COMMUTATIVE GRAMMARS

S. CRESPI-REGHIZZI [1], D. MANDRIOLI [1]

ABSTRACT - Commutative grammars are a formalism for generating bags, equivalent to vector addition systems and Petri nets. Known results are recalled and new one are presented on reachability and boundedness. In particular some subclasses of commutative grammars are introduced which admit a positive answer to these problems, and generate semilinear languages. Finally the equivalence, via Parikh mapping, of commutative and matrix grammars is proven.

1. Introduction.

In this work we introduce commutative phrase structure grammars as a device for generating sets of bags (commutative languages) rather than strings.

This formalism is shown to be equivalent to vector addition systems [1], and to Petri nets [2, 3, 4, 5] two well known models of parallel processing.

Commutative grammars are also quite close to the transformation expression of [6].

Using this approach some relevant problems of Petri nets, namely reachability, liveness, safety and boundedness are discussed. Moreover the semilinearity of commutative languages is introduced as a tool for studying the periodical behaviour of a system.

Some known results on boundedness are recalled in the new formalism.

New results are obtained on the decidability and semilinearity of bounded languages. Furthermore we introduce modular grammars as a subclass of commutative grammars which generate unbounded and semilinear languages.

Received: 5 th May 1975.

[1] Istituto di Elettrotecnica ed Elettronica del Politecnico e Centro C N R per l'Ingegneria dei Sistemi per l'Elaborazione dell'Informazione - Milano, Piazza L. Da Vinci 32 - 20133.

In the sequel we investigate, through a Parikh mapping, the relation of commutative to non-commutative grammars, and non commutative matrix grammars [7, 8].

2. Definitions and first properties.

Let V a finite vocabulary. A *bag* (or *multiset*) is a finite set, possibly with repetition, of elements in V, (i. e. a bag is an unordered string) [6].

Let λ denote both the nul string and nul bag.

Let ${}^*V$ (resp. V^*) denote the set of all bags (resp. strings) made of elements in V.

Let ${}^+V$ (resp. V^+) denote the sets ${}^*V - \{\lambda\}$ (resp. $V^* - \{\lambda\}$). Notation: bags (strings) are written as lowercase (greek) letters at the end of the alphabet.

Operations on bags.

Let

$$V = \{a_1, \ldots, a_k\}.$$

Let

$$p = a_1^{p_1} \ldots a_k^{p_k} \qquad q = a_1^{q_1} \ldots a_k^{q_k},$$

$$r = a_1^{r_1} \ldots a_k^{r_k} \text{ be in } {}^*V.$$

We define the following operations and relation, where $i = 1, \ldots, k$.

Union:

$$r = p \cup q = a_1^{r_1} \ldots a_k^{r_k}, \text{ where } r_i = \max(p_i, q_i).$$

Sum or concatenation:

$$r = p \cdot q = pq = a_1^{r_1} \ldots a_k^{r_k}, \text{ where } r_i = p_i + p_i.$$

Intersection:

$$r = p \cap q = a_1^{r_1} \ldots a_k^{r_k}, \text{ where } r_i = \min(p_i, q_i).$$

Inclusion:

$$p \subseteq q \quad \text{iff} \quad p_i \leq q_i.$$

COMMUTATIVE GRAMMARS.

We now introduce the new concepts of commutative grammar and language.

A *Commutative grammar* GC is a triple (V_N, V_T, PC) where

$V_N = \{A, B, ...\}$ is a finite non empty nonterminal alphabet,

$V_T = \{a, b, ...\}$ is a finite non empty, terminal alphabet, $V_N \cap V_T = \varnothing$,

PC is a finite, non empty, set of rules of the form $u \longrightarrow v$, u in ^+V_N, v in $^*V$, where $V = V_T \cup V_N$.

DERIVATIONS. Let q in $^*V$, p in ^+V.

Then q *immediately derives* from q (written $p \underset{GC}{\Longrightarrow} q$) iff there exist bags u, v, x such that $p = ux$, $q = vx$ and $u \longrightarrow v$ in PC.

Also q *derives from* p (written $p \overset{*}{\underset{GC}{\Longrightarrow}} q$) iff for some $n \geq 0$ there exist $n+1$ bags $r_0 \dots r_n$ such that $r_0 = p$, $r_n = q$, $r_i \underset{GC}{\Longrightarrow} r_{i+1}$, $i = 0, \dots, n-1$.

