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

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

CU-CS-299-86 September, 1985

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

ON THE FULL RECORDS OF THE 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

Right-boundary grammars, *rb grammars* for short, form the basic component of *coordinated pair systems*, *cp systems* for short, which in turn correspond very closely to (are another formulation of) *push-down automata* (a rb 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 rules are allowed). A cp system is a pair of grammars the first of which is right-linear and the second right-boundary. This pair of grammars works (rewrites) in a coordinated (synchronized) fashion; the right-linear component models the input and and the finite control of a push-down automaton while the rb component models the push-down store.

Adjusting consecutive words of a derivation δ in a rb grammar G under each other (letter-by-letter), for each positive integer n one gets (a "column" word which is) a record of the use of the n-th memory cell during δ. Collecting *all* such records for *all* "successful" derivations δ one gets the *n-full record language of G*, denoted $FR_n(G)$. If one takes from the full record of the use of the n-th cell (during δ) only these observations (letters) which correspond to the moments when the n-th memory cell is rewritten (is "active") then one gets the *active record* of the use of the n-th cell (during δ). Collecting all such records (for all successful δ) one gets the *n-active language of G*, denoted $ACT_n(G)$.

In a previous paper we have proved that, for each rb system G and for each n, $ACT_n(G)$ is regular and moreover for each set I of positive integers $\bigcup_{n \in I} ACT_n(G)$ is regular. In the present paper we prove that, for each rb system G and each n, $FR_n(G)$ is regular *but* $\bigcup_{n \in I} FR_n(G)$ does not have to be regular for arbitrary sets I of positive integers.

INTRODUCTION.

The theory of *ects systems* provides a common framework for a variety of grammar and machine models studied in the literature (see [R]). Within this approach various models are represented as tuples of basic components all of which are rewriting systems that work together in a coordinated (synchronized) manner. One of such very basic building blocks are *right-boundary grammars*, *rb grammars* for short (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 rules are allowed).

In particular rb grammars form the basic component of *coordinated pair systems*, *cp systems* for short. A cp system consists of a pair of grammars the first of which is right-linear (*rl* for short) and the second right-boundary. A rewriting in a cp system consists of a pair of rewritings: one in the rl component and one in the rb component - in this way one gets coordinated (or synchronized) rewriting in both components. It is easily seen that cp systems correspond very closely to (are another formulation of) push-down automata: the rl component models, the input and the finite state control while the rb component models the infinite push-down store.

Since *rb* grammars play such prominent role in the theory of grammars and machines they should be thoroughly investigated. First of all we notice that rb grammars form a special case of *Buchi canonical regular systems* (see [B]) which were a subject of intense investigation in the "early days" of automata theory (see, e.g., [S1] and [G]). More recently rb grammars were investigated in connection with their use in cp systems, see, e.g., [ER] and [EHR]. In particular in [EHR] we have initiated the investigation into the use of memory in rb grammars. The main idea presented in [EHR] is as follows.

A rb grammar G may be viewed as a processor operating on a data structure which is a one-way infinite array of (memory) cells - the processing (of a special type) takes place at the right end of the nonempty prefix of the array. So observing during a derivation δ a fixed (say n-th) cell of the array yields the *n-th record of δ*. Depending on what is recorded one gets different kinds of n-th records. If a note is made of the contents of the n-th cell each time this cell is being processed (rewritten) then one gets (a word that is) the *active record* of the cell (during δ). If we collect together *all* active records of the n-th cell during *all successful* derivations of G then we get 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).

It is proved in [EHR] that for each rb grammar G and each positive integer n, $ACT_n(G)$ is regular. Moreover this regularity is so "strong" that the union $\bigcup_{n \in I} ACT_n(G)$, where I is an arbitrary (!) subset of positive integers, is also regular.

In the present paper we consider a different way of recording the use of a memory cell during a computation. We simply record during *each* step of a derivation δ the contents of the n-th cell (where if at a given step a cell to the left of the n-th cell is processed, then we insert the special "$" symbol symbolizing the idle state of the given cell). In this way we get the *full record* of the n-th cell and the collection of all such full records of the n-th cell during *all* successful derivations in G forms the *n-full record language of G*, denoted $FR_n(G)$.

In this paper we prove that, as in the case of active records, $FR_n(G)$ is regular for each n, *but unlike* in the case of active records, infinite unions $\bigcup_{n \in I} FR_n(G)$ do not have to be regular (even if I is taken to be the set of *all* positive integers). Using the representation theorem for cp systems (through rb

grammars) we transfer the above mentioned two results to the level of cp systems (and hence push-down automata).

The conclusion from the results from the present paper and from [EHR] is that the evaluation of the memory behavior (memory use) in rb grammars and push-down automata may differ quite considerably depending on the "observation method".

0. PRELIMINARIES

We assume the reader to be familiar with basic formal language theory (see, e.g., [H] or [S2]). Perhaps the following notational matters deserve some attention.

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 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*.

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 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 : BASIC NOTIONS AND RESULTS

In this section we introduce basic notions concerning right-boundary grammars. Moreover we will formalize two ways of describing the 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 $\delta : x_0 \overset{*}{\underset{G}{\Longrightarrow}} x_n$.

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 $(\delta_1(0), \delta_1(1), \ldots, \delta_1(m), \delta_2(1), \ldots, \delta_2(n))$. ■

Definition 1.3. Let G be a rb grammar and let δ be a derivation in G with $n = lg(\delta)$. The sequence $(\pi_1, \ldots, \pi_n)$ of productions such that $\delta(i-1) \underset{G}{\overset{\pi_i}{\Longrightarrow}} \delta(i)$ 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) The control sequence of a derivation is well-defined : if x directly derives y, then there exists a unique production π such that $x \underset{G}{\overset{\pi}{\Longrightarrow}} y$.

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

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

In order to simplify our notation we will omit the inscription "G" whenever G is understood from the context. Thus we will use $\Longrightarrow$ and $\overset{*}{\Longrightarrow}$ rather than $\underset{G}{\Longrightarrow}$ and $\underset{G}{\overset{*}{\Longrightarrow}}$ respectively.

A derivation δ consists of lines - if we write these lines under each other (adjusted letter-by-letter) then we will get a "column" of all first letters (in all lines of δ), a column of all second letters, etc. Each such column is a record of the "use" of the given memory cell during δ; i.e., the first column records the use of the first memory cell, the second column records the use of the second memory cell, etc. Here we see a rb grammar as a processor of the data structure consisting of a one-way (potentially infinite) array of cells, where the processing (of a specific kind) takes place at the right end of the (nonempty prefix of the) array. So the "column point of view" represents the record of the use

(the behavior) of a particular cell during a particular derivation. If we now single out from a given column word only these letters which correspond to the moments (within the derivation) when the cell is active (its contents is being rewritten) then we get the "active" record of the use of the given cell. This leads us to the following definition.

Definition 1.4. Let $G = (\Sigma, P, \omega)$ be a rb grammar and let $n \in \mathbf{N}^+$.

(1) Let δ be a derivation in G and let $k = lg(\delta)$. A line $\delta(i)$ of δ with $|\delta(i)| = n$ is called an *n-active* line of δ.

