BOUNDED PARALLELISM AND REGULAR LANGUAGES

D. WOOD

Department of Applied Mathematics, McMaster University

CONTENTS

Section 1: Introduction and Overview.

If we examine the rewriting systems of Ibarra[1], the so called Simple Matrix Grammars (SMG), we see that rewriting has the following three facets, namely

(1) rewriting occurs in PARALLEL,

(2) the parallelism is, a priori, BOUNDED, and

(3) the rewriting is, a priori, CONTROLLED.

Secondly, if we examine the rewriting systems of Lindenmayer [61], the so called E0L systems, and compare and contrast with the

(1) Ibarra, O. H., Simple matrix languages, Information and Control 17 (1970), 359-394.

SMG's we find:

(1) rewriting again occurs in PARALLEL,

(2) the parallelism is, a priori, UNBOUNDED and EXHAUSTIVE, and

(3) the rewriting is, a priori, UNCONTROLLED.

Later extensions to E0L systems have lead to two kinds of CONTROL

(a) 'RULE-CONTEXT-FREE' CONTROL - Rozenberg's Tabled Systems [81]; rules are applied EXHAUSTIVELY from one table of rules from a given set of tables. However, within this restriction a particular rule is applied independently of the other rules that are applied at the same time.

(b) 'RULE-CONTEXT' CONTROL - [11]. The application of a particular rule is dependent upon its context within the sequence of rules applied at a particular time.

In the light of these developments in E0L systems, simple matrix grammars can be considered to have either 'RULE-CONTEXT' or 'RULE-CONTEXT-FREE' CONTROL. In the following we present a survey of results on the effect of changing facet (3) for SMG to:

(3) the rewriting is UNCONTROLLED.

This is the investigation of E0L-like systems where the parallelism is bounded, and can be considered to be the investigation of the development of filamentous organisms under an environment which deprives the cells of "food". We consider two different kind of generating systems, finite state generators and right linear grammars, which we generalise to give n-parallel finite state generators and n-parallel right linear grammars.

Section 2: n-parallel finite state generators.

Definition

A *finite state generator* (FSG) G, is a quintuple (N,T,E,S,F) where

N is a finite set of *points*,

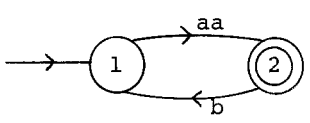
T is a terminal *alphabet*,

E is a set of *edges*, E VxT*xV,

S N is a set of *entry* points, and

F N is a set of *exit* points.

Example 1



N = {1,2}, T = {a,b},

E = {(1,aa,2), (2,b,1)}

S = {1} and F = {2}.

Definition

Given G, an FSG, then we write $u \to vx$ if u,v in N, x in T* and (u,x,v) in E.

Similarly, we write $u \to^{i} vx$, i>0 if there exist sequences $u_0,\ldots,u_i$ and $x_1,\ldots,x_i$ such that

$u_j \to u_{j+1} x_{j+1}$, $0 \leq j < i$,

$x = x_1 \ldots x_i$, $u = u_0$ and $v = u_i$.

We write $u \to^{+} vx$ if there exists i>0 such that $u \to^{i} vx$, and we write $u \to^{*} vx$ if either $u \to^{+} vx$ or u = v and x = ε. The *language* generated by an FSG G is

$L(G) = \{x: \ u \to^{*} vx, \ u \text{ in } S, \ v \text{ in } F\}$.

We say $L \subseteq T^{*}$ is a *finite state language* (FSL) iff there exists an FSG G such that L = L(G). We usually say L is a *regular set*, so we denote the family of FSL's by $\mathcal{R}$.

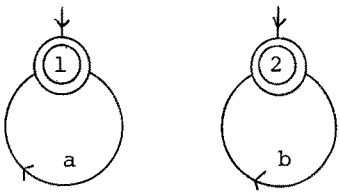
Example 1 (cont.)

$L(G) = \{aa, aabaa, aabaabaa, \ldots\}$

$= \{aa(baa)^{i}: \ i \geq 0\}$.

We can now consider n-parallel FSGs. The basic idea is to have n FSG's operating synchronously and in parallel. Rather than repeating a generalised form of the notation above, we develop the model informally.