The *length* of the derivation is n. The name GC of the grammar will be omitted when obvious.

Let GC a commutative grammar and s a bag in ^+V. The *language generated by the pair* (GC, s) is

$$L(GC, s) = \{t \mid t \text{ in } {}^*V_T \text{ and } s \overset{*}{\underset{GC}{\Longrightarrow}} t\}.$$

Notice that, in contrast with the classical definition, a grammar is here defined as a triple with the axiom excluded. The same convention will be applied also to non-commutative grammars.

CLASSES OF GRAMMARS.

The same restrictions on the rules characterize the following classes of noncommutative and commutative grammars.

CONTEXT-SENSITIVE: all rules are of the form $u \longrightarrow v$, where $|u| \leq |v|$ [2].

CONTEXT-FREE: all rules are of the form $A \longrightarrow u$, where A in V_N.

REGULAR: all rules are of the form $A \longrightarrow aB$ or $A \longrightarrow a$, where A, B in V_N, and a in V_T. It is immediate that the three above types of commutative grammars generate recursive languages.

---

[2] $|u|$ denotes the length, i. e. number of symbols, in the bag or string u.

EQUIVALENCE. Let GC and GC' be commutative grammars, and s, s' bags respectively in $+V$ and $+V'$. Then the *pairs* (GC, s) and (GC', s') are *equivalent* if $L(GC, s) = L(GC', s')$.

THRESHOLD-FREE GRAMMARS. A commutative grammar is *threshold-free* iff, for any production $u \longrightarrow v$ in PC, $u \cap v = \lambda$.

It is immediately seen that any commutative grammar can be transformed into an equivalent threshold-free grammar as stated in the next lemma.

LEMMA 2.1. Let $GC = (V_N, V_T, PC)$.

Then a threshold-free grammar $GC' = (V_N', V_T, PC')$ can be constructed such that the pairs (GC, s), (GC', s) are equivalent for all s in $+V_N$.

The following definitions are needed in order to compare commutative and non-commutative grammars.

PARIKH MAPPING. Let $V = \{a_1, \ldots, a_n\}$, N the set of natural numbers. Define the mapping

$$\Psi: V^* \cup {}^*V \longrightarrow N^n$$

as follows.

For μ in V^*:

$$\Psi(\mu) = (\#\, a_1(\mu), \ldots, \#\, a_n(\mu))\ (^3).$$

For p in ${}^*V$, where $p = a_1^{i_1} \ldots a_n^{i_n}$

$$\Psi(p) = (i_1, \ldots, i_n).$$

Notice that the restriction of Ψ to bags is a one-to-one mapping.

The mapping is extended in the obvious way to languages. The following notion is needed in order to relate commutative and non-commutative grammars.

Ψ-EQUIVALENCE. Let GN, GC noncommutative and commutative grammars, and let σ, s, respectively a nonnul string and bag.

Then the pairs (GN, σ) and (GC, s) are *Ψ-equivalent* iff

$$\Psi(L(GN, \sigma)) = \Psi(L(GC, s)).$$

---

(3) $\#\, a_1(\mu)$ denotes the number of occurrences of a_1 in μ.

A remark is now in order. Let $GN = (V_N, V_T, P)$, $GC = (V_N, V_T, PC)$ where for each $\mu \longrightarrow \nu$ in P there exists a $u \longrightarrow v$ in PC, with $\Psi(u) = \Psi(\mu)$, $\Psi(v) = \Psi(\nu)$.

Let also $\Psi(s) = \Psi(\sigma)$. Then it is not true in general that $\Psi(L(GC, s)) = \Psi(L(GN, \sigma))$, as shown by the example.

