ON THE ACTIVE USE OF MEMORY IN RIGHT-BOUNDARY GRAMMARS AND PUSH-DOWN AUTOMATA

by
A. Ehrenfeucht, H. J. Hoogeboom and G. Rozenberg

CU-CS-298-86 September, 1985

University of Colorado, Department of Computer Science, Boulder, Colorado.

ON THE ACTIVE USE OF MEMORY IN RIGHT-BOUNDARY GRAMMARS AND PUSH-DOWN AUTOMATA

by

A. Ehrenfeucht*, H.J. Hoogeboom** and G. Rozenberg***

All correspondence to the third author.

*University of Colorado, Department of Computer Science, Boulder, Colorado

**Institute of Applied Mathematics and Computer Science, University of Leiden, Leiden, The Netherlands.

***Institute of Applied Mathematics and Computer Science, University of Leiden, Leiden, The Netherlands and University of Colorado, Department of Computer Science, Boulder, Colorado.

ABSTRACT

A *coordinated pair system (cp system* for short) consists of a pair of grammars the first of which is right-linear *(rl)* and the second right-boundary *(rb)*. A *right-boundary grammar* is like a right-linear grammar except that one does not distinguish between terminal and nonterminal symbols - still the rewriting is applied to the last symbol of a string only (and erasing productions are allowed). A rewriting in a cp system consists of a pair of rewritings: one in the first and one in the second grammar - such a rewriting is possible if the pair of productions involved is in the finite set of *rewrites* given with the system. It is easily seen that cp systems correspond very closely to (are another formulation of) push-down automata: the right-linear component models the input and the finite state control while the rb component models the push-down store.

A rb grammar G transforms (rewrites) strings which are stored in a one-way (potentially infinite) tape. If one observes during a derivation δ the use of a fixed n-th cell of the tape and one notes the symbol stored there each time that (the contents of) the cell is rewritten, then one gets the *n-active record of δ;* the set of all n-active records for all successful derivations δ forms the n-active language of G, denoted $ACT_n(G)$. It is proved that for each rb grammar G and each $n \in \mathbf{N}^+$, $ACT_n(G)$ is regular and moreover, for each $M \subseteq \mathbf{N}^+$, $\bigcup_{n \in M} ACT_n(G)$ is regular.

Then we provide a representation theorem allowing one to represent a cp system through a finite number of rb grammars and using this theorem we transfer the above results on the "active use of memory" to cp systems.

INTRODUCTION.

The literature is full of various notions of machines (automata) and grammars each one developed with a specific, practical or theoretical, motivation behind it (see, e.g., [H] and [S]). The notion of an *ects system* provides a common framework for quite a variety of these models (see [R]). Within the ects model various notions of *machines and grammars* are considered as systems of basic units (which are rather simple rewriting systems working together in a "coordinated fashion"). It is demonstrated in [R] that *right-boundary grammars (rb grammars* for short) constitute such a basic (perhaps the most basic) unit. A right-boundary grammar is like a *right-linear grammar* except that one does not distinguish between terminal and nonterminal symbols - still the rewriting is applied to the last symbol of a string only (and erasing productions are allowed); the notion of a rb grammar is a special case of the *regular canonical system* of Buchi (see [B]). A well-known subclass of ects systems are *coordinated pair systems (cp systems* for short). A cp system consists of two grammars the first of which is right-linear and the second is right-boundary; it turns out that cp systems correspond very closely to (are another formulation of) *push-down automata.* The theory of cp systems (or: the cp system approach to the theory of push-down automata) is presented in [EHR1], [EHR2], [EHR3] and [EHR4].

This paper continues the research on the theory of cp systems and in particular it presents results describing the use of memory in right-boundary grammars (and cp systems). The basic idea investigated in the paper is as follows.

A right-boundary grammar G represents (transformations of) a data structure which is a linear one-way (potentially) infinite array of (memory) cells the processing of which takes place at the (right) end of the array. Hence during each derivation in G one can record the history of the use (the "scheduling") of each single cell. In other words each time (the contents of) a given cell is

rewritten a note is made of the letter being stored there at that time (the *active* letter at this moment) and the sequence of all such "notes of activity" during a given derivation δ forms the *active record* of this cell during the derivation δ. The set of all active records of the n-th memory cell in all successful derivations forms the *n-active language of G*, denoted $ACT_n(G)$ (a derivation is *successful* if it leads from the axiom of G to the empty word).

We prove that for each rb grammar and each n, $ACT_n(G)$ is regular (Corollary 1.3). Actually this regularity is quite "deep"; it turns out that for an arbitrary subset M of positive integers $\bigcup_{m \in M} ACT_m(G)$ is regular (Theorem 3.4) - this is strong regularity indeed!!

In order to transfer these results to cp systems we prove a *rb representation theorem* for cp systems (Theorems 5.1 and 5.2): rather than to consider a cp system one can consider a finite number of rb grammars. This representation theorem allows us to prove the above mentioned result about strong regularity of active languages also for cp systems (Theorem 5.3).

0. PRELIMINARIES

We assume the reader to be familiar with basic formal language theory (see, e.g., [H] or [S]).

For a set Z, $\#Z$ denotes its cardinality. If V is a finite set of integers we use $\max V$ and $\min V$ to denote the maximal and the minimal element of V respectively.

For a word x, $|x|$ denotes its length and, if $1 \leq k \leq |x|$, then $x(k)$ denotes the k-th letter of x. If x is nonempty, then we use $last(x)$ to denote $x(|x|)$. Λ denotes the empty word.

A letter to letter homomorphism is called a *coding* and a homomorphism that maps each letter either into a letter or into the empty word is called a *weak coding*.

A *context-free grammar*, abbreviated *cf grammar*, is specified in the form $G = (\Sigma, P, S, \Delta)$, where Σ is its alphabet, P its set of productions, $S \in \Sigma$ its axiom and Δ its terminal alphabet. For $x, y \in \Sigma^*$ we write $x \underset{G}{\overset{\pi}{\Longrightarrow}} y$ if x directly derives y using production π.

A *right-linear grammar*, abbreviated *rl grammar*, is a context-free grammar $G = (\Sigma, P, S, \Delta)$ which has its productions in the set $(\Sigma - \Delta) \times \Delta^* ((\Sigma - \Delta) \cup \{\Lambda\})$.

1. RIGHT-BOUNDARY GRAMMARS AND THEIR ACTIVE RECORDS

In this section we introduce basic notions concerning right-boundary grammars and then we formalize the (active) use of memory by derivations in these grammars.

Definition 1.1. A *right-boundary grammar*, abbreviated *rb grammar*, is a triple $G = (\Sigma, P, \omega)$, where

Σ is an *alphabet*,

$P \subseteq \Sigma \times \Sigma^*$ is a finite set of *productions*, and

$\omega \in \Sigma^+$ is the *axiom* of G. ■

For a rb grammar $G = (\Sigma, P, \omega)$ we use $maxr(G)$ to denote $\max\{|w| \mid A \to w \in P\}$.

Definition 1.2. Let $G = (\Sigma, P, \omega)$ be a rb grammar.

(1) Let $x, y \in \Sigma^*$ and let $\pi = A \to w \in P$. x *directly derives* y *in* G *(using* π*)*, written $x \underset{G}{\Longrightarrow} y$ $(x \overset{\pi}{\underset{G}{\Longrightarrow}} y)$, if $x = zA$ and $y = zw$ for some $z \in \Sigma^*$.

Let $\overset{*}{\underset{G}{\Longrightarrow}}$ be the reflexive and transitive closure of $\underset{G}{\Longrightarrow}$. If $x \overset{*}{\underset{G}{\Longrightarrow}} y$, then we say that x *derives* y *in* G.

(2) A *derivation (in* G*)* is a sequence $\delta = (x_0, x_1, \ldots, x_n)$, $n \geq 0$, of words from Σ^* such that, for every $1 \leq i \leq n$, $x_{i-1} \underset{G}{\Longrightarrow} x_i$. We say that δ *derives* x_n *from* x_0 and denote it by δ: $x_0 \overset{*}{\underset{G}{\Longrightarrow}} x_n$.

For $0 \leq i \leq n$, x_i is called the *i-th line of* δ and is denoted by $\delta(i)$. n is called the *length of* δ *and is denoted* by $lg(\delta)$.

(3) A derivation $\delta : \omega \overset{*}{\underset{G}{\Longrightarrow}} \Lambda$ is called *successful*.

(4) Let $\delta_1 = (\delta_1(0), \delta_1(1), \ldots, \delta_1(m))$ and $\delta_2 = (\delta_2(0), \delta_2(1), \ldots, \delta_2(n))$ be two

derivations in G such that $\delta_1(m) = \delta_2(0)$. The *composition of* δ_1 *and* δ_2, denoted $\delta_1 \otimes \delta_2$, is the derivation $(\delta_1(0), \delta_1(1), \ldots, \delta_1(m), \delta_2(1), \ldots, \delta_2(n))$. ■

Lemma 1.1. Let $G = (\Sigma, P, \omega)$ be a rb grammar and let $x, y \in \Sigma^+$. If $x \underset{G}{\Longrightarrow} y$, then there exists a unique production $\pi \in P$ such that $x \overset{\pi}{\underset{G}{\Longrightarrow}} y$. ■

Definition 1.3. Let G be a rb grammar and let $\delta = (\delta(0), \ldots, \delta(n))$, $n \geq 0$ be a derivation in G. The sequence $(\pi_1, \ldots, \pi_n)$ of productions such that $\delta(i-1) \overset{\pi_i}{\underset{G}{\Longrightarrow}} \delta(i)$ for every $1 \leq i \leq n$ is called the *control sequence of* δ and is denoted by $cont(\delta)$; if $n = 0$, then $cont(\delta)$ is the empty sequence. ■

Remark 1.1. (1) Lemma 1.1 guarantees the uniqueness of the control sequence (for each derivation δ).