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

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

(2) The *n-active language of G*, denoted $ACT_n(G)$, is the language $\{act_n(\delta) \mid \delta : \omega \overset{*}{\underset{G}{\Longrightarrow}} \Lambda\}$. ■

If, for a given derivation δ and for a given $n \in \mathbf{N}^+$, we take the "full" (rather than only active) column description (where if a given line $\delta(i)$ of a derivation δ is shorter than n, then we insert the $\$$ symbol symbolizing the idle state of the given cell), then we get the n-full record of δ. This is formalized as follows.

Definition 1.5. Let $G = (\Sigma, P, \omega)$ be a rb grammar, and let $\$$ be a symbol not in Σ. Let $N \in \mathbf{N}^+$.

(1) Let δ be a derivation in G and let $k = lg(\delta)$. The *n-full record of δ* denoted $fr_n(\delta)$, is the word $\psi_n(\delta(0))\psi_n(\delta(1)) \cdots \psi_n(\delta(k-1))$ where $\psi_n : \Sigma^* \to \Sigma \cup \{\$\}$ is the mapping defined by

$$\psi_n(u) = \begin{cases} u(n) & , |u| \geq n, \\ \$ & , \text{otherwise.} \end{cases}$$

(2) The *n-full record language of G*, denoted $FR_n(G)$, equals

$\{fr_n(\delta) \mid \delta : \omega \overset{*}{\underset{G}{\Longrightarrow}} \Lambda\}$ ■

Remark 1.2. (1) In defining the n-active record and the n-full record of a derivation δ we do not consider the last line of δ. Therefore:

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

Example 1.1. Let $G = (\{A,B,C,D\}, P, AC)$ be the rb grammar with

$P = \{A \to BC,\ B \to \Lambda,\ C \to DC,\ C \to \Lambda,\ D \to \Lambda\}$.

For each pair $k,l \in \mathbf{N}$ there exists a derivation $\delta_{k,l}$ of the form

$(AC, ADC, AD^2C, \ldots, AD^kC, AD^k, \ldots, AD, A, BC, BDC, \ldots, BD^lC, BD^l, \ldots, BD, B, \Lambda)$.

Obviously $fr_1(\delta_{k,l}) = A^{2k+2}B^{2l+2}$.

Since each derivation $\delta : AC \overset{*}{\Longrightarrow} \Lambda$ is of the above form we have $FR_1(G) = (AA)^+(BB)^+$.

Consider $\delta = \delta_{6,3}$. Then

$fr_2(\delta) = CD^{12}\$CD^6\$$,

$fr_5(\delta) = \$^3CD^6\$^7C\$^4$,

$fr_8(\delta) = \$^6C\$^{15}$ and

$fr_n(\delta) = \$^{22}$ for each $n \geq 9$.

It is easily seen that $ACT_1(G) = \{AB\}$, $ACT_2(G) = \{CD, C\}^2$ and

$ACT_n(G) = \{CD, C, \Lambda\}^2$ for each $n \geq 3$. ■

Active records of rb grammars were investigated in [EHR] where the following result has been proved.

Proposition 1.1 Let G be a rb grammar and let $I \subseteq \mathbf{N}^+$. Then $\bigcup_{n \in I} ACT_n(G)$ is a regular language. ∎

Hence not only each n-active language of a rb grammar is regular but also arbitrary (infinite) unions of these languages are regular - this is strong regularity indeed!!

In the rest of this paper we will investigate n-full record languages of rb grammars and it will turn out that these languages do not have this strong regularity property.

2. A RESULT ON SEQUENCES OF NUMBERS

In this section we prove a combinatorial result concerning sequences of integers having subsequences satisfying certain conditions. This result will form the technical basis in the proof of the main result in the next section.

We begin by defining subsequences (of sequences of integers) that will be of interest in this paper.

Definition 2.1. Let $(x_1, x_2, \ldots, x_n)$, $n \geq 1$, be a sequence of nonnegative integers. A subsequence $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$, $k \geq 1$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ is called *nice* if, for each $1 \leq l \leq k$ and each $i_l \leq j \leq i_k$, $x_j \geq x_{i_l}$. A nice subsequence is called *constant* if all its elements are the same (thus $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$). ∎

The following sequence of lemmas will lead us to the main result of this section.

Lemma 2.1. Let $\sigma = (x_1, x_2, \ldots, x_n)$, $n \geq 1$, be a sequence of nonnegative integers. Let $p = \#\{x_i \mid 1 \leq i \leq n\}$ and let $k \geq 1$.
If $n \geq k^p$, then σ has a constant nice subsequence of length k.

Proof. The lemma is obvious if $k = 1$. So we assume that k is fixed and greater than 1 and we will prove the lemma by induction on p, the number of different values in the sequence.

Basis. If $p = 1$, then the sequence itself is nice and constant.

Induction step. Let $p \geq 2$ and assume that the lemma is proved for sequences that have up to $p-1$ different values. We will show that the lemma holds for the sequence σ with p different values.
Consider the subsequence $(x_{j_1}, x_{j_2}, \ldots, x_{j_l})$ of minimal elements of σ. Since this subsequence is nice and constant σ satisfies our lemma if $l \geq k$.

So assume that $l < k$ and consider the (at most k) subsequences

$\sigma_0 = (x_1, \ldots, x_{j_1-1})$, $\sigma_1 = (x_{j_1+1}, \ldots, x_{j_2-1})$, $\ldots$, $\sigma_l = (x_{j_l+1}, \ldots, x_n)$.

Each of these sequences has at most $p-1$ different values, therefore by our induction hypothesis there exists a constant nice subsequence of σ_i, $1 \le i \le l$, and so a constant nice subsequence of σ (of length k) whenever σ_i contains at least k^{p-1} elements.

If each σ_i contains less than k^{p-1} elements, then the length of σ is at most $(l+1)(k^{p-1}-1) + l < k\cdot(k^{p-1}-1) + k = k^p$. Hence if $n \ge k^p$ then one of the subsequences σ_i has at least k^{p-1} elements, so it has a constant nice subsequence of length k.

This proves the lemma. ∎

Lemma 2.2. Let $\sigma = (x_1, x_2, \ldots, x_n)$, $n \ge 1$, be a sequence of nonnegative integers and let $d \in \mathbf{N}^+$ be such that $x_{i+1} - x_i \le d$ for all $1 \le i < n$. Let $k, m \in \mathbf{N}^+$ be such that $m \ge x_1$ and let $r = \max\{x_i \mid 1 \le i \le n\}$.

If $r > m + (k-1)\cdot d$, then there exists a nice subsequence $(x_{i_1}, \ldots, x_{i_k})$ of σ such that $x_{i_1} > m$.

Proof.

Let σ, d, r and m be as specified in the statement of the lemma and assume that $r > m + (k-1)\cdot d$.

We construct a nice subsequence $(x_{i_1}, x_{i_2}, \ldots, x_{i_p})$ inductively as follows.

Let t be the index of a maximal element of σ, thus $x_t = r$, and we choose i_1 as the minimal index such that $x_l \ge m$ for each $i_1 \le l \le t$.

Hence $i_1 = \min\{i \mid x_l > m \text{ for each } i \le l \le t\}$ and, for $s \ge 1$, we set

$i_{s+1} = \min\{i \mid i_s < i \le t \text{ and } x_l \ge x_i \text{ for each } i \le l \le t\}$

The construction terminates when

$\{i \mid i_s < i \le t \text{ and } x_l \ge x_i \text{ for each } i \le l \le t\} = \emptyset$ (or, equivalently, when $i_s = t$).