Example

$$GC: \quad \begin{matrix} A \longrightarrow B\,a\,C \\ BC \longrightarrow b \end{matrix}, \quad s = A$$

$$GN: \quad \begin{matrix} A \longrightarrow BaC \\ BC \longrightarrow b \end{matrix}, \quad \sigma = A.$$

Then

$$\Psi(L(GC, s)) = \Psi(\{ab\}) = (1, 1),$$

whereas

$$\Psi(L(GN, \sigma)) = \Psi(\varnothing) = \varnothing.$$

On the other hand, if GN and GC are context-free then, for every s, such that $\Psi(s) = \Psi(\sigma)$, whe have $\Psi(L(GC, s)) = \Psi(L(GN, \sigma))$ since for every derivation $s \overset{*}{\underset{GC}{\Longrightarrow}} t$ we can find a derivation $\sigma \overset{*}{\underset{GN}{\Longrightarrow}} \tau$, and conversely, such that the rules applied at each step of both derivations are respectively $\mu \longrightarrow \nu$, $u \longrightarrow v$ with

$$\Psi(u) = \Psi(\mu), \quad \Psi(v) = \Psi(\nu).$$

It follows that $\Psi(t) = \Psi(\tau)$.

CLOSURE PROPERTIES.

It is immediate to verify that commutative languages are closed with respect to the following classical operations:

a) Union

b) Intersection

c) Sum or Concatenation

d) Homomorphism and reverse homomorphism

e) Substitution.

COMMUTATIVE GRAMMARS AND EQUIVALENT MODELS FOR PARALLEL PROCESSING.

In studies of parallel processing two models, vector addition systems [1] and Petri nets [2] have been proposed which are known to be equivalent

to each other [3, 5]. Next we recall the aforementioned models and we show their equivalence to commutative grammars.

A *Petri net* (fig. 1) is an oriented graph where nodes are partitioned in two types called *places* and *transitions*, respectively represented by circles and short segments. An arc can interconnect two nodes of different types. Each place can contain any number of *token*, represented as dots. The *state* of the net (fig. 1) is the number of tokens in each place and can be represented as a bag (e. g. BCD^2).

A transition t_i is *firable* if, for each arc terminating on t_i and originating from a place A_j, A_j contains at least one token.

In the example t_1 is not firable, whereas t_2 and t_3 are firable.

The *firing* of a firable transition has the effect of subtracting one token per arc from each ingoing place and adding one token per arc to each out-going place. Thus the firing of t_3 produces the state ABC, and t_1 becomes firable.

Let N a Petri net. It is immediate to construct a commutative grammar GC such that, for any initial state s of the net, the set of reachable states in N equals $L(GC, s)$, and conversely.

For the net of fig. 1, the following grammar is equivalent, up to an isomorphism.

$$A \to BC \qquad B \to b$$

$$BC \to AD \qquad C \to c$$

$$CD^2 \to AC \qquad D \to d.$$

$$A \to a$$

An n-dimensional *vector addition system* is a pair (r, W) where r (starting state) is a n-dimensional vector of nonnegative integers and W is a finite set of n-dimensional integer vectors, i. e. a matrix. The set $R(r, W)$ of reachable states is the set of all vectors of the form

$$r + c_1 + c_2 + \ldots + c_q = q,$$

where c_i in W, $i = 1, 2, \ldots, q$ and

$$r + c_1 + c_2 + \ldots + c_i \geq 0, \qquad i = 1, 2, \ldots, q.$$

From the preceding definitions the equivalence of threshold-free commutative grammars and vector addition systems follows immediately. From Lemma 2.1 the equivalence extends to general commutative grammars.

3. Decision problems for commutative grammars.

In this section we recast, in terms of commutative grammars, some typical problems of Petri nets.

Let $GC = (V_N, V_T, PC)$.

REACHABILITY: [4, 5]. Does it exist an algorithm which, for any pair (GC, s) and t in V_T, determines whether $s \overset{*}{\underset{GC}{\Longrightarrow}} t$?

LIVENESS: [9, 4, 5]. A rule $u \rightarrow v$ in PC is *live* with respect to a bag s in ^+V_N, if for any t in $^*V_N$ such that $s \overset{*}{\Longrightarrow} t$, there exists q in $^*V_N$ such that $t \overset{*}{\Longrightarrow} q$ and $q \supseteq u$.

Does it exist an algorithm which, for any pair (GC, s) and for any rule in PC, determines whether the rule is live with respect to s?

BOUNDEDNESS: A *pair* (GC, s) is *nonterminal bounded* (ntb) if there exists a positive integer k, which depends only on (GC, s), such that for any t such that $s \overset{*}{\Longrightarrow} t$, the number of nonterminals in t does not exceed k.