(2) Note that if $\delta_1 : u \overset{*}{\underset{G}{\Longrightarrow}} v$ and $\delta_2 : v \overset{*}{\underset{G}{\Longrightarrow}} w$ are two derivations in a rb grammar G, then

$lg(\delta_1 \otimes \delta_2) = lg(\delta_1) + lg(\delta_2)$ and

$cont(\delta_1 \otimes \delta_2) = cont(\delta_1)cont(\delta_2)$. ■

In order to simplify our notation we will skip the inscription "G" whenever G is understood from the context. Hence, e.g., we will write $\Longrightarrow$ and $\overset{*}{\Longrightarrow}$ rather than $\underset{G}{\Longrightarrow}$ and $\overset{*}{\underset{G}{\Longrightarrow}}$ respectively.

If all the lines of a derivation δ in a rb grammar G are written under each other (adjusted letter-by-letter), then the most natural way of storing δ in a memory suggests by itself that all the first letters of the lines of δ are stored in the first memory cell, all the second letters are stored in the second memory cell, etc.

This can be depicted as follows:

Figure 1.

Now if one wants to get an idea of the use of memory within this particular derivation δ, then one can choose $n \geq 1$ and then observe the actions performed on the n-th memory cell. The significant moments are those when this particular memory cell becomes *active* (i.e. the symbol stored there is rewritten).

This natural intuition of the memory use associated with a derivation in a right-boundary grammar leads us to the following definition.

Definition 1.4. Let $\delta = (\delta(0), \ldots, \delta(k))$ be a derivation in a rb grammar G and let $n \in \mathbf{N}^+$.

(1) A line $\delta(i)$ of δ with $|\delta(i)| = n$ is called an *n-active* line of δ.

(2) The *n-active record of* δ, denoted $act_n(\delta)$, is the word $\varphi_n(\delta(0))\varphi_n(\delta(1)) \cdots \varphi_n(\delta(k-1))$, where $\varphi_n : \Sigma^* \to \Sigma \cup \{\Lambda\}$ is the mapping defined by

$$\varphi_n(u) = \begin{cases} u(n), \text{ if } |u| = n, \\ \Lambda \quad , \text{ otherwise.} \end{cases} \quad \blacksquare$$

Remark 1.2. (1) Note that in determining the n-active record of δ the last word of δ is not taken into account. Therefore:

(2) If $\delta_1 : u \overset{*}{\Longrightarrow} v$, $\delta_2 : v \overset{*}{\Longrightarrow} w$ are two derivations in a rb grammar G, then $act_n(\delta_1 \otimes \delta_2) = act_n(\delta_1)act_n(\delta_2)$. ■

Example 1.1. Let $G = (\{A, B, C\}, P, A)$ be the rb grammar with $P = \{A \to BC, B \to AB, B \to \Lambda, C \to BB\}$.

Then $\delta = (BBB, BB, BAB, BAAB, BAA, BABC, BABBB, BABB, BAB)$ is a derivation in G of length 8 with

$cont(\delta) = (B \to \Lambda, B \to AB, B \to AB, B \to \Lambda, A \to BC, C \to BB, B \to \Lambda, B \to \Lambda)$,

$act_1(\delta) = \Lambda$, $act_2(\delta) = B$, $act_3(\delta) = BBA$, $act_4(\delta) = BCB$, $act_5(\delta) = B$ and $act_n(\delta) = \Lambda$ for each $n \geq 6$.

This is easily seen if we write the consecutive lines of δ under each other.

Figure 2.

■

If we record now, for an $n \geq 1$, the n-active records for *all* successful derivations in a rb grammar G, then we get a complete picture of the use of the n-th memory cell (from our "intuitive model") in G.

Definition 1.5. Let $G = (\Sigma, P, \omega)$ be a rb grammar and let $n \in \mathbf{N}^+$. The *n-active language of G*, denoted $ACT_n(G)$, is the language $\{act_n(\delta) \mid \delta : \omega \overset{*}{\underset{G}{\Longrightarrow}} \Lambda\}$. ■

We demonstrate now that for each n, $ACT_n(G)$ is regular, which intuitively means that this "schedule of active use" of each memory cell may be realized (implemented) by a finite automaton. Actually we can prove a somewhat stronger result.

Theorem 1.2. Let $G = (\Sigma, P, \omega)$ be a rb grammar and let $w_1, w_2 \in \Sigma^*$. Then $\{act_n(\delta) \mid \delta : w_1 \overset{*}{\underset{G}{\Longrightarrow}} w_2\}$ is regular for each $n \in \mathbf{N}^+$.

Proof.

Let $n \in \mathbf{N}^+$. We construct a finite automaton $\mathbf{A}_n = (Q_n, \Sigma, \Pi_n, I_n, F_n)$ as follows.

$\mathbf{A}_n$ has as its set of states the set of words of length n over Σ together with w_2 if $|w_2| \neq n$; i.e., $Q_n = \Sigma^n \cup \{w_2\}$.

For $u \in \Sigma^n$, $v \in Q_n$ and $A \in \Sigma$ there exists an edge $(u, A, v) \in \Pi_n$ if and only if

there exists a derivation $\delta : u \overset{*}{\underset{G}{\Longrightarrow}} v$ such that $act_n(\delta) = A$. (Note that then $A = u(n)$.) If $|w_2| \neq n$, then w_2 has no outgoing edges.

I_n consists of those $u \in Q_n$ for which there exists a derivation $\delta : w_1 \overset{*}{\Longrightarrow} u$ such that $act_n(\delta) = \Lambda$. (Note that if $|w_1| = n$, then $I_n = \{w_1\}$.)

$F_n = \{w_2\}$.

$L(\mathbf{A}_n)$ is regular and obviously $L(\mathbf{A}_n) = \{act_n(\delta) \mid \delta : w_1 \overset{*}{\underset{G}{\Longrightarrow}} w_2\}$. ∎

Remark 1.3. The effectiveness of the construction of the automaton $\mathbf{A}_n$ from the above proof relies on deciding whether or not, for $u, v \in \Sigma^n$, there exists a derivation $\delta : u \overset{*}{\underset{G}{\Longrightarrow}} v$ with $act_n(\delta) = A$, $A = u(n)$. It is obvious that such a derivation cannot have lines of length n other than its first (we mean $\delta(0)$) and last lines. Moreover it is easily seen that, for all p, j such that $p \leq j < lg(\delta)$, $|\delta(p)| > n$ implies $|\delta(j)| > n$. Hence it follows that we may write $\delta = \delta_1 \otimes \delta_2$ where $\delta_1 : u \overset{*}{\Longrightarrow} w$, $\delta_2 : w \overset{*}{\Longrightarrow} v$ are such that $|\delta_1(i)| < n$ for $0 < i < lg(\delta_1)$ and $|\delta_2(i)| > n$ for $0 \leq i < lg(\delta_2)$. Note that if δ contains no lines shorter (longer) than n, then $lg(\delta_1) = 1 (lg(\delta_2) = 0$ respectively). All lines in δ_2 have the first n symbols in common since they are not rewritten during δ_2. Thus $w = vz$ for some $z \in \Sigma^*$, $0 \leq |z| < maxr(G)$ and there exists a derivation $\mu : z \overset{*}{\Longrightarrow} \Lambda$ with $cont(\mu) = cont(\delta_2)$.

In this way the problem of deciding whether or not $(u, A, v) \in \Pi_n$ is reduced to the problem of deciding whether or not $z \overset{*}{\underset{G}{\Longrightarrow}} \Lambda$ for a given word z. The latter problem is easily seen to be decidable. The reasoning as above can be extended in a straightforward way to the problem of deciding whether or not $u_1 \overset{*}{\underset{G}{\Longrightarrow}} u_2$ for arbitrary $u_1, u_2 \in \Sigma^*$. (If $u_2 \neq \Lambda$, then take $n = |u_2|$; every deriva-

tion $\delta : u_1 \overset{*}{\underset{G}{\Longrightarrow}} u_2$ can be decomposed into an initial part leading to the first line of length n and a number of derivations of the form discussed above.) Thus the construction of the automaton $\mathbf{A}_n$ is effective. ■

Corollary 1.3. Let G be a rb grammar and let $n \in \mathbf{N}^+$. Then $ACT_n(G)$ is regular. ■

Example 1.1. (continued) A finite automaton accepting $ACT_3(G)$ can easily be constructed using the following diagram. This diagram has "virtual"nodes to represent derivations with lines of length less than 3. Note that $X \overset{*}{\underset{G}{\Longrightarrow}} \Lambda$ for every $X \in \Sigma$.

Figure 3.

The finite automaton for $ACT_3(G)$ looks as follows.

Figure 4.

■

2. SPACE USED BY DERIVATIONS IN RB GRAMMARS

We have learned in the last section that for an arbitrary rb grammar $ACT_n(G)$ is regular for each n. As a matter of fact we are going to prove (in Section 3) that the regularity of the use of memory cells in rb grammars is much deeper than that: it turns out that for an arbitrary subset M of $\mathbf{N}^+$ the union $\bigcup_{m \in M} ACT_m(G)$ is also regular - this is strong regularity indeed !

In this section we prove an auxiliary technical result (Lemma 2.1) that is interesting on its own: in deriving a word v from a word u in a rb grammar it suffices to use (in addition to the space occupied by u and v) no more than some constant (for the grammar) extra amount of space.

The amount of space used by a derivation is formalized as follows.

Definition 2.1. Let $G = (\Sigma, P, \omega)$ be a rb grammar.

(1) The *breadth* of a derivation δ in G, denoted $brd(\delta)$, is

$\max\{|\delta(i)| \mid 0 \le i \le lg(\delta)\}$.

(2) Let $u \overset{*}{\underset{G}{\Longrightarrow}} v$ for some $u, v \in \Sigma^*$. The (u,v)*−breadth*, denoted $brd(u,v)$, equals $\min\{brd(\delta) \mid \delta : u \overset{*}{\underset{G}{\Longrightarrow}} v\}$. ■

Lemma 2.1. Let $G = (\Sigma, P, \omega)$ be a rb grammar. There exists an integer m_G such that for each pair $u, v \in \Sigma^*$, $u \overset{*}{\underset{G}{\Longrightarrow}} v$ implies that

$brd(u,v) \le m_G + \max\{|u|, |v|\}$.

Proof.

Let $m_G = \max\{brd(w_1, w_2) \mid w_1 \overset{*}{\underset{G}{\Longrightarrow}} w_2 \text{ and } |w_1|, |w_2| \le maxr(G)\}$. Note that $maxr(G) \le m_G$.

