}$$

and the row catenation ⊖ is defined only when n=n' and is given by

$$X\ominus Y=\begin{pmatrix}a_{11}\cdots a_{1n}\\ \cdots\\ a_{m1}\cdots a_{mn}\\ b_{11}\cdots b_{1n'}\\ \cdots\\ b_{m'1}\cdots b_{n'n'}\end{pmatrix}$$

Definition A2 Let M and M' be two sets of matrices, the column product (catenation) is defined as $M\odot M'=\{X\odot Y \mid X\in M,\ Y\in M'\}$ and the row product(catenation) is defined as $M\ominus M'=\{X\ominus Y \mid X\in M, Y\in M'\}$.

Notation. I^{*} denotes the set of all horizontal sequences of symbols from I and $I^{+}=I^{*}-\{\varepsilon\}$, where ε is the empty word of I^{*}. I_{*} denotes the set of all vertical sequences of symbols over I and $I_{+}=I_{*}-\{\varepsilon\}$, where ε is the empty word of I_{*}. $x^{i+1}=x^{i}\odot x$, $x_{i+1}=x_{i}\ominus x$, $x\in I^{++}$, i=1,2,... .

Kleene's closure can be defined recursively as follows:

Definition A3 Let M be a set of matrices and $M^1 = M, M^2 = M \otimes M, \ldots,$ $M^{i+1} = M^i \otimes M$, then $M^+ = \bigcup_{i=1}^{\infty} \{M^i\}$, (column +) and $M^* = M^+ \cup \{\Lambda\}$ (column *). Similarly, $M_1 = M, M_2 = M \Theta M, \ldots, M_{i+1} = M_i \Theta M$, then $M_+ = \bigcup_{i=1}^{\infty} \{M_i\}$, (row +), and $M_* = M_+ \cup \{\Lambda\}$ (row *).

Definition A4 A matrix grammar (MG) is a triple $G = (G_1, G_2, M)$, where $G_1 = (V_1, I_1, P_1, S)$ is a phrase structure grammar (PSG or type 0), context-sensitive grammar (CSG or type 1), context-free grammar (CFG or type 2) or regular grammar (RG or type 3), where

V_1 is a finite nonempty set of horizontal nonterminals

I_1 is a finite nonempty set of intermediates, $I_1 = \{S_1, \ldots, S_k\}$

P_1 is a finite nonempty set of type 0(1,2,3) production rules called horizontal rules

$S \in V_1$ is the start symbol

$G_2 = \bigcup_{i=1}^{k} \{G_{2i}\}$, where $G_{2i} = (V_{2i}, I_2, P_{2i}, S_i)$, $i = 1, \ldots, k$ are k type 0(1,2,3) grammars with I_2 a finite nonempty set of terminals, V_{2i} a finite set of vertical nonterminals, S_i the start symbol, and P_{2i} a finite nonempty set of production rules, $V_{2i} \cap V_{2j} = \emptyset$, for $i \neq j$.

$M = \{m_i \mid i = 1, \ldots, p\}$ where each m_i is a kx1 matrix consisting of k rules from the k G_{2i}'s. In each matrix, the length of the both right hand and left hand sides of rules are identical.

The derivations are obtained by first applying the horizontal productions of G_1 until every symbol becomes an intermediate and then applying the vertical productions. In the vertical production whenever a matrix is applied, all of the applicable rules in the matrix should be applied simultaneously as each nonterminal is rewritten. If no such matrix is found, the

derivation halts. In each step only one instance is replaced in each column.

Definition A5 An MG $G = (G_1, G_2, F)$ is called a type $(i:j)$MG if G_1 is a type i and G_2 is a type j grammar for $0 \leq i,j \leq 3$. We use also the notation (CF:CS) for type(1:2)MG and so on. Let α and β be in $(V_1 \cup I_1)^*$. By $\alpha \Rightarrow \beta$ we mean after P_1 is applied[$] to α we get β. Let A,B in $(\bigcup_i^k U_{1i} V_{2i} \cup I_2)^{++}$. By $A \Downarrow B$ we mean after a column matrix is applied to A we get B. Let $\overset{*}{\Rightarrow}$ and $\overset{*}{\Downarrow}$ be the transitive closure of $\Rightarrow$ and $\Downarrow$ respectively. The language generated by G is defined to be

$$L(G) = \left\{ (a_{ij}),\ i=1,\ldots,m\ j=1,\ldots,n \mid S \overset{*}{\underset{G_1}{\Rightarrow}} f \overset{*}{\underset{G_2}{\Downarrow}} (a_{ij}),\ f \in I_1^*,\ a_{ij} \in I_2 \right\}$$