If GC is such that each terminal occurs exactly in one rule $A \rightarrow a$, (as in the example of fig. 1), the above definition coincides with the definition of

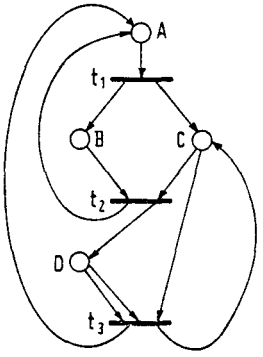

Fig. 1

boundedness found in [5]. A *grammar* GC is *nonterminal bounded* if any pair (GC, s) is ntb.

Does it exist an algorithm which, for any pair (GC, s) (resp. for any grammar GC) determines whether (GC, s) (resp. GC) is ntb?

SAFETY [9]. A pair (GC, s) is *m safe* if for any A in V_N and for any t such that $s \overset{*}{\Longrightarrow} t$, $t \not\supseteq A^{m+1}$.

Another property which deserves attention is semilinearity, because of its connection with the periodicity of nets behaviour. Semilinearity has been investigated in [3].

LINEAR AND SEMILINEAR SETS [10, 11].

Let $A \subseteq N^n = \{n\text{-tuples of positive integers}\}$.

The *set* A is *linear* iff there exist k in N and $k+1$ elements in N^n, denoted by $C, P_1, P_2, \ldots, P_k$, such that

$$A = \left\{ x \mid x = C + \sum_1^k h_i p_i,\ h_i \text{ in } N \right\}.$$

Then C is the constant and P_i, $i = 1, \ldots, k$ are the *periods* of the linear set.

A set $B \subseteq N^n$ is *semilinear* iff B is the finite union of linear sets.

SEMILINEAR LANGUAGES.

A commutative language $L \subseteq {}^*V$, $V = \{a_1, a_2, \ldots, a_n\}$, is *semilinear* (*linear*) if $\Psi(L) \subseteq N^n$ is a semilinear (linear) set.

It is interesting to investigate under which conditions $L(GC, s)$ is semilinear.

Next we briefly comment on the the results concerning the above problems before presenting new formal developments.

Reachability and liveness problems are presently not solved for a general commutative grammar. Recently Hack [5] has shown that the two problems are recursively equivalent. Proper subclasses of Petri nets have been introduced for which one or both problems are decidable. These subclasses include marked graphs [9] conflict-free nets [3] [17], free-choice nets [4], each class being included by the following one. In the next section we prove the decidability of a new subclass.

4. Modular commutative grammars.

The starting point for the considerations of this section is nonterminal-boundedness.

The decidability of n. t. b., for a pair (GC, s) and therefore of m-safety, for every m, is a straightforward corollary of the *coverability theorem* by Karp and Miller [1] (in the formalism of vector addition systems): from the algorithm of [1] it is also easy to construct a pair (GC, s), GC regular,

which is equivalent to a n. t. b. pair (GC, s) [4]: thus the semilinearity of n. t. b. pairs immediately follows from Parikh theorem [10, 11].

Now we can define an operation of composition of commutative grammar pairs, each pair semilinear, which guarantees the semilinearity of the compound grammar pair, without implying nonterminal boundedness for the latter. Our construction is based on the following results due to Greibach [13, 14].

Let $L \subseteq {}^*V$, and, for each a in V, let $L_a \subseteq {}^*V_a$, where V and V_a are alphabets.

Define a substitution τ as follows:

$$\tau(a) = L_a \text{ for each } a \text{ in } V;$$

$$\tau(x \cdot y) = \tau(x) \cdot \tau(y) \text{ for } x, y \text{ in } {}^*V;$$

$$\tau(L) = \{r \mid r \text{ in } \tau(w), \ w \text{ in } L\}.$$

A *nested iterated substitution* is defined as follows: Let τ a substitution on V such that a in $\tau(a)$ for all a in V, extend the definition to $V \cup (\bigcup_a V_a)$ by defining $\tau(b) = \{b\}$ for b in $(\bigcup_a V_a) - V$. Then:

$$\tau^1(L) = \tau(L).$$

$$\tau^{n+1}(L) = \tau(\tau^n(L)), \quad \text{for} \quad n \geq 1$$

$$\tau^\infty(L) = \bigcup_n \tau^n(L).$$

It was proved by Greibach [17, 14] for the non-commutative languages that:

THEOREM 4.1. Let L and L_a semilinear languages. Then $\tau^\infty(L)$ is semilinear, i. e. the family of semilinear languages is closed under nested iterated substitution.