Consider a 2-PFSG (2-parallel FSG) G:



Initially these two processors (or generators) (G_1 and G_2) are both *alive* and *idle*. We start them up simultaneously and synchronously (by which we mean we move along edges at the same time).

Thus we have:

point reached and word generated so far:

	G_1	G_2
time = 0	1	2
= 1	1a	2b
= 2	1aa	2bb
= 3	1aaa	2bbb
...	...	...

If the two processors each reach an exit point at the same time, then we catenate the two words generated so far. Here we have, since both points 1 and 2 are exit points that:

	word generated by G so far
time = 0	ε
1	ab
2	a^2b^2
3	a^3b^3
...	...

Thus, we say $L(G) = \{a^ib^i: \ i \geq 0\}$.

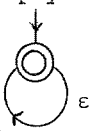
In a similar way we can generate languages given by an n-PFSG, $n \geq 1$, called n-PFSL's. Let $\mathcal{L}_n$ denote the family of n-PFSL's.

We have:

Result 1: $\mathcal{R} = \mathcal{L}_1$, trivially.

Result 2: For $n \geq 1$, $\mathcal{L}_n \subseteq \mathcal{L}_{n+1}$.

Proof: Simply add an extra processor G_{n+1} to G, which generates nothing but the empty word, giving an (n+1)-PFSG, i.e.



Result 3: ε-edges do add generating power.

Proof: Consider $L = \{a^i b^j : 0 \leq i \leq j\}$, L cannot be generated without an ε-edge occurring in at least one processor.

Result 4: For $n \geq 2$, $L_n = \{a_1^i \ldots a_n^i : i \geq 0\}$ is in $\mathcal{L}_n$ but not in $\mathcal{L}_{n-1}$.

Corollary: For $n \geq 1$, $\mathcal{L}_n \subsetneq \mathcal{L}_{n+1}$: an infinite hierarchy of languages.

Result 5: $\mathcal{L}_2 \subsetneq$ family of one-counter languages.

Result 6: For $n \geq 1$, $\mathcal{L}_n \subsetneq$ family of context-sensitive languages.

Result 7: For $n \geq 3$, $\mathcal{L}_n$ and the family of context-free languages are incomparable.

These results will be found in Wood(2).

Section 3: Variations on the basic model of n-PFSG's.

In the previous section, the terminology *alive* and *idle* with respect to n-PFSG's was introduced. If we say that an FSG *dies* when it reaches an exit point, then a word is output by an n-PFSG whenever all n processors die at the same time. We also say a processor is *active* when it is alive and not idle.

Variation 1: Given m FSGs, $m>0$ and n, $0<n<m$ then allow n live processors (at most) to be active at one time, and no longer require that n processors die together. A word is generated by $G_1, \ldots, G_m$ if initially, they become alive simultaneously and n of them become active for one time step, then another n become active for the next

(2) Wood, D., Properties of n-parallel finite state languages, Utilitas Mathematica 4 (1973), 103-113.

time step, and so on until all m processors are dead. Then we catenate the m words together left to right, to give a word generated by $G_1,\ldots,G_m$. Let $\mathcal{L}_{(m,n)}$ be the family of languages defined by this model. Then we have:

Result 8: $\mathcal{R} = \mathcal{L}_{(m,n)}$ for all m,n, $0<n<m$. That is the model degenerates.

Variation 2: Let the m processors proceeding from left to right become active n at a time (if there are less than n, then the remaining number of processors), and all n die simultaneously. Let $\mathcal{L}_{m,n}$ denote the corresponding family of languages.

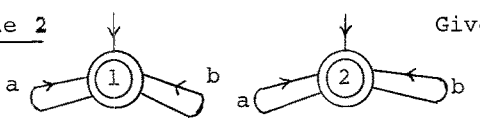
Result 9: For $m>0$, $0<n\leq m$, where $m = np+q$, $0\leq q<n$, then $\mathcal{L}_{m,n} = (\mathcal{L}_n)^p \mathcal{L}_q$.

Result 10: $\mathcal{L}_{m,n} \subsetneqq \mathcal{L}_{m+1,n}$, for all m,n, $0<n\leq m$.