We observe that $x_{i_{s+1}} = \min\{x_l \mid i_s < l \le t\}$. This is seen as follows. Let x_{l_0} be a minimal element in the set $\{x_l \mid i_s < l \le t\}$ and assume that $x_{i_{s+1}} > x_{l_0}$. Then obviously $i_s < l_0 < i_{s+1}$, because $x_l \ge x_{i_{s+1}}$ for each $i_{s+1} \le l \le t$. But $x_l \ge x_{l_0}$ for each $l_0 \le l \le t$, because $x_{l_0} = \min\{x_l \mid i_s < l \le t\}$. This contradicts the minimality of i_{s+1}. Thus $x_{i_{s+1}} = \min\{x_l \mid i_s < l \le t\}$.

Hence, informally speaking, $x_{i_{s+1}}$ is the first minimal element of $(x_{i_s+1}, \ldots, x_t)$.

The sequence $(x_{i_1}, \ldots, x_{i_q})$ constructed as above is obviously a nice subsequence of σ. In order to prove the lemma we still have to show that $q \ge k$.
This follows from the following observations.

Claim 1. (a) $x_{i_1} \le m + d$,

(b) $x_{i_{s+1}} - x_{i_s} \le d$ for $1 \le s < q$.

Proof of Claim 1.

(a) This is clear if $i_1 = 1$. Otherwise (for $i_1 > 1$) $x_{i_1} \le x_{i_1-1} + d \le m + d$, because $x_{i_1-1} \le m$.

(b) Assume that $x_{i_{s+1}} - x_i > d$. This implies that $x_{i_s+1} \le x_{i_s} + d < x_{i_{s+1}}$, which contradicts the minimality of $x_{i_{s+1}}$ in $(x_{i_s+1}, \ldots, x_t)$. ∎

Thus $m + (k-1)\cdot d < r = x_t = x_{i_q} \le m + d + (q-1)\cdot d = m + q\cdot d$. Consequently $q > k-1$, and so $q \ge k$. This proves the lemma. ∎

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

Theorem 2.3. Let $\sigma = (x_1, x_2, \ldots, x_n)$, $n \ge 1$, be a sequence of nonnegative integers and let $m, d \in \mathbf{N}^+$ be such that $m \ge x_1$ and $x_{i+1} - x_i \le d$ for all $1 \le i < n$. Let $k \in \mathbf{N}^+$.
If $n \ge k^{m+(k-1)\cdot d}$, then there exists a nice subsequence $(x_{i_1}, \ldots, x_{i_k})$ of σ such

that either this sequence is constant or $x_{i_1} > m$.

Proof. Let σ, d and m be as in the statement of the lemma. Define $r = \max\{x_i \mid 1 \le i \le n\}$ and $p = \#\{x_i \mid 1 \le i \le n\}$.

If $r > m + (k-1)\cdot d$, then, according to Lemma 2.2, σ has a nice subsequence $(x_{i_1}, \ldots, x_{i_k})$ such that $x_{i_1} > m$.

On the other hand, if $r \le m + (k-1)\cdot d$, then $n \ge k^{m+(k-1)\cdot d}$ implies that $n \ge k^p$ (because $r > p$).

Thus in this case Lemma 2.1 implies that σ has a constant nice subsequence of length k. ∎

3. THE LENGTH OF DERIVATIONS IN RB GRAMMARS

For a rb grammar G and two words u, v over its alphabet the (u,v)-*spectrum* is the set of *all* lengths of *all* derivations in G leading from u to v. In this section we will prove that the (u,v)-spectrum is ultimately periodic for all pairs (u,v). This result certainly says something about the nature of derivations in rb grammars. Moreover it will be an essential tool in proving the properties of full record languages in the next section.

We begin by formally defining spectra.

Definition 3.1. Let $G = (\Sigma, P, \omega)$ be a rb grammar. For two words $u, v \in \Sigma^*$ the *(u,v)-spectrum (in G)*, denoted $spec_G(u,v)$, is defined by

$$spec_G(u,v) = \{lg(\delta) \mid \delta : u \overset{*}{\underset{G}{\Longrightarrow}} v\}. \blacksquare$$

As usual we will omit the index G whenever G is clear from the context.

To prove our result on spectra we need the following lemma.

Lemma 3.1. Let $G = (\Sigma, P, \omega)$ be a rb grammar and let $u, v \in \Sigma^*$. There exist constants n_0 , $q_0 \in \mathbf{N}$ satisfying:
if $n \in spec(u,v)$ with $n \geq n_0$, then $\{n + c_n \cdot k \mid k \in \mathbf{N}\} \subseteq spec(u,v)$ for some $c_n \in \mathbf{N}$ with $1 \leq c_n \leq q_0$.

Proof.

The main idea of this proof (although expressed in combinatorial terms) is rather classical: one "pumps" certain subderivations in a given derivation.

Let $G = (\Sigma , P , \omega)$ be a rb grammar and let $u, v \in \Sigma^*$.
Set $n_0 = k_0^{m_0+(k_0-1)\cdot d_0}$, where $k_0 = \#\Sigma+1$, $m_0 = \max\{|u|, |v|\}$ and $d_0 = maxr(G)-1$.

Furthermore set $q_0 = n_0 + \gamma_0 \cdot n_0 \cdot d_0$, where

$$\gamma_0 = \max\{ \min\{lg(\delta) \mid \delta : A \overset{*}{\Longrightarrow} \Lambda\} \mid A \in \Sigma \text{ and } A \overset{*}{\Longrightarrow} \Lambda\}.$$

Note that this implies that if $z \overset{*}{\Longrightarrow} \Lambda$, then there exists a derivation $\delta : z \overset{*}{\Longrightarrow} \Lambda$ with $lg(\delta) \leq \gamma_0 \cdot |z|$.

We will show that the lemma holds for the above choice of n_0 and q_0.

Take $n \in spec(u,v)$ with $n \geq n_0$. (If no such n exists, then our lemma trivially holds.) Let $\delta : u \overset{*}{\underset{G}{\Longrightarrow}} v$ be an arbitrary derivation such that $lg(\delta) = n$; thus $u = \delta(0)$ and $v = \delta(n)$.

Consider now the sequence $\sigma = (|\delta(0)|, |\delta(1)|, \ldots, |\delta(n_0)|)$ of the lengths of the first n_0+1 successive lines in δ. Since obviously $|\delta(i+1)| - |\delta(i)| \leq d_0$ for all $0 \leq i \leq n_0$ and since $n_0 \geq k_0^{m_0+(k_0-1)\cdot d_0}$, where $m_0 \geq |\delta(0)|$, by Theorem 2.3 there exists a nice subsequence $\tau = (|\delta(i_1)|, \ldots, |\delta(i_{k_0})|)$ of σ such that either $|\delta(i_1)| = \cdots = |\delta(i_{k_0})|$ or $|\delta(i_1)| > m_0$.

This subsequence contains $k_0 = \#\Sigma+1$ elements, thus there exist two indices $s, t \in \{i_1, i_2, \ldots, i_{k_0}\}$, $s < t$, such that $last(\delta(s)) = last(\delta(t))$.

We consider separately two cases.

(A) $|\delta(s)| = |\delta(t)|$.

We can split δ into three subderivations $\delta_1 = (\delta(0), \ldots, \delta(s))$, $\delta_2 = (\delta(s), \ldots, \delta(t))$ and $\delta_3 = (\delta(t), \ldots, \delta(n))$.