The result obviously applies also to commutative languages. Next we define a new class of commutative grammars.

A *modular commutative grammar* $GC = (V_N, V_T, P)$ is the union of a finite set of components $G_1, \ldots, G_m, \ldots, G_n$ satisfying conditions 1), 2), 3) and 4):

1) $G_i = (V_i, V_{T_i}, P_i), \quad i = 1, \ldots, m, \quad 0 \leq m \leq n$

is a context-free commutative grammar.

---

[4] In [3] it is reported a complete proof of this statement and it is also proved the decidability of n. t. b. for a grammar GC.

2) $G_j = (V_j, V_{T_j}, P_j), \quad j = m+1, \ldots, n$

is a utb grammar.

Thus

$$V_T = \bigcup_{i=1 \ldots n} V_{T_i}, \quad V_N = \bigcup_{i=1 \ldots n} V_i, \quad P = \bigcup_{i=1 \ldots n} P_i .$$

For h, k in $1 \ldots n$ and $h \neq k$ let $D_{hk} = \{A \text{ in } V_N \mid u \to Av \text{ in } P_h \text{ and } Aw \to y \text{ in } P_k\}$ hence

$$V_h \cap V_k = D_{hk} \cup D_{kh} .$$

3) If A is in D_{hj}, then A never occurs in a production $Aw \to y$, $w \neq \lambda$, of P.

4) For any grammar G_j, if $A \xrightarrow[G_j]{*} Bx$ where A in D_{hj}, B in D_{jk}, x in ${}^*(V_j \cup V_T)$, then no nonterminal in x may occur in a production $Aw \to y$, $w \neq \lambda$ of P.

Notice that condition 4) is decidable because $L(G_j, A)$ is semilinear.

THEOREM 4.2. A *modular pair* (GC, s), s in ${}^+(\bigcup_{i,j} D_{ij})$, generates a semilinear language.

PROOF: We demonstrate the existence of a substitution h such that $L(GC, s) = h^\infty(s)$.

Define, for all B in V_N,

$$h(B) = \{x \mid B \xrightarrow[G_i]{*} x, \ i = 1 \ldots n\}$$

and for b in V_T

$$h(b) = b.$$

Extend h in the obvious way to bags over ${}^*V$. Consider a derivative:

$$s = x_0 \xrightarrow[G_{i_1}]{*} x_1 \xrightarrow[G_{i_2}]{} x_2 \ldots \xrightarrow[G_{i_p}]{} x_p = x.$$

Then:

$$x_1 \text{ in } h^1(x_0), \quad x_2 \text{ in } h^1(x_1) = h^2(x_0), \ldots,$$

$$x_p \text{ in } h^p(x_0), \quad \text{i. e. } x \text{ in } h^\infty(s).$$

Conversely, let x in $h^\infty(x)$, we show that x in $L(GC, s)$. Let x in $h^p(s)$, i. e. x in $h(h^{p-1}(s))$.

Then there exists a derivative $x_{p-1} \xrightarrow[G_{i_p}]{} x_p = x$, and by induction $s \xrightarrow[GC]{p} x_{p-1}$. Thus $h^\infty(s) = L(GC, s)$ and by Theor. 4.1 the proof is completed.

EXAMPLE. Consider the following grammar GC, which can be partitioned into three components:

$$
\begin{array}{lll}
 & A_3 \to A_1 & \text{Context free;} \\
G_1: & A_1 \to A_1A_2 & \\
 & A_2 \to A_1A_2 & \\
 & & \\
 & A_2 \to B_1B_2 & \text{ntb;} \\
G_2: & B_1B_2 \to A_3 & \\
 & B_1B_2 \to A_3C_4 & \\
 & & \\
 & A_2 \to C_1C_2 & \text{ntb;} \\
 & C_1C_2 \to A_1 & \\
G_3: & C_1C_2 \to C_1C_3 & \\
 & C_1C_3 \to A_1A_2 & \\
 & C_4 \to C_1C_3\,. &
\end{array}
$$

We have:

$$D_{12} = \{A_2\},\ D_{13} = \{A_2\},\ D_{21} = \{A_3\},\ D_{31} = \{A_1, A_2\},\ D_{23} = \{C_4\},\ D_{32} = \varnothing.$$

The corresponding Petri net is represented in Fig. 2. Notice that the boundaries of the three components in Fig. 2 are arbirary to some extent: e. g. $A_3 \to A_1$ could be moved from G_1 to G_2. Let $s = A_1A_2$ the axiom. Then the conditions of Theor. 4.2 are met and $L(GC, s)$ is semilinear.