Result 11: $\mathcal{L}_{m,n}$ and $\mathcal{L}_{m,n+1}$ are incomparable, for all m,n, $0<n\leq m$. Note that $\mathcal{L}_{m,m} = \mathcal{L}_m$ and $\mathcal{L}_{m,1} = \mathcal{R}$, for all $m>0$.

Variation 3: We can impose more structure on the n-PFSG by introducing a traffic cop, who by making use of a book of regulations determines which edges are allowable edges for the n FSG's at this time instant.

Example 2 Given the 2-PFSG G:



with the rule book $\{<(1,a,1),(2,a,2)>, <(1,b,1),(2,b,2)>\}$ then processor G_1 can only traverse the a-edge whenever processor G_2 traverses an a-edge, thus $L(G) = \{ww:\ w \text{ in } \{a,b\}^*\}$. The rule book is usually called a *control set*, the corresponding family of languages is denoted $\mathcal{L}_n^C$.

Result 12: $\mathcal{L}_n \subsetneqq \mathcal{L}_n^C$, for all $n>1$.

Result 13: $\mathcal{L}_n^C = \mathcal{R}_{[n]}$, the family of *n-right linear SML*(3).

(3) see footnote 1.

These results can be found in Wood[4]

Section 4: An alternative formulation, n-PRLG's and closure properties.

Definition

For $n \geq 1$, let $G = (N_1,\ldots,N_n,T,S,P)$ where N_i, $1 \leq i \leq n$, are disjoint nonterminal alphabets,

T is a terminal alphabet,

S is a sentence symbol, S not in $\bigcup_{i=1}^{n} N_i = N$,

P is a finite set of rules of the form:

(i) $S \rightarrow X_1 \ldots X_n$, X_i in N_i,

(ii) $X \rightarrow aY$, X,Y in N_i, some i, a in T^*, and

(iii) $X \rightarrow a$, X in N_i, some i, a in T^*.

G is an *n-parallel right linear grammar, n-PRLG.*

We write $x \Rightarrow y$ iff

either $x = S$ and $S \rightarrow y$ in P,

or $x = y_1X_1 \ldots y_nX_n$, $y = y_1x_1 \ldots y_nx_n$, y_i in T^*, X_i in N_i, x_i in $T^* \cup T^*N_i$ and $X_i \rightarrow x_i$ in P, $1 \leq i \leq n$.

In the usual way we obtain $x \overset{+}{\Rightarrow} y$ and $x \overset{*}{\Rightarrow} y$, notice that either a valid sentential form, other than S itself, has either no nonterminals or exactly n nonterminals. $L(G) = \{x: S \overset{+}{\Rightarrow} x, x \text{ in } T^*\}$, and $L \subseteq T^*$ is an n-PRLL iff there exists an n-PRLG G such that $L = L(G)$. Let the family of n-PRLL's be $\mathscr{L}_n$.

Result 14: $\mathscr{L}_1 = \mathscr{L}$.

Result 15: $\mathscr{L}_n \not\subseteq \mathscr{L}_n$, for all $n>1$.

Result 16: Closure and non-closure results for $\mathscr{L}_n$ and $\mathscr{L}_n$, $n>1$.

---

(4) Wood, D., Two variations on n-parallel finite state generators, McMaster University CS TR 73/3 (1973).

operation	$\mathcal{J}_n$	$\mathcal{R}_n$
union	No	Yes
homomorphism	No	Yes
finite substitution	No	Yes
substitution	No	No
catenation	No	No
intersection	No	No
complementation	No	No
intersection with regular set	No	Yes
a-NGSM maps (accepting states)	No	Yes

These results are detailed in Rosebrugh and Wood[5] and Wood[6].

Section 5: Characterisation Theorems for $\mathcal{J}_{||}$ and $\mathcal{R}_{||}$.

Definition

Let $\mathcal{J}_{||} = \bigcup_{i=1}^{\infty} \mathcal{J}_i$ and $\mathcal{R}_{||} = \bigcup_{i=1}^{\infty} \mathcal{R}_i$ and <M> denote an *infinite sequence* $M_1, M_2, \ldots$ where $M_i \subseteq T^*$. Define $L(M) = \{x: x \text{ in } M_i, \text{ some } i \geq 1\}$. A sequence <M> is a *regular sequence* iff there exists an FSG G such that $L(M) = L(G)$ and $M_i = \{x: u \vdash^i vx, u \text{ in } S, v \text{ in } F\}$.