Let $\delta : u \overset{*}{\Longrightarrow} v$ be an arbitrary derivation in G. We will prove that there exists a

derivation $\delta' : u \overset{*}{\Longrightarrow} v$ in G, with $brd(\delta') \leq m_G + M$, where $M = \max\{|u|, |v|\}$.

A line $\delta(i)$ of δ is called *special* if $M < |\delta(i)| \leq M + maxr(G)$. Let $\tau = (\delta(i_1), \delta(i_2), \ldots, \delta(i_t))$ be the subsequence of δ consisting of all special lines. Since $|\delta(j-1)| - 1 \leq |\delta(j)| < |\delta(j-1)| + maxr(G)$ for each $1 \leq j \leq lg(\delta)$, δ cannot have lines longer than $M + maxr(G)$ without having special lines. So if τ is empty then the lemma holds.

Assume now that τ is nonempty. Both $\delta_0 = (\delta(0), \ldots, \delta(i_1-1))$ and $\delta_{t+1} = (\delta(i_t+1), \ldots, \delta(lg(\delta)))$ consist of lines not longer than M.

Consider now the subsequence $\delta_k = (\delta(i_k+1), \ldots, \delta(i_{k+1}-1))$ for some $k \in \{1,2,\ldots,t\}$. Either all these lines are not longer than M or they are all longer than $M+maxr(G)$.

Assume that δ_k is nonempty and that all its lines are longer than $M+maxr(G)$ - we say then that δ_k is an *external segment* of δ. We can write $\delta(i_k) = xw_1$ and $\delta(i_{k+1}) = xw_2$, where $|x| = M$, $|w_1|, |w_2| \leq maxr(G)$. Note that $w_1 \overset{*}{\underset{G}{\Longrightarrow}} w_2$, because $\delta(i_k) \overset{*}{\underset{G}{\Longrightarrow}} \delta(i_{k+1})$ and the derivation steps in-between do not influence (rewrite) x. From the definition of m_G it follows that there exists a derivation $\mu : w_1 \overset{*}{\Longrightarrow} w_2$, with $brd(\mu) \leq m_G$.

Consequently there exists a derivation $\mu' : \delta(i_k) = xw_1 \overset{*}{\Longrightarrow} xw = \delta(i_{k+1})$ with $cont(\mu) = cont(\mu')$ such that $brd(\mu') \leq m_G+M$.

Hence, by replacing the external segments of δ by "new" derivations as above, we obtain a derivation $\delta' : u \overset{*}{\Longrightarrow} v$, such that $brd(\delta') \leq m_G+M$. Thus the lemma holds. ∎

Example 1.1. (continued) Consider the following derivation $\delta : CAB \overset{*}{\Longrightarrow} \Lambda$:

Figure 5.

Special lines in this figure are indicated by short arrows. δ has one external segment; it is a derivation of *CAABBB* from *CAABBB*. The "roof part" (i.e., the part above M, see Figure 6) represents the derivation (*BBB*,*BBAB*,*BBA*,*BBBC*,*BBBBB*,*BBBB*,*BBB*).

Figure 6.

Replacing it by (*BBB*) yields the derivation δ'.

Figure 7.

■

3. UNIONS OF ACTIVE LANGUAGES

As we have indicated already, arbitrary unions of n-active languages in a rb grammar are regular. We prove this result in this section.

The reason that this result holds is that each n-active language is of a very special form (Lemma 3.3). We start by defining some classes of languages useful in our considerations.

Definition 3.1. A language K is *down-closed* if for each word $w \in K$ all sparse subwords of w are also in K. ■

The following well-known result (see [C], pp. 63-64) is very crucial in our proof of Lemma 3.3.

Proposition 3.1. Each down-closed language is regular. ■

We use DC_Θ to denote the family of down-closed languages over the alphabet Θ. For a regular substitution π, $DC_{\Theta,\pi}$ denotes the family $\{\pi(L) \mid L \in DC_\Theta\}$.

Lemma 3.2. Let π be a regular substitution over an alphabet Θ. Then

(1) each language in $DC_{\Theta,\pi}$ is regular, and

(2) $DC_{\Theta,\pi}$ is closed under arbitrary (possibly infinite) unions.

Proof. (1) Obvious. By Proposition 3.1 DC_Θ consists of regular languages and regularity is preserved under regular substitutions.

(2) DC_Θ is clearly closed under arbitrary unions. Consequently $DC_{\Theta,\pi}$ is closed under arbitrary unions because

$$\bigcup_{L \in \mathbf{F}} \pi(L) = \pi\left(\bigcup_{L \in \mathbf{F}} L\right)$$

for any language family $\mathbf{F}$ over Θ. ■

Lemma 3.3. Let $G = (\Sigma,h,\omega)$ be a rb grammar. There exist an integer n_G, an alphabet Θ and a regular substitution π of Θ into Σ^* such that $ACT_n(G) \in DC_{\theta,\pi}$ for each $n > n_G$.

Proof.

Let m_G be a constant (dependent of G) satisfying the statement of Lemma 2.1, let $m_0 = m_G+1$ and let $n_G = m_0+|\omega|$. Let $n > n_G$.

Consider a derivation $\delta : \omega \overset{*}{\Longrightarrow}\Lambda$ and let $\delta(i)$ and $\delta(j)$, $i \leq j$, be two n-active lines of δ. We say that $\delta(i)$ and $\delta(j)$ are *n-related* if $|\delta(t)| > n-m_0$ for each $i \leq t \leq j$.

Assume now that $\delta(i)$ and $\delta(j)$ are two n-related lines of δ. Since all lines of δ between $\delta(i)$ and $\delta(j)$ are longer than $n-m_0$ they all have a common prefix x of length $n-m_0$. Hence for each $i \leq t \leq j$ we can write $\delta(t) = xw_t$, where $|x| = n-m_0$ and $w_i \Longrightarrow w_{i+1} \Longrightarrow \cdots \Longrightarrow w_j$. Moreover $|w_i| = |w_j| = m_0$.

Obviously the notion of n-related lines defines an equivalence relation on the (occurences of) n-active lines of δ. If we take the first ($\delta(i)$) and last ($\delta(j)$) line of each n-related equivalence class, then we obtain a "subsequence"

$\delta(i_1), \delta(j_1), \delta(i_2), \delta(j_2), \ldots, \delta(i_k), \delta(j_k)$

of δ, such that $0 \leq i_1 \leq j_1 < i_2 \leq j_2 < \cdots < i_k \leq j_k \leq lg(\delta)$; this subsequence is referred to as the *n-characteristic sequence of δ*. (Note that it may happen that $i_t = j_t$ because an equivalence class can consist of just one single n-active line of δ; in that case this line occurs twice in our sequence.)

Let $\Theta = \{<u,v> \mid u,v \in \Sigma^*, |u| = |v| = m_0\}$.

The word $<u_1,v_1><u_2,v_2>\cdots<u_k,v_k>$ over Θ such that u_t and v_t are the suffixes of length m_0 of $\delta(i_t)$ and $\delta_{(j_t)}$ respectively, is called the *n-signature of δ* and denoted by $sig_n(\delta)$.

Now let $K_n = \{sig_n(\delta) \mid \delta : \omega \overset{*}{\Longrightarrow} \Lambda\}$.

The notions that we have introduced above can be illustrated as follows. Let δ be a derivation of the following form:

Figure 8.

Then the lines $\delta(i_1)$ and $\delta(j_1)$ are n-related, while the lines $\delta(j_1)$ and $\delta(i_2)$ are not n-related. Note that there are no other n-active lines related to $\delta(i_2)$.

The n-characteristic sequence of δ is the sequence

$\delta(i_1)$, $\delta(j_1)$, $\delta(i_2)$, $\delta(i_2)$, $\delta(i_3)$, $\delta(j_3)$.

The n-signature of δ is $<u_1,v_1><u_2,u_2><u_3,v_3>$.

First we will show that K_n is down-closed.

Claim 1. $K_n \in DC_\theta$.

Proof of Claim 1.

In order to prove that K_n is down-closed, consider a derivation $\delta : \omega \overset{*}{\underset{G}{\Longrightarrow}} \Lambda$ with $sig_n(\delta) = <u_1,v_1><u_2,v_2> \cdots <u_k, v_k>$, $k \geq 1$.

Let $\delta(i_1)$, $\delta(j_1)$, ..., $\delta(i_k)$, $\delta(j_k)$ be the n-characteristic sequence of δ. Since $\delta(j_t)$ and $\delta(i_{t+1})$, $1 \leq t < k$, are not n-related they are separated by a line $\delta(l_t)$, $j_t < l_t < i_{t+1}$, with $|\delta(l_t)| \leq n - m_0$. Furthermore if we set $l_0 = 0$ and $l_k = lg(\delta)$, then obviously we have $l_0 < i_1$, $j_k < l_k$, $|\delta(l_0)| = |\omega| < n - m_0$ and $|\delta(l_{k+1})| = |\Lambda| < n - m_0$.

For each $1 \leq t \leq k$ we have $\delta(l_{t-1}) \overset{*}{\Longrightarrow} \delta(l_t)$. Thus, by Lemma 2.1, there exists a derivation $\mu_t : \delta(l_{t-1}) \overset{*}{\Longrightarrow} \delta(l_t)$ with $brd(\mu_t) \leq m_G + \max\{|\delta(l_{t-1})|, |\delta(l_t)|\} \leq m_G + n - m_0 = n - 1$. Hence $act_n(\mu_t) = \Lambda$ or, in other words, μ_t does not contain n-active lines.

So if we repace the lines $\delta(l_{t-1}), \ldots, \delta(l_t)$ from δ by the lines of μ_t, then we obtain a new derivation $\delta_t' : w \overset{*}{\Longrightarrow} \Lambda$ such that

$sig_n(\delta_t') = <u_1,v_1> \cdots <u_{t-1},v_{t-1}><u_{t+1},v_{t+1}> \cdots <u_k,v_k>$.