We note that conditions 1) to 4) of our definition provide a sufficient but restrictive condition for the applicability of Greibach result to com-

mutative languages. Therefore it would be expedient to relax some of the conditions. However, it seems that modular Petri nets as they stand, have a meaningful interpretation in terms of models for parallel processing. A

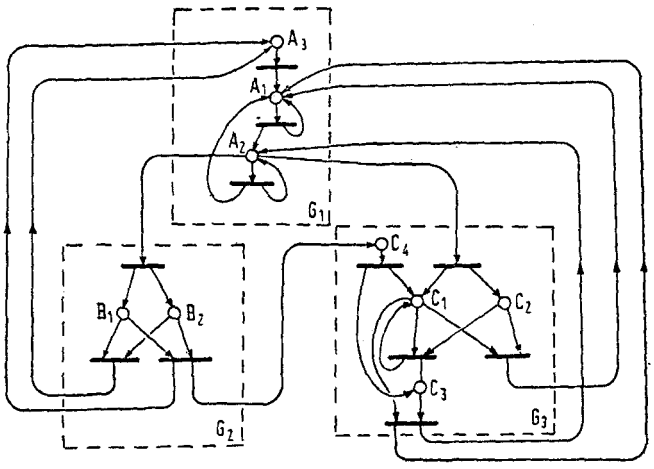

Fig. 2

ntb-component could represent a synchronization process which must be completed prior to the reactivation of the same process. On the other hand, processes which do not require synchronization, i. e. context free components, are not subject to the previous constraint.

5. Commutative and matrix grammars.

In this section we prove the equivalence of commutative and matrix [7, 8] grammars (5).

A *matrix grammar* GM of *order* n is a triple (V_N, V_T, P) where

$V_N =$ nonterminal alphabet

$V_T =$ terminal alphabet

$V = V_T \cup V_N$

$P = \{p_i\}$. Each production p_i is a k_i-tuple of context-free components:

$$[A_{i,1} \to \mu_{i,1}, A_{i,2} \to \mu_{i,2}, \dots, A_{i,k_i} \to \mu_{i,k_i}]$$

(5) The same result that appeared in [3] was also independently obtained by [18].

where $1 \leq k_i \leq n$, $A_{i,j}$ in V_N, $\mu_{i,j}$ in V^* for $i = 1, 2, \ldots, k_i$. Let (i, j) the j-th component of p_i.

Let GN the (non-commutative) context-free grammar having as production set the set of all the components (i, j) of MG. Let ϱ, τ in V^+. Then ϱ *immediately derives* τ, written $\varrho \underset{MG}{\Longrightarrow} \tau$, iff there exists a derivation in GN

$$\varrho = \varrho_0 \overset{(i,1)}{\underset{GN}{\Longrightarrow}} \varrho_1 \overset{(i,2)}{\underset{GN}{\Longrightarrow}} \cdots \overset{(i,k_i)}{\underset{GN}{\Longrightarrow}} \varrho_{k_i} = \tau.$$

Then a derivation is defined as usually.

The *language* generated by a matrix grammar pair (GM, σ), σ in V^+, is then:

$$L(MG, \sigma) = \{\tau \mid \tau \text{ in } V_T^* \text{ and } \sigma \underset{MG}{\overset{*}{\Longrightarrow}} \tau\}.$$

The next theorem establishes the Ψ-equivalence of commutative and matrix grammars.

THEOREM 5.1. For any commutative grammar pair (GC, s), $GC = (V_N, V_T, PC)$, there exists a Ψ-equivalent matrix grammar pair (MG, σ), $MG = (V_N, V_T, P)$, and conversely.

PROOF: First, if GC is not threshold-free, replace it by an equivalent threshold-free grammar (v. s. Lemma 2.1).