All lines of δ_2 are at least as long as $\delta(s)$ (because τ is a nice subsequence) thus they all have a common prefix x of length $|\delta(s)|-1$: rewritings take place at positions greater than or equal to $|\delta(s)|$.

On the other hand $|\delta(t)| = |\delta(s)|$ and $last(\delta(t)) = last(\delta(s))$. This implies that $\delta(s) = \delta(t)$. Consequently δ_2 can be repeated within δ any number of times without influencing the other lines of the derivation.

Hence if we set $c_n = lg(\delta_2) = t-s$, then for any $k \in \mathbf{N}$ there exists a derivation $\delta^{(k)} : u \overset{*}{\Longrightarrow} v$ of length $n + k \cdot c_n$. Note that $c_n \leq n_0 \leq q_0$.

(B) $|\delta(s)| < |\delta(t)|$.

In this case τ is obviously not a constant subsequence, thus the condition $|\delta(i_1)| > m_0$ for τ holds. Hence $|\delta(s)| \geq |\delta(i_1)| > m_0 \geq |v|$.

Since all lines between $\delta(s)$ and $\delta(t)$ are at least as long as $\delta(s)$ they all have a common prefix x of length $|\delta(s)|-1$.

Moreover $\delta(s)$ and $\delta(t)$ have the same last letter so we may write

$\delta(s) = xA$, $\delta(t) = xzA$ where $x, z \in \Sigma^*$, $A \in \Sigma$ with $|x| \geq |v|$ and $|z| \geq 1$.

In general, in a single derivation step, the length of a line may decrease by at most one. Since $|\delta(t)| > |x| \geq |\delta(n)| = |v|$ there exists a line $\delta(p)$, $t < p \geq n$ such that $|\delta(p)| = |x|$.

If we assume that p is chosen to be minimal, then $\delta(p) = x$ because in that case all lines of δ between $\delta(t)$ and $\delta(p)$ are longer than $\delta(p)$.

In the same way we find a line $\delta(p_1) = xz$ with $t < p_1 < p$.

Now we split δ into five parts as follows:

$\delta_1 = (\delta(0), \ldots, \delta(s) = xA)$, $\delta_2 = (\delta(s), \ldots, \delta(t) = xzA)$, $\delta_3 = (\delta(t), \ldots, \delta(p_1) = xz)$, $\delta_4 = (\delta(p_1), \ldots, \delta(p) = x)$ and $\delta_5 = (\delta(p), \ldots, \delta(n))$.

This may be depicted as follows.

Figure 1.

We make the following observations concerning z and A.

(1) Using a derivation δ_2' with $cont(\delta_2') = cont(\delta_2)$ we derive zA from A. Then $lg(\delta_2') = lg(\delta_2) = t-s \leq n_0$, hence $|zA| \leq |A| + d_0 \cdot lg(\delta_2) \leq 1 + d_0 \cdot n_0$ or equivalently $|z| \leq d_0 \cdot n_0$.

(2) Using a derivation δ_3' with $cont(\delta_3') = cont(\delta_3)$ we derive Λ from A.

(3) Using a derivation δ_4' with $cont(\delta_4') = cont(\delta_4)$ we derive Λ from z.

Let $\mu : z \overset{*}{\Longrightarrow} \Lambda$ be such that all other derivations of Λ from z are at least as long as μ. Then $lg(\mu) \leq \gamma_0 \cdot |z| \leq \gamma_0 \cdot n_0 \cdot d_0$.

Set $c_n = lg(\delta_2) + lg(\mu)$; it is clear that $c_n \leq q_0$.

Figure 2.

It is obvious that for each $k \in \mathbf{N}^+$ there exists a derivation $\delta(k)$ of the form $(u, \ldots, xA, \ldots, xzA, \ldots, xz^2A, \ldots, xz^kA, \ldots, xz^k, \ldots, xz, \cdots, x, \ldots, v)$ such that $cont(\delta^{(k)}) = cont(\delta_1) \cdot cont(\delta_2) \cdot cont^{k-1}(\delta_2) \cdot cont(\delta_3) \cdot cont^{k-1}(\mu) \cdot cont(\delta_4) \cdot cont(\delta_5)$.

Figure 3.

Thus $lg(\delta^{(k)}) = lg(\delta) + (k-1) \cdot (lg(\delta_2) + lg(\mu)) = n_0 + (k-1) \cdot c_n \in spec(u,v)$ for each $k \in \mathbf{N}^+$.

This proves the lemma. ∎

The above lemma implies directly the main result of this section.

Theorem 3.2. Let $G = (\Sigma, P, \omega)$ be a rb grammar and let $u, v \in \Sigma^*$. Then $spec(u,v)$ is an ultimately periodic set.

Proof.

Let $u \in \Sigma^*$ and let n_0, q_0 be constants satisfying the statement of Lemma 4.2. Let $\bar{c}$ be a common multiple of the numbers $1, 2, \ldots, q_0$. Thus, if $n \in spec(u,\Lambda)$ with $n \geq n_0$, then $\{n + \bar{c} \cdot k \mid k \in \mathbf{N}\} \subseteq \{n + c_n \cdot k \mid k \in \mathbf{N}\} \subseteq spec(u,\Lambda)$ for some $c_n \in \mathbf{N}$ with $1 \leq c_n \leq q_0$.

Let $I = \{n \mid n \in spec(u,\Lambda) \text{ and } n < n_0\}$ and $P = \{n \mid n \in spec(u,\Lambda),\ n \geq n_0$ and $n - \bar{c} \notin spec(u,\Lambda)\}$.

Then I and P are finite sets (elements of P are all different modulo $\bar{c}$) and consequently $spec(u,\Lambda) = I \bigcup \{n + k \cdot \bar{c} \mid n \in P \text{ and } k \in \mathbf{N}\}$ is an ultimately periodic set. ■

4. REGULARITY OF FULL RECORD LANGUAGES.

We are ready now to say something about the nature of full record languages. It will turn out that

(1) *as* in the case of active records, $FR_n(G)$ is regular for each $n \in \mathbf{N}^+$,

(2) *unlike* in the case of active records an arbitrary union of $FR_n(G)$ languages do not have to be regular.

To simplify the notation throughout the section we will consider an arbitrary but fixed rb grammar $G = (\Sigma, P, \omega)$.

Let $u,v \in \Sigma^*$ and $n \in \mathbf{N}^+$. We distinguish a set of special derivations.

$G_n(u,v) = \{\delta : u \overset{+}{\Longrightarrow} v \mid \text{for each } 0 < i < lg(\delta)\ ,\ |\delta(i)| > n\}$.

The corresponding set of full records is denoted by

$FRG_n(u,v) = \{fr_n(\delta) \mid \delta \in G_n(u,v)\}$.

In order to prove that, for each $n \in \mathbf{N}^+$, $FR_n(G)$ is regular we will first prove that the above set of full descriptions is regular.

Lemma 4.1. Let $u,v \in \Sigma^*$ and let $n \in \mathbf{N}^+$. Then $FRG_n(u,v)$ is regular.

Proof.

First we observe that if $|v| < n$, then $G_n(u,v)$ contains only derivations of length 1. For assume that $\delta \in G_n(u,v)$ with $k = lg(\delta)$. Then $|\delta(k-1)| \leq |\delta(k)|+1 = |v|+1 < n+1$ which contradicts the condition $|\delta(k-1)| \geq n+1$ for $k-1 > 0$.