Consequently by erasing an arbitrary symbol in a word of K_n we obtain a word in K_n. Thus K_n is down-closed. ∎

In order to illustrate the construction used above consider again the derivation depicted in Figure 8. According to our construction it is possible to replace the subderivation $(\delta(l_1), \delta(l_1+1), \ldots, \delta(l_2))$ of δ by a derivation μ_2 of $\delta(l_2)$ from $\delta(l_1)$ which looks as follows:

Figure 9.

The resulting derivation δ_2' has the n-signature $<u_1,v_1><u_3,v_3>$.

Let $\pi : \Theta \to \Sigma^*$ be the substitution defined by

$\pi(<u,v>) = \{wA \mid A = v(m_0)$ and $w = act_{m_0}(\mu)$ for some $\mu : u \overset{*}{\Longrightarrow} v\}$

Theorem 1.2 implies that π is a regular substitution.

In order to prove that $ACT_n(G) \in DC_{\Theta,\pi}$ we will demonstrate that $ACT_n(G) = \pi(K_n)$. This in turn is accomplished by proving the following two claims (corresponding to the two inclusions involved).

Claim 2. $ACT_n(G) \subseteq \pi(K_n)$.

Proof of Claim 2.

Let $w \in ACT_n(G)$, so $w = act_n(\delta)$ for some derivation $\delta : \omega \overset{*}{\underset{G}{\Longrightarrow}} \Lambda$. Let $\delta(i_1), \delta(j_1), \ldots, \delta(i_k), \delta(j_k)$ be the n-characteristic sequence of δ and let $sig_n(\delta) = <u_1,v_1> \cdots <u_k,v_k>$.

If we choose $l_0, l_1, \ldots, l_k$ as in the proof of Claim 1, then

$act_n(\delta) = act_n(\delta_1) \cdot \cdots \cdot act_n(\delta_k)$, where each derivation δ_t, $1 \leq t \leq k$, is of the form $\delta_t = (\delta(l_{t-1}), \delta(l_{t-1}+1), \ldots, \delta(l_t))$.

Moreover, for each $1 \leq t \leq k$, $\delta_t = \delta_t^1 \otimes \delta_t^2 \otimes \delta_t^3$

where $\delta_t^1 = (\delta(l_{t-1}), \ldots, \delta(i_t))$, $\delta_t^2 = (\delta(i_t), \ldots, \delta(j_t))$ and $\delta_t^3 = (\delta(j_t), \ldots, \delta(l_t))$.

Thus $act_n(\delta_t) = act_n(\delta_t^1) act_n(\delta_t^2) act_n(\delta_t^3) = act_n(\delta_t^2) \cdot \delta(j_t)(n)$.

The claim now follows by observing that

$act_n(\delta_t^2) \cdot \delta(j_t)(n) = act_{m_0}(\mu_t) \cdot v_t(m_0)$ for the derivation $\mu_t : u_t \overset{*}{\underset{G}{\Longrightarrow}} v_t$ with $cont(\delta_t^2) = cont(\mu_t)$ and consequently $act_n(\delta_t) \in \pi(<u_t, v_t>)$; thus $act_n(\delta) \in \pi(sig_n(\delta)) \subseteq \pi(K_n)$. ∎

Claim 3. $\pi(K_n) \subseteq ACT_n(G)$.

Proof of Claim 3.

If $wA \in \pi(<u, v>)$, where $A = v(m_0)$, then there exists a derivation $\mu : u \overset{*}{\Longrightarrow} v$ with $act_{m_0}(\mu) = w$. Thus for an arbitrary $x \in \Sigma^*$ with $|x| = n - m_0$ there exists a derivation $\delta' : xu \overset{*}{\Longrightarrow} xv$ with $act_n(\delta') = w$.

This enables us to replace in a derivation $\delta : \omega \overset{*}{\Longrightarrow} \Lambda$ each of the subderivations $\delta_t = (\delta(l_{t-1}), \ldots, \delta(l_t))$ (in the notation as above) by a corresponding derivation $\delta_t' : \delta(l_{t-1}) \overset{*}{\Longrightarrow} \delta(l_t)$ such that $act_n(\delta_t') = z_t$, where z_t is an arbitrarily chosen element of $\pi(<u_t, v_t>)$.

From these observations the claim easily follows. ∎

So we have shown that $ACT_n(G) = \pi(K_n)$ for each $n > n_G = m_0 + |\omega|$, where K_n is a down-closed language over the alphabet Θ. Hence the lemma holds. ∎

We are ready now to prove the main result of this section.

Theorem 3.4. Let G be a rb grammar. Then $\bigcup_{i \in I} ACT_i(G)$ is regular for arbitrary $I \subseteq \mathbf{N}^+$.

Proof.

Let $I \subseteq \mathbf{N}^+$. According to Lemma 3.3 there exists a constant n_G for G such that $ACT_n(G) \in DC_{\Theta,\pi}$ for every $n > n_G$, where π is a suitably chosen regular substitution over an alphabet Θ.

Let $I_1 = \{i \in I \mid i > n_G\}$ and $I_2 = \{i \in I \mid i \leq n_G\}$. From Lemma 3.2 it follows that $\bigcup_{i \in I_1} ACT_i(G)$ is regular.

Since $\bigcup_{i \in I} ACT_i(G) = \bigcup_{i \in I_1} ACT_i(G) \cup \bigcup_{i \in I_2} ACT_i(G)$, I_2 is finite and the class of regular languages is closed under finite unions, Corollary 1.3 implies that $\bigcup_{i \in I} ACT_i(G)$ is regular. ∎

4. CP SYSTEMS AND INTERNAL CP SYSTEMS

Right boundary grammars form a very basic building block in the general theory of automata and grammars (as presented in [R]). In particular they form a basic component of push-down automata; a push-down automaton can be seen as two grammars, one right-linear , the other right-boundary, cooperating together: this way of formalizing push-down automata leads one to the theory of coordinated pair systems (see, e.g., [R], [EHR1] and [EHR2]). Using this point of view we will translate now our results on the use of memory in right-boundary grammars to results concerning the use of memory in push-down automata.

We begin by recalling the notion of a coordinated pair system.

Definition 4.1. A *coordinated pair system, cp system* for short, is a triple $G = (G_1 , G_2 , R)$, where

(1) $G_1 = (\Sigma_1 , P_1 , S_1 , \Delta)$ is a rl grammar,

(2) $G_2 = (\Sigma_2 , P_2 , S_2)$ is a rb grammar with $S_2 \in \Sigma_2$, and

(3) $R \subseteq P_1 \times P_2$. ■

Elements of R are referred to as *rewrites of* G.

Definition 4.2. Let $G = (G_1 , G_2, R)$ be a cp system, where $G_1 = (\Sigma_1 , P_1 , S_1 , \Delta)$ and $G_2 = (\Sigma_2 , P_2 , S_2)$.

(1) Let $x_1 , y_1 \in \Sigma_1^*$ and $x_2 , y_2 \in \Sigma_2^*$. If $x_1 \underset{G_1}{\overset{\pi_1}{\Longrightarrow}} y_1$ and $x_2 \underset{G_2}{\overset{\pi_2}{\Longrightarrow}} y_2$ for a rewrite $\pi = (\pi_1 , \pi_2) \in R$, then we say that (x_1 , y_1) *directly computes* (x_2 , y_2) *in* G *using* π and we denote this by $(x_1 , x_2) \underset{G}{\overset{\pi}{\Longrightarrow}} (y_1 , y_2)$.

$\underset{G}{\overset{*}{\Longrightarrow}}$ denotes the reflexive and transitive closure of $\underset{G}{\Longrightarrow}$. If $(x_1 , x_2) \underset{G}{\overset{*}{\Longrightarrow}} (y_1 , y_2)$, then we say that (x_1 , x_2) *computes* (y_1 , y_2) *(in* G*).*

(2) A *computation (in* G*)* is a sequence $\rho = (x_0 , x_1 , \ldots , x_n)$, $n \geq 0$, of elements

from $\Sigma_1^* \times \Sigma_2^*$ such that $x_{i-1} \underset{G}{\Longrightarrow} x_i$ for each $1 \leq i \leq n$.

We say that ρ *computes* x_n *from* x_0 and denote this by $\rho : x_0 \overset{*}{\Longrightarrow} x_n$. n is called the *length of* ρ and is denoted by $lg(\rho)$. For $0 \leq i \leq n$ we use $\rho(i)$ to denote x_i.

If $\rho : (S_1, S_2) \overset{*}{\Longrightarrow} (w, \Lambda)$ for some $w \in \Delta^*$, then ρ is called *successful*.

(3) The *language of* G, denoted $L(G)$, is the set $\{w \in \Delta^* \mid (S_1, S_2) \overset{*}{\Longrightarrow} (w, \Lambda)\}$.

■

We extend the formal notions describing the active use of memory in rb grammars to cp systems as follows.

Definition 4.3. Let G be a cp system and let $n \in \mathbf{N}^+$.

(1) Let $\rho = (\rho(0), \rho(1), \ldots, \rho(k))$ be a computation in G. The *n-active record* of ρ, denoted $act_n(\rho)$, is the word $\varphi_n(\rho(0)), \varphi_n(\rho(1)) \cdots \varphi_n(\rho(k-1))$, where

$\varphi_n : \Sigma_1^* \times \Sigma_2^* \to \Sigma_2 \cup \{\Lambda\}$ is the mapping defined by

$$\varphi_n((u, v)) = \begin{cases} v(n) & \text{, if } |u| = n, \\ \Lambda & \text{, otherwise.} \end{cases}$$

(2) The *n-active-language* of G, denoted $ACT_n(G)$, is the language

$\{act_n(\rho) \mid \rho : (S_1, S_2) \Longrightarrow (w, \Lambda)$ for some $w \in \Delta^*\}$. ■

In studying the active use of memory in a cp system G we are mainly interested in the behavior of the second component of successful computations. This behavior does not change if we erase all terminal symbols in all productions of the first component (and in the corresponding rewrites) of G. The cp system obtained in this way is called the *internal version of* G ; a cp system is called *internal* if it equals its internal version (i.e., it has no terminal symbols on its first component).

Definition 4.4. Let $G = (G_1, G_2, R)$ be a cp system with

$G_1 = (\Sigma_1, P_1, S_1, \Delta)$.