Let $PC = \{u \to v, \ldots\}$ where we assume without loss of generality

$$u = A_1 A_2 \ldots A_p, \quad A_i \text{ in } V_N, \quad i = 1, 2, \ldots, p$$

and

$$v = B_1 B_2 \ldots B_q, \quad B_i \text{ in } V_N, \quad i = 1, 2, \ldots, q$$

or

$$u = A, \quad A \text{ in } V_N$$

and

$$v = a, \quad a \text{ in } V_T.$$

Let

$$n = \max \{p \mid (u \to v) \text{ in } PC\}.$$

Construct a matrix grammar GM of order n as follows. For each rule $A_1 \ldots A_p \to B_1 \ldots B_q$ construct the rule. $[A_1 \to B_1, \ldots, A_p \to B_p \; B_{p+1} \ldots B_q]$ if $q \geq p$, or otherwise

$$[A_1 \to B_1, \ldots, A_q \to B_q, \; A_{q+1} \to \lambda, \ldots, A_p \to \lambda].$$

Similarly, for each rule $A \longrightarrow a$, construct the rule $[A \longrightarrow a]$. It is easy to see that

$$u \underset{GC}{\Longrightarrow} v$$

iff there exist two strings ϱ, τ such that

$$\varrho \underset{GM}{\Longrightarrow} \tau$$

where $\Psi(\varrho) = \Psi(u)$, $\Psi(\tau) = (v)$.

It follows that for any string σ, such that $\Psi(\sigma) = \Psi(s)$,

$$\Psi(L(GM, \sigma)) = \Psi(L(GC, s)).$$

Conversely, consider, without loss of generality, GM of order n with productions of the two types:

1) $[A_{i,1} \longrightarrow \mu_{i,1}\ ,\ A_{i,2} \longrightarrow \mu_{i,2}\ , \dots\ , A_{i,k_i} \longrightarrow \mu_{i,k_i}]$

where

$$\mu_{i,h} \text{ in } V_N^*,\ h = 1, \dots, k_i,\ 1 \leq k_i \leq n.$$

2) $[A \longrightarrow a]$

where a in V_T.

Define the integer vector Δ_i with $|\ V_N\ |$ components:

$$\Delta_{i,j} = \Psi_j(\mu_{i,1}\ \mu_{i,2} \dots \mu_{i,k_i}) - \Psi_j(A_{i,1}\ A_{i,2} \dots A_{i,k_i})$$

$j = 1, \dots, |\ V_N\ |$. Construct a second integer vector U_i with nonnegative components:

$$U_{i,j} = \max_{h=1 \dots k_i} \{0, (\Psi_j(A_{i,1}\ A_{i,2} \dots A_{i,h}) - \Psi_j(u_{i,0}\ u_{i,1} \dots u_{i,h-1}))\}$$

$j = 1, \dots, |\ V_N\ |$, and we assume $\mu_{i,0} = \lambda$, $\Psi_j(\mu_{i,0}) = 0$.

The production set of GC is constructed as follows. For each rule of type 1) construct the rules

$$\Psi^{-1}(U_i) \longrightarrow \Psi^{-1}(\Delta_i + U_i),$$

and for each rule of type 2) construct the rule $A \longrightarrow a$. It should be obvious that, for any s such that $\Psi(\sigma) = \Psi(s)$ the pairs (GM, σ) (GC, s) are Ψ-equivalent.

Q. E. D.

EXAMPLE: the following grammar pairs are Ψ-equivalent:

MG [15]: $[A_1 \to A_2A_3A_4A_5]$

$[A_2 \to AA_2,\ A_3 \to A_3A,\ A_4 \to A_4A,\ A_5 \to AA_5]$

$[A_2 \to BA_2,\ A_3 \to A_3B,\ A_4 \to A_4B,\ A_5 \to BA_5]$

$[A_2 \to \lambda,\ A_3 \to \lambda,\ A_4 \to \lambda,\ A_5 \to \lambda]$

$[A \to a]$

$[B \to b]$

$\sigma = A_1$

GC: $A_1 \to A_2A_3A_4A_5$

$A_2A_3A_4A_5 \to A^4A_2A_3A_4A_5$

$A_2A_3A_4A_5 \to B^4A_2A_3A_4A_5$

$A_2A_3A_4A_5 \to \lambda$

$A \to a$

$B \to b$

$s = A_1$

Unfortunately the generative capacity of matrix grammars, with the above definition of derivation, is not known [8]. In particular, it is an open problem whether matrix languages are recursive. As a consequence, the Ψ-equivalence of matrix and commutative grammars is of no use, in the general case and at the present state of knowledge, for solving the reachability problem (6).