Thus in this case $G_n(u,v)$ is at most a singleton and $FRG_n(u,v)$ is obviously regular.

So we may assume that $|v| \geq n$. Let $A = v(n)$ and let $v = xz$ with $|x| = n$. Let $\delta \in G_n(u,v)$ with $k = lg(\delta) > 1$. Since $|\delta(i)| > n$ for every $1 \leq i < k$ we can write $\delta(i) = xw_i$ with $w_i \overset{*}{\Longrightarrow} z$. But then

$fr_n(\delta) = u(n)A^{k-1}$, if $|u| \geq n$, and $fr_n(\delta) = \$A^{k-1}$, if $|u| < n$.

Thus $FRG_n(u,v) = \sigma\{A^l \mid l \in spec(w,z)$ for some $w \in \Sigma^*$ such that $u \Longrightarrow xw\}$, where σ equals $u(n)$ if $|u| \geq n$ and $\$$ otherwise. This language is regular because by Theorem 3.2 $spec(u,v)$ is an ultimately periodic set.

This completes the proof of Lemma 4.1. ▪

Now we are ready to prove the regularity of $FR_n(G)$.

Theorem 4.2. $FR_n(G)$ is a regular language for every $n \in \mathbf{N}^+$.

Proof.

Let $\delta : \omega \overset{*}{\Longrightarrow} \Lambda$ and let $(\delta(i_1), \ldots, \delta(i_t))$ be the subsequence consisting of all lines $\delta(i)$ with $0 < i < lg(\delta)$ and $|\delta(i)| \leq n$.

Then δ is composed of $t+1$ derivations $\delta_r = (\delta(i_r), \ldots, \delta(i_{r+1}))$ $r = 0, 1, \ldots, t$ (where we set $i_0 = 0$ and $i_{t+1} = lg(\delta)$), such that $fr_n(\delta) = fr_n(\delta_0) \cdots fr_n(\delta_t)$. From the construction of the sequence $(\delta(i_1), \ldots, \delta(i_t))$ it follows that $\delta_r \in G_n(\delta(i_r), \delta(i_{r+1}))$ for $r = 0, 1, \ldots, t$.

Consider now the finite automaton $\mathbf{A}_n$ defined as follows.

The set of states of $\mathbf{A}_n$ equals the set $\Sigma^{\leq n}$ of all nonempty words of length at most n over Σ together with ω and Λ, which are the initial and the final state of $\mathbf{A}_n$ respectively.

For each pair $x, y \in \Sigma^{\leq n}$, $\mathbf{A}_n$ has an edge from x to y labelled by a special symbol $[x,y]$.

Furthermore there are edges from ω to x labelled by $[\omega, x]$ for each $x \in \Sigma^{\leq n} \cup \{\Lambda\}$ and edges from y to Λ labelled by $[y, \Lambda]$ for each $y \in (\Sigma^{\leq n} - \{\Lambda\}) \cup \{\omega\}$.

Note that ω has outgoing edges only (except when $|\omega| = n$) and Λ has incoming edges only.

Let π be the regular substitution from the set Θ of labels of $\mathbf{A}_n$ into the set $(\Sigma \cup \{\$\})^*$ defined as follows:

$\pi([x,y]) = FRG_n(x,y)$ for each $x \in \Sigma^n \cup \{\omega\}$ and each $y \in \Sigma^n \cup \{\Lambda\}$.

Then from the above considerations it easily follows that $FR_n(G) = \pi(\mathbf{A}_n)$. $FR_n(G)$ is regular because it is a regular substitution of a regular language. Consequently the theorem holds. ■

We end this section by demonstrating that, unlike in the case of active records, an arbitrary union of $FR_n(G)$ languages does not have to be regular. As a matter of fact we exhibit a rb grammar G for which $\bigcup_{n \in \mathbf{N}^+} FR_n(G)$ is not regular.

Theorem 4.3. There exists a rb grammar G such that $\bigcup_{n=1}^{\infty} FR_n(G)$ is not regular.

Proof.

Let $G = (\{A,B\}, P, A)$ be the rb grammar such that

$P = \{A \to BA, A \to \Lambda, B \to \Lambda\}$.

All derivations from A to Λ are of the form

$\delta_k = (A, BA, B^2A, \ldots, B^kA, B^k, B^{k-1}, \ldots, B, \Lambda)$

for some $k \in \mathbf{N}$; then obviously $lg(\delta_k) = 2k+1$ and $|\delta_k(i)| \leq k+1$ for each $1 \leq i \leq 2k+1$.

Thus $fr_n(\delta_k) = \$^{2k+1}$ for each $n > k+1$.

For all $n \leq k+1$ we have $fr_n(\delta_k) = \$^{n-1}AB^{2(k+1-n)}\$^{n-1}$.

Consequently $\bigcup_{n=1}^{\infty} FR_n(G) \cap \$^*A\$^* = \{\$^nA\$^n \mid n \in \mathbf{N}\}$.

Since $\{\$^nA\$^n \mid n \in \mathbf{N}\}$ is not regular and the class of regular languages is closed

under intersections, $\bigcup_{n=1}^{\infty} FR_n(G)$ is not regular. ■

Remark 4.1. (1) If (for the rb grammar from the proof of the previous theorem) we consider *arbitrary* unions $\bigcup_{n \in M} FR_n(G)$ for $M \subseteq \mathbf{N}^+$, then we can get even nonrecursive languages (by taking M nonrecursive):

$$\bigcup_{n \in M} FR_n(G) \cap \$^* A \$^* = \{fr_n(\delta_k) \mid n \in M, k = n-1\} = \{\$^{n-1} A \$^{n-1} \mid n \in M\}.$$

(2) In general, given a rb grammar G, $\bigcup_{n=1}^{\infty} FR_n(G)$ does not have to be even context-free. An example of such a situation is the rb grammar given in Example 1.1. We have then $fr_n(\delta_{k,l}) \in \$^* C \$^* C \$^*$ if and only if $n = k+2 = l+2$.

Since $fr_{k+2}(\delta_{k,k}) = \$^k C \$^{2k+1} C \$^{k+1}$, we have

$$\bigcup_{n=1}^{\infty} FR_n(G) \cap \$^* C \$^* C \$^* = \{\$^k C \$^{2k+1} C \$^{k+1} \mid k \in \mathbf{N}\}$$

which is not a context-free language. Thus, because the class of context-free languages is closed under intersections with regular languages, $\bigcup_{n=1}^{\infty} FR_n(G)$ is not context-free. ■

5. COORDINATED PAIR SYSTEMS AND RB GRAMMARS

Right boundary grammars form a very basic building block in the general theory of grammars and automata presented in [R]. In particular, within this theory a push-down automaton is seen as a pair of "cooperating grammars", the first one rl (modelling the input and the finite state control) and the other one rb (modelling the infinite push-down store); such a pair is called a coordinated pair system.

In this section we will "transfer" our results concerning the (active and full records of the) 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). In this way investigating the use of memory in rb grammars is being used for learning about the use of memory in push-down automata.