(1) G is called *internal* if $\Delta = \emptyset$.

(2) The *internal version of* G, denoted $int(G)$, is the cp system $\tilde{G} = (\tilde{G}_1 , G_2 , \tilde{R})$ where $\tilde{G}_1 = (\Sigma_1 - \Delta, \tilde{P}_1 , S_1 , \emptyset)$ with

$\tilde{P} = \{(X,Z) \mid (X,wZ) \in P$ for some $X \in \Sigma_1 - \Delta$, $w \in \Delta^*$ and $Z \in (\Sigma_1 - \Delta) \cup \{\Lambda\}\}$ and

$\tilde{R} = \{(X \to Z, \pi_2) \mid (X \to wZ, \pi_2) \in R$ for some $X \in \Sigma_1 - \Delta$, $w \in \Delta^*$ and

$Z \in (\Sigma_1 - \Delta) \cup \{\Lambda\}\}$. ■

Hence $int(G)$ works precisely as G does except that it ignores the "input aspect" of G. Consequently as far as the use of memory is concerned one can consider $int(G)$ rather than G.

Lemma 4.1. Let G be a cp system and let $n \in \mathbf{N}^+$. Then $ACT_n(G) = ACT_n(int(G))$. ■

5. INTERNAL CP SYSTEMS AND RB GRAMMARS.

In the previous section we have seen that rather than investigating the use of memory in general cp systems one may restrict oneself to internal cp systems. Actually this is the first step needed in "reducing" cp systems to rb grammars. The second step is taken in this section. We demonstrate how to represent (computations in) an internal cp system by (derivations in) a finite set of rb grammars.

Definition 5.1. Let Θ, Σ_1 and Σ_2 be alphabets and let $\psi : \Theta^* \to \Sigma_1^*$ and $\varphi : \Theta^* \to \Sigma_2^*$ be weak codings. Let $(Z,w) \in (\Sigma_1 \cup \{\Lambda\}) \times \Sigma_2^*$ and $u \in \Theta^*$.

We say that u *(ψ,φ)-represents* (Z,w), denoted $u\ [\psi,\varphi > (Z,w)$ if the following holds.

(i) If $w \neq \Lambda$, then $|u| = |w|$, $\psi(last(u)) = Z$ and $\varphi(u) = w$.

(ii) If $w = \Lambda$ and $Z \neq \Lambda$, then $|u| = 1$, $\psi(u) = Z$ and $\varphi(u) = \Lambda$.

(iii) If $w = \Lambda$ and $Z = \Lambda$, then $u = \Lambda$. ■

Definition 5.2. Let $G = (G_1 , G_2 , R)$ be an internal cp system with $G_1 = (\Sigma_1 , P_1 , S_1 , \emptyset)$, $G_2 = (\Sigma_2 , P_2 , S_2)$ and let $H_i = (\Theta_i , Q_i , \omega_i)$, $0 \leq i \leq k$, be a collection of rb grammars with $\Theta = \bigcup_{i=1}^{k} \Theta_i$. Finally let $\psi : \Theta^* \to \Sigma_1^*$ and $\varphi : \Theta^* \to \Sigma_2^*$ be weak codings.

(1) Let ρ, be a computation in G and let δ be a derivation in H_i $(0 \leq i \leq k)$. We say that δ *(ψ,φ)-represents* ρ, denoted $\delta\ [\psi,\varphi > \rho$, if $lg(\delta) = lg(\rho)$ and $\delta(j)\ [\psi,\varphi > \rho(j)$ for each $0 \leq j \leq lg(\delta)$.

(2) G is *(ψ,φ)-represented by* $H_0 , H_1 , \ldots, H_k$ if the following holds.

(i) For each computation $\rho : (S_1 , S_2) \overset{*}{\Longrightarrow} (x,w)$ in G, there exists an $i \in \{0,1,\ldots k\}$ and a derivation $\delta : \omega_i \overset{*}{\Longrightarrow} u$ in H_i such that $\delta\ [\psi,\varphi > \rho$.

(ii) For each $i \in \{0,1, \ldots,k\}$ and each derivation $\delta : \omega_i \overset{*}{\underset{H_i}{\Longrightarrow}} u$ in H_i there exists a computation $\rho : (S_1 , S_2) \overset{*}{\Longrightarrow}(x,w)$ in G such that $\delta\ [\psi,\varphi > \rho$. ■

Theorem 5.1. For every internal cp system G there exist weak codings ψ and φ and a collection of rb grammars $H_0 , H_1 , \ldots, H_k$ such that G is (ψ,φ)-represented by $H_0 , H_1 , \ldots, H_k$.

Proof.

Let $G = (G_1 , G_2 , R)$ be an internal cp system where $G_1 = (\Sigma_1 , P_1 , S_1 , \emptyset)$ and $G_2 = (\Sigma_2 , P_2 , S_2)$. Let $\Sigma_1 = \{X_1 , X_2 , \ldots, X_k\}$. The rb grammars $H_i = (\Theta_i , Q_i , T_i)$, $i = 0, 1, \ldots, k$, are defined as follows.

(1) $\Theta_0 = \{[A, Y, Z] \mid A \in \Sigma_2$ and $Y, Z \in \Sigma_1 \cup \{\Lambda\}\}$,

$T_0 = [S_2 , \Lambda , S_1]$ and

Q_0 contains the following productions

$[A, Y, X] \to \Lambda$ if and only if $(X \to Y, A \to \Lambda) \in R$ for $A \in \Sigma_2 , X \in \Sigma_1 , Y \in \Sigma_1 \cup \{\Lambda\}$,

$[A, Y, X] \to [B, Y, Z]$ if and only if $(X \to Z, A \to B) \in R$ for $A \in \Sigma_2 , X \in \Sigma_1$ and $Y, Z \in \Sigma_1 \cup \{\Lambda\}$, and

$[A, Y, X] \to [B_1, Y, Z_1][B_2 , Z_1 , Z_2] \cdots [B_r , Z_{r-1} , Z]$ if and only if $(X \to Z, A \to B_1 B_2 \cdots B_r) \in R$ for $r \geq 2$, $A, B_1 , \ldots, B_r \in \Sigma_2 , X \in \Sigma_1$ and $Y, Z, Z_1 , \ldots, Z_{r-1} \in \Sigma_1 \cup \{\Lambda\}$.

(2) For each $i \in \{1, 2, \ldots, k\}$,

$\Theta_i = \{[A, Y, Z] \mid A \in \Sigma_2 , Y \in \Sigma_1 \cup \{\bar{X}_i\}$ and $Z \in \Sigma_1\} \cup \{\bar{X}_i\}$,

$T_i = [S_2 , \bar{X}_i , S_1]$,

Q_i contains the productions

$[A, \bar{X}_i, X] \to \bar{X}_i$ if and only if $(X \to X_i , A \to \Lambda) \in R$ for $A \in \Sigma_2$ and $X \in \Sigma_1$,

$[A,Y,X] \to \Lambda$ if and only if $(X \to Y, A \to \Lambda) \in R$ for $A \in \Sigma_2$ and $X, Y \in \Sigma_1$,

$[A,Y,X] \to [B,Y,Z]$ if and only if $(X \to Z, A \to B) \in R$ for $A, B \in \Sigma_2 , X, Z \in \Sigma_1$ and

$Y \in \Sigma_1 \cup \{\overline{X}_i\}$ and

$[A,Y,X] \to [B_1, Y, Z_1][B_2, Z_1, Z_2] \cdots [B_r, Z_{r-1}, Z]$ if and only if

$(X \to Z, A \to B_1 B_2 \cdots B_r) \in R$ for

$r \geq 2$, $A, B_1, \ldots, B_r \in \Sigma_2$, $X, Z_1, \ldots, Z_{r-1}, Z \in \Sigma_1$ and $Y \in \Sigma_1 \cup \{\overline{X}_i\}$.

(3) Finally for $A \in \Sigma_2$, $Y, Z \in \Sigma_1 \cup \{\Lambda\}$ and $X, W \in \Sigma_1$ we define

$\psi([A, Y, Z]) = Z$, $\psi([A, \overline{X}, W]) = W$, $\psi(\overline{X}) = \overline{X}$,

$\varphi([A, Y, Z]) = A$, $\varphi([A, \overline{X}, W]) = A$, $\varphi(\overline{X}) = \Lambda$,

For each $1 \leq i \leq k$ the rb grammar H_i is used to represent computations $\rho : (S_1, S_2) \overset{*}{\underset{G}{\Longrightarrow}} (X_i, \Lambda)$. Computations of the form $\rho : (S_1, S_2) \overset{*}{\underset{G}{\Longrightarrow}} (x, w)$ with $w \neq \Lambda$ as well as computations of the form $\rho : (S_1, S_2) \overset{*}{\underset{G}{\Longrightarrow}} (\Lambda, \Lambda)$ are represented by suitable derivations in H_0. Computations of these forms are respectively called *blocked, unfinished and successful.*

Claim 1. For every computation $\rho : (S_1, S_2) \overset{*}{\underset{G}{\Longrightarrow}} (x, w)$ in G there exists an $i \in \{0, 1, \ldots, k\}$ and a derivation $\delta : \omega_i \overset{*}{\underset{H_i}{\Longrightarrow}} u$ in H_i such that $\delta[\psi, \varphi > \rho$.

Proof of Claim 1.

Let ρ be a computation of length n in G with $\rho(0) = (S_1, S_2)$. For $0 \leq j \leq n$ we write $\rho(j) = (Z_j, w_j)$.

(a) If ρ is either unfinished or successful (so either $w_n \neq \Lambda$ or $(Z_n, w_n) = (\Lambda,\Lambda)$), then it is possible to construct a derivation δ of length n in H_0 such that $\delta\,[\psi,\varphi> \delta$.

This is done as follows. Let $\delta(0) = [S_2, \Lambda, S_1]$.

We proceed inductively. Assume that $\delta(j)$, $j < n$, is already constructed.