Given two sequences $<M_1>$ and $<M_2>$ define $<M_1> \odot <M_2>$, the *synchronised product* of $<M_1>$ with $<M_2>$, as the sequence $M_{11}M_{21}, M_{12}M_{22}, \ldots$. Define $<M_1> \oplus <M_2>$, the *synchronised union* of $<M_1>$ with $<M_2>$, as the sequence, $M_{11} \cup M_{21}, M_{12} \cup M_{22}, \ldots$

Result 17: For $n>0$, L in $\mathcal{R}_n$ iff there exist $m \times n$ regular sequences $<M_{ij}>$, $1 \leq i \leq m$, $1 \leq j \leq n$ for some $m \geq 1$ such that

(5) Rosebrugh, R. D., and Wood, D., Restricted parallelism and right linear grammars, McMaster University CS TR 72/6 (1972).
(6) See footnote 2.

$L = L(M)$ where $\langle M \rangle = \langle M_{11} \rangle \odot \langle M_{12} \rangle \odot \ldots \langle M_{1n} \rangle$

$\oplus \langle M_{21} \rangle \ldots .$

$\vdots$

$\oplus \langle M_{m1} \rangle \odot \ldots \odot \langle M_{mn} \rangle$

Result 18: For $n>0$, L in $\mathcal{F}_n$ iff there exist n regular sequences $\langle M_i \rangle$ such that $L = L(M)$ where $\langle M \rangle = \langle M_1 \rangle \odot \ldots \odot \langle M_n \rangle$.

Definition

Let $\mathcal{S}$ be the smallest family of sequences containing the regular sequences and closed under $\odot$ and $\oplus$ and let $L(\mathcal{S}) = \{L(M): \langle M \rangle \text{ in } \mathcal{S}\}$.

Result 19: Sequence Characterisation of $\mathcal{R}_{||}$.

$\mathcal{R}_{||} = L(\mathcal{S})$.

Definition

Let $\mathcal{J}$ be the smallest family of sequences containing the regular sequences and closed under $\odot$.

Result 20: Sequence Characterisation of $\mathcal{F}_{||}$.

$L(\mathcal{J}) = \mathcal{F}_{||}$

Definition

An a-NGSM is a *nondeterministic generalised sequential machine with accepting states*. Let $L_n = \{a_1^i \ldots a_n^i : i \geq 0\}$.

Result 21: Image Theorem for $\mathcal{R}_n$ and $\mathcal{F}_n$

For all L in $\mathcal{R}_n$, $n>0$ there exists an a-NGSM M such that $L = M(L_n)$. Since $\mathcal{F}_n \nsubseteq \mathcal{R}_n$ the result holds for $\mathcal{F}_n$.

Result 22: Language Characterisation of $\mathcal{R}_n$.

For $n>0$, $\mathcal{R}_n$ is the smallest family of languages containing L_n and closed under union and a-NGSM maps.

For further details see Wood[7] and Rosebrugh and Wood[8].

---

(7) See footnote 2.

(8) Rosebrugh, R. D., and Wood, D., A characterization theorem for n-parallel right linear languages, Journal of Computer and System Sciences 7 (1973), 579-582.

Section 6: Concluding Remarks.

We close by first posing two open problems and secondly introducing some possibilities for future research.

Open problem 1: Is the equivalence of two n-PRLG's decidable or undecidable?

Open problem 2: Prove $\mathcal{R}_{||}$ is not closed under intersection. It is easy to construct examples of languages which are formed by the intersection of two n-PRLL, which intuitively are not in $\mathcal{R}_{||}$. But some new (or adapted) proof techniques are needed.

Future research possibilities

(1) Can generalise n-PRLG (or n-RLSMG) in the same way that Salomaa [103] has generalised E0L systems.

(2) Rewriting in E0L systems has been considered as a one-state NGSM map, Salomaa [103] has considered one extension of this notion, however it can be extended in another way by allowing more than one state in the NGSM. This extension can then be applied to n-PRLG (n-PFSG and n-RLSMG).