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

Definition 5.1. A *coordinated pair system*, *cp system* for short, is triple $G = (G_1 , G_2 , R)$, where

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

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

$R \subseteq P_1 \times P_2$, the set of *rewrites* of G. ■

Definition 5.2. 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 $x_1 \overset{\pi_1}{\underset{G_1}{\Longrightarrow}} y_1$ and $x_2 \overset{\pi_2}{\underset{G_2}{\Longrightarrow}} y_2$ for a rewrite $\pi = (\pi_1 , \pi_2) \in R$, then we say that (x_1 , x_2) *directly computes* (y_1 , y_2) *in* G *using* π and we denote this by $(x_1 , x_2) \overset{\pi}{\underset{G}{\Longrightarrow}} (y_1 , y_2)$.

$\overset{*}{\underset{G}{\Longrightarrow}}$ denotes the reflexive and transitive closure of $\underset{G}{\Longrightarrow}$. If $(x_1 , x_2) \overset{*}{\underset{G}{\Longrightarrow}} (y_1 , y_2)$, 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 $x_0 = (S_1, S_2)$ and $x_n = (w, \Lambda)$ for some $w \in \Delta^*$, then ρ is called *successful*.

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

The formal notions describing the use of memory in rb grammars are extended to cp systems in an obvious way.

Definition 5.3. 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 δ, 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

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

(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 $w \in \Delta^*\}$. ■

Definition 5.4. 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-full record* of δ, denoted $fr_n(\rho)$, is the word $\psi_n(\rho(0))\, \psi_n(\rho(1)) \cdots \psi_n(\rho(k-1))$, where $\psi_n : \Sigma_1^* \times \Sigma_2^* \to \Sigma_2 \cup \{\$\}$ is the mapping defined by

$$\psi_n((u,v)) = \begin{cases} v(n) \text{ , if } |v| \leq n, \\ \$ \quad \text{ , otherwise.} \end{cases}$$

(2) The *n-full record language* of G, denoted $FR_n(G)$, is the language $\{fr_n(\rho) \mid \rho : (S_1 , S_2) \Longrightarrow (w,\Lambda) \text{ for some } w \in \Delta^* \}$. ■

Since in this paper we are mainly interested in the behavior of the second component of successful computations in a cp system we introduce the notion of an *internal* cp system. A cp system is internal if it has only chain-rules (i.e. of the form $A \to B$) and Λ-rules (i.e. of the form $A \to \Lambda$) on its first component. If, for a give cp system G, we erase all terminal symbols in all productions of the first component (and in the corresponding rewrites), then we obtain the *internal version* of G.

Definition 5.5. 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

$\tilde{P} = \{(X \to Z) \mid (X \to wZ) \in P \text{ for some } X \in \Sigma_1 - \Delta, w \in \Delta^* \text{ and } Z \in (\Sigma_1 - \Delta) \cup \{\Lambda\}\}$ and $\tilde{R} = \{(X \to Z, \pi_2) \mid (x \to wZ, \pi_2) \in R \text{ for some } X \in \Sigma_1 - \Delta, w \in \Delta^* \text{ and } y \in (\Sigma_1 - \Delta) \cup \{\Lambda\}\}$. ■

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 5.1. For each cp system G and each $n \in \mathbf{N}^+$,

$ACT_n(G) = ACT_n(int(G))$ and

$FR_n(G) = FR_n(int(G))$. ■

We have reduced the considerations concerning (the use of memory in) cp systems to considerations concerning internal cp systems. It is possible to further reduce the problem: rather than to consider internal cp systems it suffices to consider rb grammars.

The following definitions formalize how to represent (successful computations in) an internal cp system by (successful derivations in) a suitable rb grammar. Using this notion of representation we will transfer the results on the regularity of the (full) use of memory in rb grammars to (arbitrary) cp systems.

Definition 5.6. Let Θ, Σ_1 and Σ_2 be alphabets and let $\psi : \Theta^* \to \Sigma_1^*$ and $\varphi : \Theta^* \to \Sigma_2$ be 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) $Z = \Lambda$ if $u = \Lambda$ and $Z = \psi(last(u))$ otherwise.

(ii) $w = \varphi(u)$. ∎

Definition 5.7. 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 $H = (\Theta, Q, T)$, $T \in \Theta$, be a rb grammar. Finally let $\psi : \Theta^* \to \Sigma_1^*$ and $\varphi : \Theta^* \to \Sigma_2^*$ be codings.

(1) Let ρ be a computation in G and let δ be a derivation in H. 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 if the following holds.

(i) For each successful computation ρ in G there exists 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. The notion of the representation we have defined above is more narrow than the one defined in [EHR]. In the present paper we are concerned only with *successful* computations, while in [EHR] we have used the notion of a representation that was suitable for *all*, i.e. successful and not successful, computations. However the representation we use here suffices for transferring our results (on the use of memory) from the level of rb grammars to the level of cp systems. ▪

From Theorem 5.1 in [EHR] (and its proof) we get the following result.

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

Now Proposition 1.1, Theorem 4.2 and Theorem 4.3 yield the following results as their corollaries.

Theorem 5.3. Let G be a cp system. For each $n \geq 1$ both $ACT_n(G)$ and $FR_n(G)$ are regular.

Proof.

For $ACT_n(G)$ this result was proved in [EHR].

We consider here $FR_n(G)$. Let H be a rb grammar that (ψ,φ)-represents $int(G)$ for some codings ψ and φ. The coding φ maps Θ (the alphabet of H) into Σ_2 (the alphabet of the second component of G). We extend φ to the coding $\varphi_{\$}$ mapping $\Theta \cup \{\$\}$ into $\Sigma_2 \cup \{\$\}$ by defining $\varphi_{\$}(A) = \varphi(A)$ for $A \in \Theta$ and $\varphi_{\$}(\$) = \$$.

Then it is easily seen that for every $n \in \mathbf{N}^+$ $fr_n(\rho) = \varphi_{\$}(fr_n(\delta))$, where ρ is a successful derivation in H such that $\delta[\psi,\varphi>\rho$.

Hence, by Lemma 5.1, $FR_n(G) = FR_n(int(G)) = \varphi(FR_n(H))$ and consequently $FR_n(G)$ is regular since $FR_n(G)$ is regular and the class of regular languages is closed under homomorphisms. ▪

Theorem 5.4.

(1) For each cp system G and each $I \subseteq \mathbf{N}^+$, $\bigcup_{n \in I} ACT_n(G)$ is regular.

(2) There exists a cp system G such that $\bigcup_{n \in \mathbf{N}} FR_n(G)$ is not regular.

Proof.

(1) This is Theorem 5.3 in [EHR].

(2) Every rb grammar $H = (\Sigma, P, A)$ with $A \in \Sigma$ can be transformed in a natural way into a cp system by "adding" a (dummy) first component with one nonterminal only.

Formally, let $H_{cp} = (G_1, G_2, R)$ be the (internal) cp system with $G_1 = (\{S\}, P_1, S, \emptyset)$, $P_1 = \{S_1 \to S_1, S_1 \to \Lambda\}$, $G_2 = H$ and $R = P_1 \times P$.
One easily verifies that H_{cp} is (ψ,φ)-represented by H, where $\psi : \Sigma^* \to \{S\}^*$ is the coding that maps each element of Σ to S and $\varphi : \Sigma^* \to \Sigma^*$ is the identity on Σ^*. Thus, for $n \in \mathbf{N}^+$, $FR_n(H_{cp}) = FR_n(H)$.