From the form of the productions in H_0 it easily follows that $\delta(j)$ is of the form $[A_1, \Lambda, Y_1][A_2, Y_1, Y_2] \cdots [A_l, Y_{l-1}, Y_l]$, for some $l \geq 1$; thus $Z_j = \psi([A_l, Y_{l-1}, Y_l]) = Y_l$ and $w_j = \varphi(\delta(j)) = A_1 A_2 \cdots A_l$. (We will say that this occurrence of A_l is *active* in w_j.) Let π be the rewrite such that $\rho(j) \overset{\pi}{\underset{G}{\Longrightarrow}} \rho(j+1)$.

We consider separately three cases.

(a1) $\pi = (X \to Z, A \to B_1 B_2 \cdots B_r)$ for some $r \geq 1$, $Z \neq \Lambda$.

Obviously $X = Y_l$ and $A = A_l$. We use now a production $[A_l, Y_{l-1}, Y_l] \to [B_1, Y_{l-1}, W_1][B_2, W_1, W_2] \cdots [B_r, W_{r-1}, Z]$ to derive $\delta(j+1)$ from $\delta(j)$. The variables $W_1, \ldots, W_{r-1}$ are determined as follows. For $i \in \{1,2, \ldots, r-1\}$ let $\rho(s_i)$ be the element of the computation ρ in which the occurrence B_i "becomes active" on the second component; more precisely: s_i is the least integer larger than j such that $w_{s_i} = A_1 A_2 \cdots A_{l-1} B_1 \cdots B_i$ (thus $s_r = j+1$). We then choose $W_i = Z_{s_i}$. Note that it may happen that $W_i = \Lambda$ (if $s_i = n$). If B_i never "becomes active" (thus it is never rewritten) in ρ, then W_i may be chosen arbitrarily.

(a2) $\pi = (X \to Z, A \to \Lambda)$, where $Z \neq \Lambda$.

Obviously $X = Y_l$ and $A = A_l$. Since ρ is not a blocking computation $l \neq 1$. The occurrence A_{l-1} will be active in the next step of the computation ρ and, since $[A_{l-1}, Y_{l-2}, Y_{l-1}]$ was introduced by a production as under (a1) above, $Y_{l-1} = Z$. So we can apply the production $[A_l, Y_{l-1}, Y_l] \to \Lambda$ to obtain $\delta(j+1)$ from $\delta(j)$.

(a3) $\pi = (X \to \Lambda, A \to \Lambda)$.

Note that this rewrite can be applied in the last step of a computation only. Again $X = Y_l$ and $A = A_l$.

If $l = 1$, then ρ is successful. Otherwise, if $l > 1$, then ρ is unfinished. Using the same arguments as before we conclude that $Y_{l-1} = \Lambda$.

So in both cases we can apply the production $[A_l, \Lambda, Y_l] \to \Lambda$ to obtain $\delta(j+1)$ from $\delta(j)$. (Note that if $l = 1$, then $\delta(j+1) = \Lambda$.)

(b) If ρ is a blocking computation with $\rho(n) = (X_t\ ,\ \Lambda)$ for $t \in \{1, \ldots, k\}$, then we construct a derivation δ of length n in H_t such that $\delta[\psi,\varphi>\rho$.

This time we set $\delta(0) = [S_2\ ,\ \overline{X}_t\ ,\ S_1]$.

The recursive construction of the sequence $\delta(1)\ ,\ \delta(2)\ ,\ \ldots,\delta(n)$ is defined as for successful computations, except for the last step which we discuss now separately.

$\rho(n) = (X_t\ ,\ \Lambda)$ can be obtained only be applying a rewrite $\pi = (X \to X_t\ ,\ A \to \Lambda)$ to $\rho(n-1)\ = (X,\ A)$; thus $\delta(n-1) = [A,\ X_t\ ,\ X]$.

Let $\delta(n) = \overline{X}_t$. Then $\delta(n-1)$ directly derives $\delta(n)$ in H_t using the production $[A,\ X_t,\ X] \to \Lambda$.

From (a) and (b) above Claim 1 follows. ∎

Claim 2. Let $i \in \{0,\ 1,\ \ldots,\ k\}$ and let $\delta : T_i \overset{*}{\Longrightarrow} u$ be a derivation in H_i. Then there exists a computation $\rho : (S_1\ ,\ S_2) \overset{*}{\Longrightarrow} x$ in G such that $\delta\ [\psi,\ \varphi > \rho$.

Proof of Claim 2.

The proof of this claim is rather obvious. Let δ be a derivation in H_i with $\delta(0) = T_i$.

First we set $\rho(0) = (S_1\ ,\ S_2)$. Then $[S_2\ ,\ \overline{X}_i\ ,\ S_1] = \delta(0)[\psi,\varphi>\rho(0)$. The other elements of ρ can be found using the weak codings ψ and φ.

For $1 \leq j \leq lg\,(\delta)$ let $\rho(j) = (Z_j\ ,\ w_j)$, where $Z_j = \Lambda$ if $\delta(j) = \Lambda$ and $Z_j = \psi(last\,(\delta(j)))$ otherwise, and let $w_j = \varphi(\delta(j))$.

It can be easily checked that the so defined sequence ρ is a computation in G. ∎

The theorem follows from the above two claims. ∎

Note that the notion of representation discussed above is very strong in the sense that $\#\Sigma_1+1$ rb grammars represent *all*, hence successful and not successful, computations that start with (S_1, S_2). If one is interested (as we will be in analyzing the use of memory) in successful computations only, then one gets a simpler representation theorem.

Definition 5.3. Let $G = (G_1, G_2, R)$ be an internal cp system with $G_1 = (\Sigma_1, P_1, S_1, \emptyset)$, $G_2 = (\Sigma_2, P_2, S_2)$ and let $H = (\Theta, Q, \omega)$ be a rb grammar. Finally let $\psi : \Theta^* \to \Sigma_1^*$ and $\varphi : \Theta^* \to \Sigma_2^*$ be codings.

G is *successfully* (ψ,φ)*-represented by* H if the following holds.

(i) For each successful computation ρ in G there eixsts a successful derivation δ in H such that $\delta[\psi,\varphi>\rho$.

(ii) For each successful derivation δ in H there exists a successful computation ρ in G such that $\delta[\psi,\varphi>\rho$. ∎

Remark 5.1. Note that if (Z,w) is an element of a successful computation, then $w = \Lambda$ implies $Z = \Lambda$. Hence we don't have to consider requirement (ii) from Definition 5.1. For this reason we can use codings ψ and φ rather than weak codings to represent an internal cp system. ∎

Theorem 5.2. For every internal cp system G there exist codings ψ and φ and a rb grammar H such that G is successfully (ψ,φ)-represented by H.

Proof.

The rb grammar H_0 as constructed in the proof of Theorem 5.1 successfully (ψ,φ)-represents the internal cp system G. This is easily seen using the arguments of that proof. Note that the homomorphisms ψ and φ are codings when restrained to the alphabet of H_0. ∎

We are ready now to translate our results on the active use of memory in rb grammars into results describing the active use of memory of push-down automata as modelled by cp systems.

Theorem 5.3. Let G be a cp system.

(1) For each $n \geq 1$, $ACT_n(G)$ is regular.

(2) For each $I \subseteq \mathbf{N}^+$, $\bigcup_{n \in I} ACT_n(G)$ is regular. ∎

Proof. Let H be a rb grammar that successfully (ψ,φ)-represents the internal cp system $int(G)$ for some codings ψ and φ. Using the fact that φ is a coding it is easily seen that for every $n \in \mathbf{N}^+$ $act_n(\rho) = \varphi(act_n(\delta))$ where ρ is a successful computation in $int(G)$ and δ is a successful derivation in H such that $\delta[\psi,\varphi>\rho$. Hence $ACT_n(int(G)) = \varphi(ACT_n(H))$ and consequently, by Lemma 4.1, $ACT_n(G) = \varphi(ACT_n(H))$ for every $n \in \mathbf{N}^+$.

Thus, for $I \subseteq \mathbf{N}^+$, $\bigcup_{n \in I} ACT_n(G) = \bigcup_{n \in I} \varphi(ACT_n(H)) = \varphi(\bigcup_{n \in I} ACT_n(H))$ is regular, because by Theorem 3.4 $\bigcup_{n \in I} ACT_n(H)$ is regular and the class of regular languages is closed under homomorphisms. ∎

DISCUSSION

Within the framework of ects systems (see [R]) rb grammars form a basic building block in constructing various types of grammars and machines known from the literature. Hence there is a need for a fundamental research concerning rb grammars.

In this paper we have investigated the use of memory in rb grammars. We have chosen a specific way of "tracing" the use of memory in rb grammars: we record the sequence of active use of a particular memory cell during a derivation. Then it turns out that *all* active records for a given cell for *all successful* derivations form a regular language (Corollary 1.3). As a matter of fact these regular languages have a very specific structure which makes the "overall active use of memory" in a rb grammar regular: the union over any set of memory cells of all active records for all successful derivations is regular (Theorem 3.4)!!!

Cp systems form a subclass of ects systems that correspond very closely to (are another formulation of) push-down automata. A cp system is a "coordinated pair" of a right-linear grammar and a rb grammar: in this combination the rb grammar component represents the (infinite) memory structure of the system. From this point of view investigating the use of memory in rb systems is very natural (and very much needed).

In order to transfer our results on the active use of memory in rb grammars to the level of cp systems (where the work of the rb component is coordinated by the right-linear component) we prove a representation theorem (Theorem 5.1) for cp systems which allows one (in the investigation of computations in cp systems) to consider a *finite* number of rb grammars rather than a cp system. Actually the representation theorem we prove is somewhat "too strong" for the considerations of this paper: if we consider *successful*

computations only, *one* rb grammar suffices to represent a cp system (Theorem 5.2). However we believe that our general theorem constitutes an important link between cp systems and rb grammars when all computations are being considered - we hope to explore this theorem more in the future research.

Using the representation theorem we transfer our result on "strong regularity" of the active use of memory to cp systems (Theorem 5.3).

We believe that this paper illustrates the usefulness of the fundamental research concerning rb grammars and of the cp systems point of view at pushdown automata. It seems to be easier (and more elegant) to prove basic results on the level of rb grammars and then transfer them to the level of cp systems (by "once and forever" established representation theorem) rather than to prove the corresponding results directly for cp systems.