Therefore we can only deduce the equivalence between recursiveness-emptiness problem for matrix languages and reachability for commutative

---

(6) In the literature certain recursive subclasses of matrix languages (simple matrix, equal matrix) [15; 16] have been defined. However their corresponding commutative grammars seem uninteresting.

languages: in fact recurviveness and emptiness problems are equivalent for matrix languages, since λ-right hand sides are allowed and the class of matrix languages is closed under intersection with regular sets [8]; thus theorem 5.1 clearly implies that

i) the solution of recursiveness problem for m. l. implies the solution of reachability problem for c. l.

ii) the solution of reachability problem for c. l. implies the solution of emptiness problem for m. l..

7. Conclusion.

In our opinion the formalism of commutative grammars provides a concise representation of parallel processing and synchronization problems.

In addition the formalism allows the transfer of well-known results of noncommutative formal languages (e. g. Parikh theorem) to the commutative ones.

However the main problems of Petri nets, i. e. decidability of reachability and liveness remain open in the general case. Also for the semilinearity problem, we have only been able to prove the property for some subclasses.

Notice that the semilinearity property was proved not to hold in the general case [5]: thus the problem is to find necessary and/or sufficient conditions for semilinearity.

It is easy to show that the non-semilinearity of a general commutative language implies also the non semilinearity of a context-sensitive commutative language, which is recursive: it follows that, as in the non commutative case, the domain of semilinearity is properly included in the domain of recursiveness.

REFERENCES

[1] KARP M. R., MILLER R. E., *Parallel program schemata,* J. Comp. Syst. Sc. **3** (1969), 147-195.

[2] PETRI C. A., *Kommunikation mit automaten,* Schriften des Rheinish, Westfalischen Inst. fur Instrumentelle Mathematik (1962), Hft. 2, Bonn.

[3] CRESPI - REGHIZZI S., MANDRIOLI D., *Petri nets and commutative grammars,* Rapp. N. 74-5, Istituto Elettronica del Politecnico di Milano, March 1974.

[4] HACK M., *Analysis of Production schemata by Petri nets,* Rept. MAC TR-94, MIT, Feb. 1972.

[5] HACK M., *Decision Problems for Petri nets and vector addition systems,* Comp. Structures Group Memo 95-2, MIT, Aug. 1974.

[6] CERF V. G., et al., *Proper termination of flow of Control in programs involving concurrent processes,* Proc. ACM Annual Conf., Boston, Aug. 1972, Vol. II, 742-754.

[7] ABRAHAM, S., *Some questions of phrase structure grammars I,* Computational Linguistics (1965), 61-70.

[8] SALOMAA A., Formal languages (1973), Academic Press, New York.

[9] HOLT A. W., COMMONER F., *Events and conditions,* Project MAC Conf. Concurrent Systems and parallel computation, Woods Hole, Mass., 1970.

[10] PARIKH R. J., *On context-free languages,* J. ACM, 13 (1966), 570-581.

[11] GINSBURG S., The mathematical theory of context-free languages (1966), Mc Graw-Hill., New York.

[12] GINSBURG S., SPANIER, E. H., *Derivation bounded languages,* J. Comp. Syst. Sc. **2** (1968), 228-250.

[13] GREIBACH S. A., *A generalization of Parikh's semilinear theorem,* Discrete Math. **2** (1972), 347-355.

[14] GREIBACH S., *Full AFLs and nested iterated substitution,* Inform. and Contr. **16** (1970), 7-35.

[15] IBARRA, O. H., *Simple matrix languages,* Inform. and Contr. **17** (1970), 359-394.

[16] SIROMONEY R., *On equal matrix languages,* Inform. and Contr. **14** (1969), 135-151.

[17] CRESPI - REGHIZZI S., MANDRIOLI D. *A decidability theorem for a class of vector addition systems,* Info. Proc. Letters **3** (1975), 78-80.

[18] VAN LEEUWEN J., *A Partial Solution to the Reachability Problem for Vector Addition Systems,* 6 th Annual ACM Symposium on Theory of Computing, May 1974, 303-309.