If we construct in this way G_{cp} for the rb grammar G given in the proof of Theorem 4.3, then $\bigcup_{n=1}^{\infty} FR_n(G_{cp}) = \bigcup_{n=1}^{\infty} FR_n(G)$, which is not regular. ∎

Remark 5.2. The results concerning rb grammars mentioned in Remark 4.1 carry over to cp systems using the construction described in the proof of Theorem 5.4.(2). Thus in general, given a rb grammar G, $\bigcup_{n=1}^{\infty} FR_n(G)$ does not have to be context-free. Moreover if we take arbitrary unions $\bigcup_{n \in M} FR_n(G)$ for $M \subseteq \mathbf{N}^+$, then we can get arbitrarily complex languages. ∎

DISCUSSION

In this paper we have continued the investigation into the use of memory in rb grammars and cp systems (push-down automata). Together with [EHR] the present paper leads to certain conclusions.

(1) Local (i.e., involving one, or a finite number of memory cells) observations of the memory behavior (during successful computations) lead to "regular results": both, $ACT_n(G)$ and $FR_n(G)$ are regular for each rb grammar G and each positive integer n.

(2) Global (i.e., involving an infinite number of memory cells) observations of the memory behavior lead to regular results if one makes active records only but may yield not regular (even "arbitrarily complex") results if one makes full records.

Also, the two papers together have provided deeper insight into the nature of rb grammars as well as technical tools to deal with them (e.g., the ultimate periodicity of spectra theorem). We hope that we have also demonstrated that the cp systems formalism to deal with push-down automata is mathematically very convenient: we can often shift the burden of work to the investigation of rb grammars and then use "transfer theorems" to transfer the results to the level of cp systems.

Certainly many problems and problem areas concerning the use of memory in rb grammars and cp systems remain open.

As far as technical problems are concerned it would be interesting to know how complicated can be the languages of the form $\bigcup_{n \in I} FR_n(G)$ even if $I = \mathbf{N}$. To this aim it would be very desirable to learn more about the structure of $FR_n(G)$ languages. In [EHR] we have provided a structural characterization of $ACT_n(G)$ languages and this characterization has allowed us to prove that $\bigcup_{n \in I} ACT_n(G)$ is

regular for every $I \subseteq \mathbf{N}^+$.

It seems now to be very natural to consider still different ways of recording the use of memory in rb grammars and cp systems. Are there methods of observation (ways of recording) that will yield non-regular languages for single memory cell? Which of them yield "strong regularity"? Are there "easy translations" of different sorts of "record languages" into each other?

These are all very interesting questions which (in our opinion) should be investigated if we want to understand the way the memory is used in rb grammars and push-down automata.

ACKNOWLEDGEMENTS

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

REFERENCES

[B] Buchi, J.R., "Regular canonical systems", *Arch. Math. Logik und Grundlagenforsch.*, v.6, pp. 91-111, 1964.

[ER] Ehrenfeucht, A. and Rozenberg, G., "A note on regular canonical systems," Techn. Rep. No. CU-CS-279-84, Dept. of Comp. Science, Univ. of Colorado at Boulder, 1984.

[EHR] Ehrenfeucht, A., Hoogeboom, H.J. and Rozenberg, G., "On the active use of memory in right-boundary grammars and push-down automata", Techn. Rep. No. CU-CS-298-85, Dept. of Comp. Science, Univ. of Colorado at Boulder, 1985.

[G] Greibach, S., "A note on push-down store automata and regular systems", *Proc. Amer. Math. Soc.*, x.18, pp. 263-268, 1967.

[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.

[S1] Salomaa, A., *Theory of automata*, Pergamon Press, Oxford, 1969.

[S2] Salomaa, A., *Formal languages*, Academic Press, London-New York, 1973.