We consider this paper as a first step into the systematic investigation of the use of memory in rb grammars and cp systems. Clearly there are other than active ways of recording the use of a memory in rb grammars (e.g., for a given n-th memory cell one could record for each line of a given computation the contents of the cell - the set of all such records for all successful computations forms the *n-full record language of G)*. How complicated are other types of "recording the memory" languages? Are they regular? Are their arbitrary unions regular?

We are presently working on a number of problems of this nature and hope to present the results of our research in a forthcoming paper.

ACKNOWLEDGEMENTS

The first and the third author gratefully acknowledge the support of NSF grant MCS 83-05245.

REFERENCES

[C] Conway, J.H., *Regular Algebra and Finite Machines*, Chapman and Hall, London, 1971.

[EHR1] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Real time coordinated pair systems", Techn. Rep. CU-CS-259-85, Department of Computer Science, University of Colorado, Boulder, 1985.

[EHR2] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Computations in coordinated pair systems", Techn. Rep. No. CU-CS-260-84, Department of Computer Science, University of Colorado at Boulder, 1984.

[EHR3] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Coordinated pair systems; Part 1: Dyck words and classical pumping", Techn. Rep. No. CU-CS-275-84, Department of Computer Science, University of Colorado, Boulder, 1984.

[EHR4] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "Coordinated pair systems; Part 2: Sparse structure of Dyck words and Ogden's Lemma", Techn. Rep. No. CU-CS-276-84, Department of Computer Science, University of Colorado at Boulder, 1984.

[H] Harrison, M., *Introduction to formal language theory*, Addison-Wesley, Reading, Massachusetts, 1978.

[R] Rozenberg, G., "On coordinated selective substitutions: Towards a unified theory of grammars and machines", *Theoretical Computer Science*, Vol.37, 1985.

[S] Salomaa, A., *Formal languages*, Academic Press, London-New York, 1969.

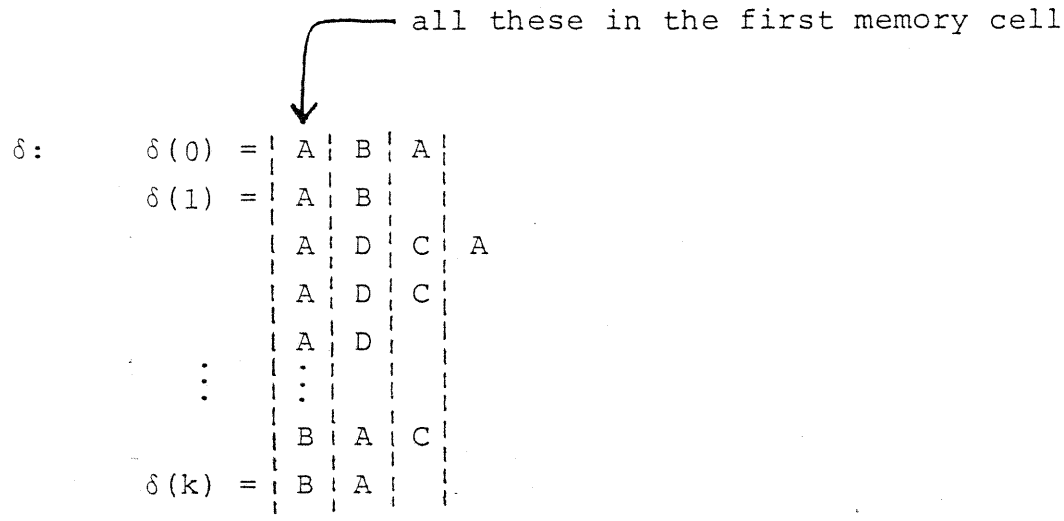
all these in the first memory cell
δ:
δ(0) = A B A
δ(1) = A B
A D C A
A D C
A D
B A C
δ(k) = B A

Figure 1

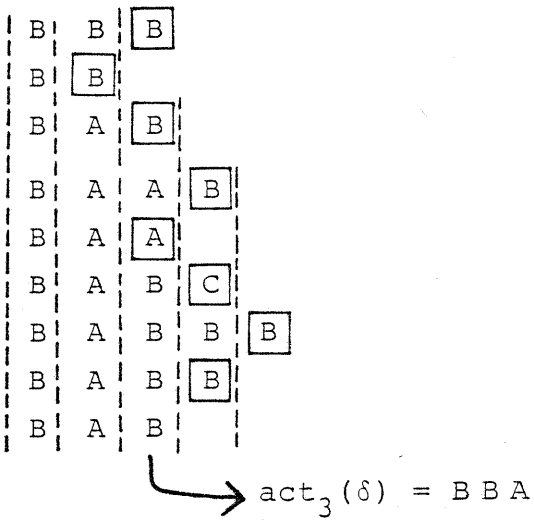

Figure 2

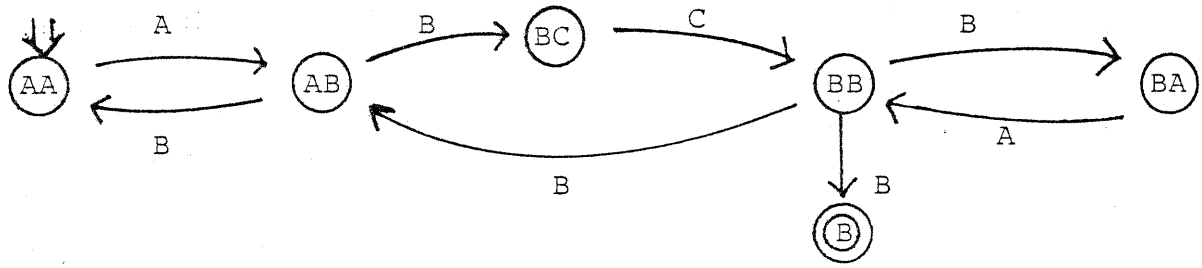

Figure 3

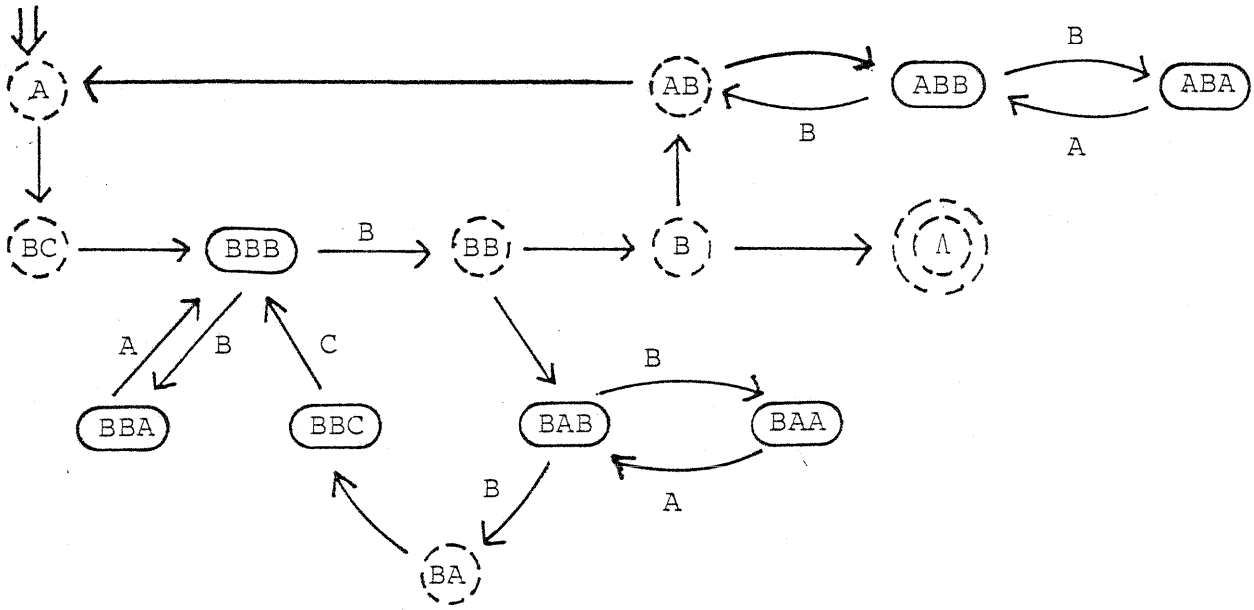

Figure 4

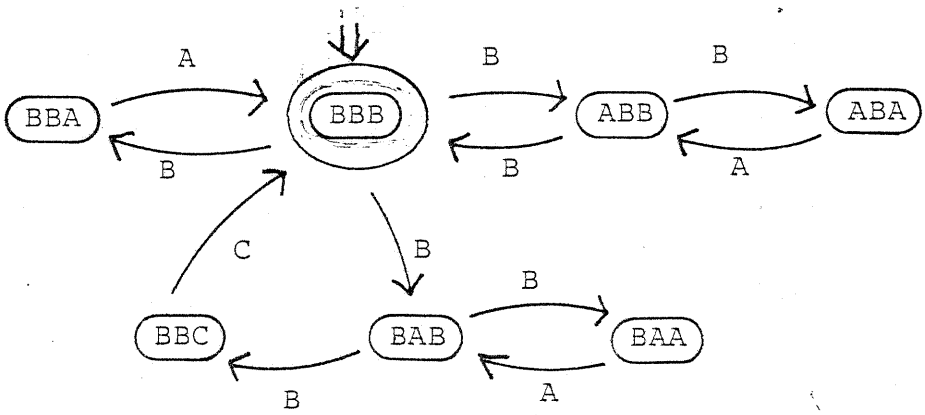

Figure 5

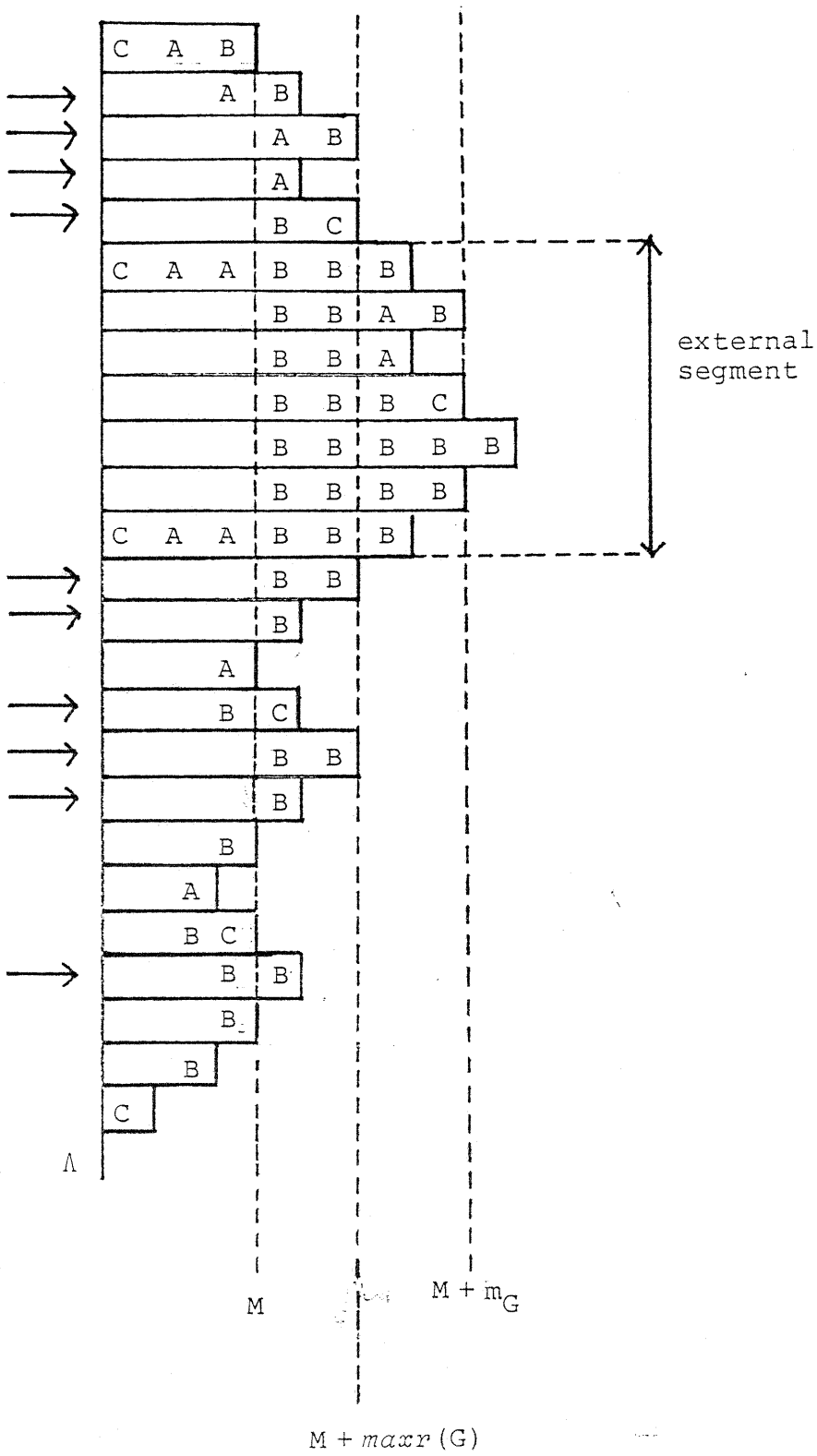
C A B
A B
A B
A
B C
C A A B B B
B B A B
B B A
B B B C
B B B B B
B B B B
C A A B B B
B B
B
A
B C
B B
B
B
A
B C
B B
B
B
B
C
Λ
external segment
M
M + m_G
M + maxr (G)

Figure 6

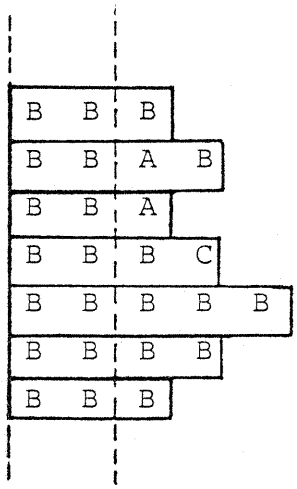

Figure 7

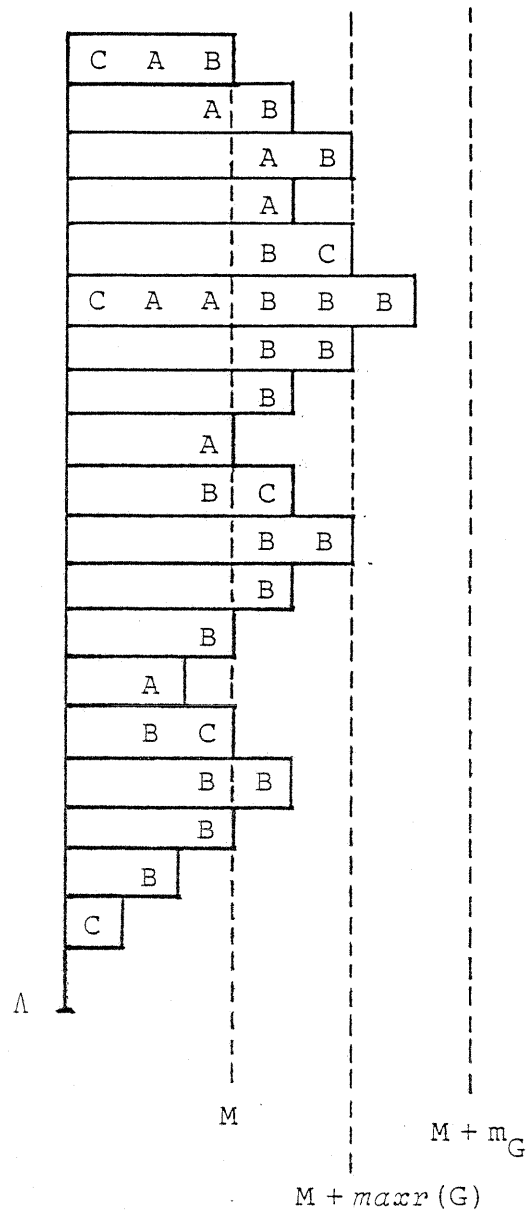

Figure 8

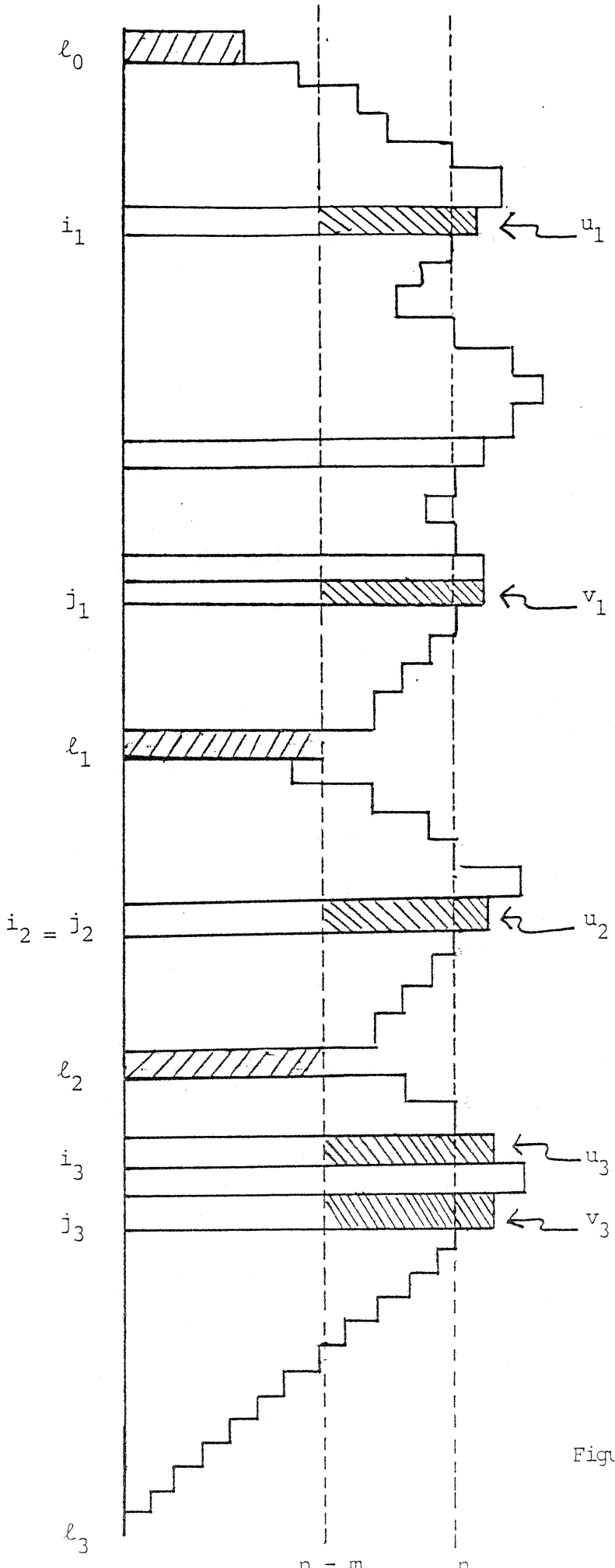

Figure 9

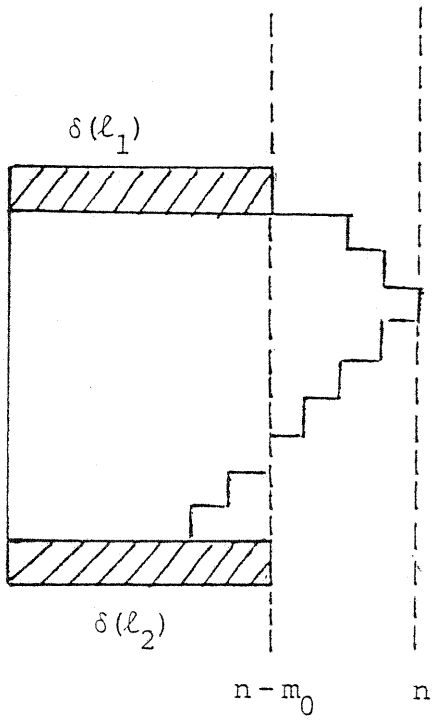

Figure 10