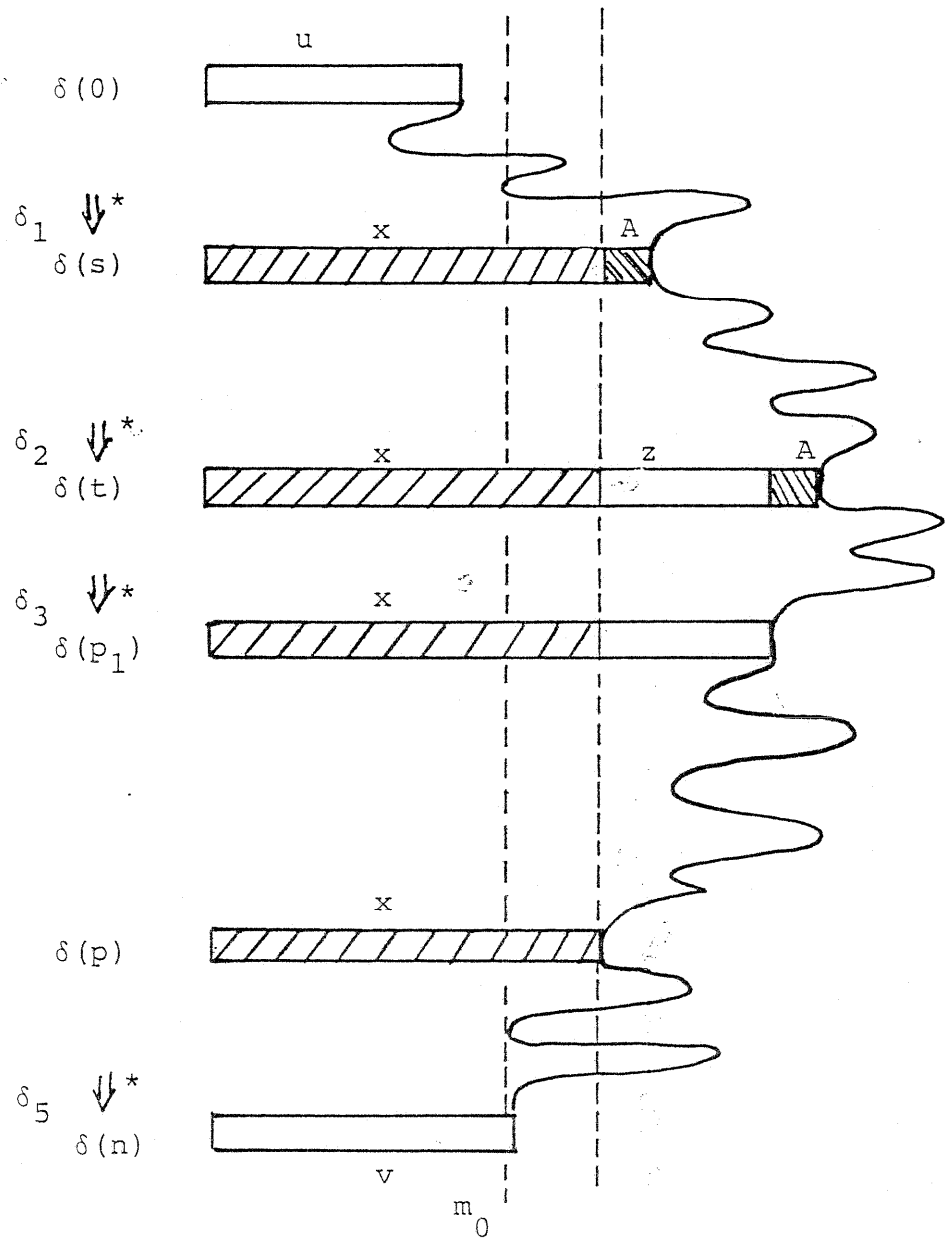

Figure 1

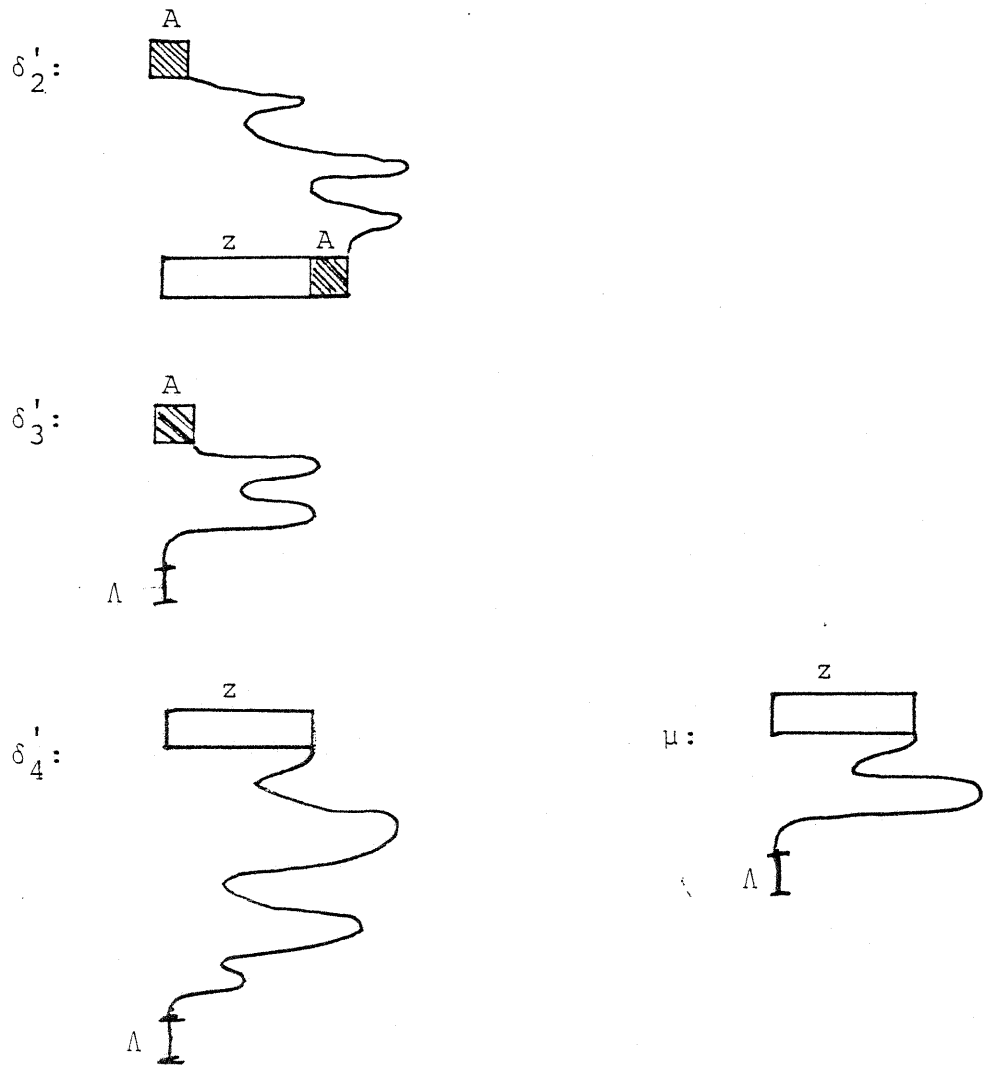

Figure 2

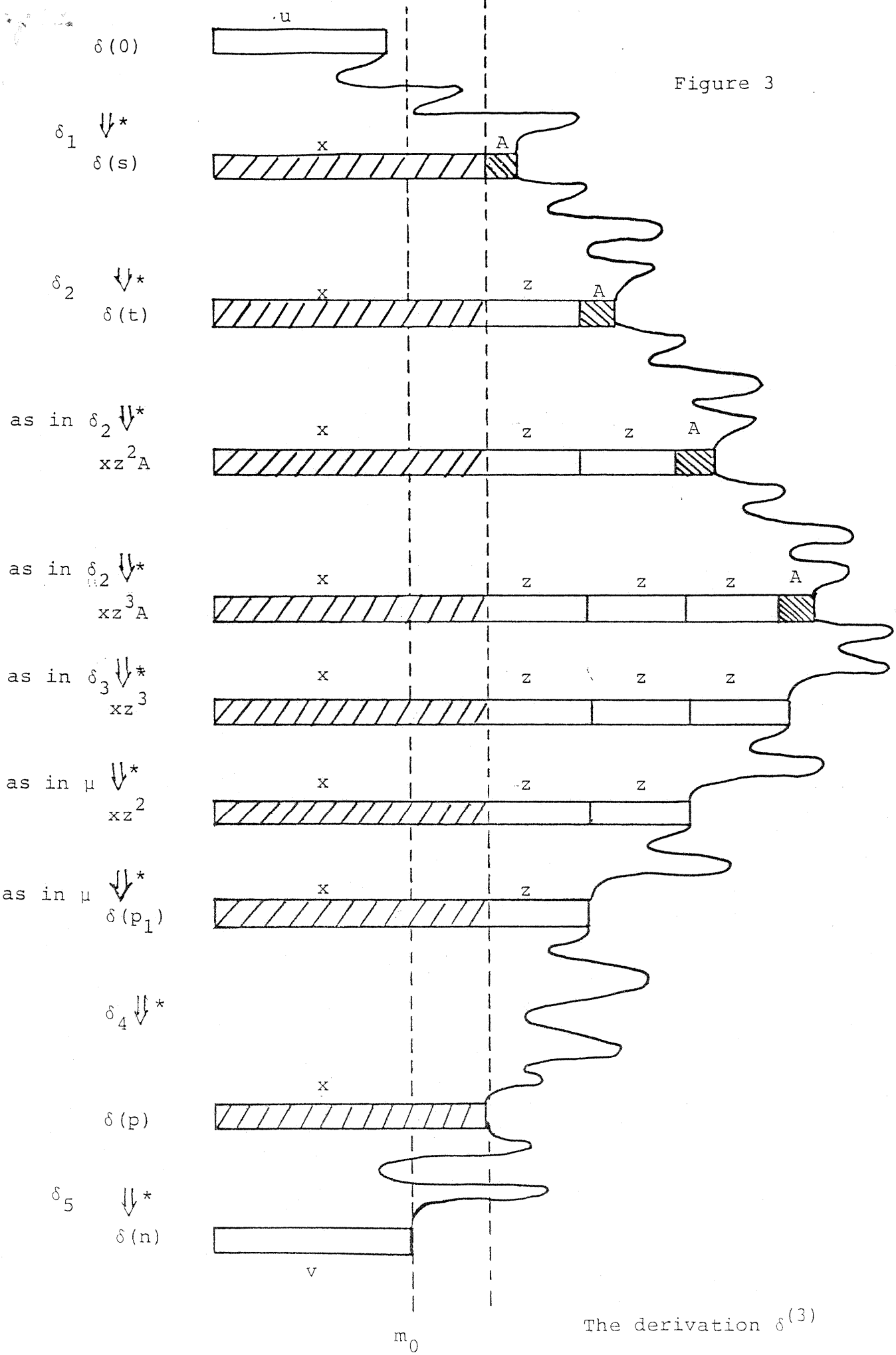

Figure 3

The derivation $\delta^{(3)}$