Acta Cybernetica **13** (1997) 141-157.

CD Grammar Systems and Trajectories*

Alexandru Mateescu†

Abstract

In this paper we consider constraints, as a new research area for cooperating distributed (CD) grammar systems. Constraints are based on the notion of a trajectory. The flexible approach provides a framework to study some interesting properties of a CD grammar system like fairness or parallelization of languages. The use of teams in the derivations of words is also considered.

1 Introduction

The cooperating distributed (CD) grammar systems were introduced in [2] with motivations from Artificial Intelligence (the so-called blackboard model in problem solving, [22]). For more details see the monograph [3].

Such a system consists of several ordinary grammars working by turns on the same sentential form; at each moment, one component is active, the others are waiting. Natural variants are systems in which more components (a team) are active at the same time. Teams can be with prescribed number of elements, nondeterministically chosen. The notion was introduced in [8].

In this paper we consider constraints that involve the general strategy to switch from one component (team) to another component (team).

Usually, the operation is modelled by the shuffle operation or restrictions of this operation, such as literal shuffle, insertion, etc.

Syntactic constraints, we consider here, are based on the notion of a *trajectory*, introduced in [16]. Roughly speaking, a trajectory is a segment of a line in the plane, starting in the origin of axes and continuing parallel with the axis Ox or Oy. The line can change its direction only in points of non-negative integer coordinates.

A trajectory defines how to skip from a component (team) to another component (team) during the derivation operation.

Languages consisting of trajectories are a special case of picture languages introduced in [20].

*The work reported here has been supported by the Academy of Finland, Project 137358. Presented at the workshop *Grammar Systems: Recent Results and Perspectives, July 26-27, 1996, Budapest.*

†Department of Mathematics, University of Bucharest, Romania and Department of Mathematics, University of Turku, Finland

2 Basic definitions

The reader is referred to [25] for basic elements of formal language theory and to [3] for details about grammar systems.

For an alphabet Σ, we denote by Σ^* the free monoid generated by Σ under the operation of concatenation; λ is the empty string and $|x|$ is the length of $x \in \Sigma^*$. If $a \in \Sigma$ and $w \in \Sigma^*$, then $|w|_a$ denotes the number of occurrences of the symbol a in w.

The *anti-catenation* operation, denoted by "$\circ$", is defined as: $u^\circ v = vu$, for any $u, v \in \Sigma^*$.

A *generalized sequential machine (GSM)* is a 6-tuple $M = (Q, \Sigma, \Delta, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is the *input alphabet*, Δ is the *output alphabet*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *final states*, and δ is the *transition function*, i.e., a function from $Q \times \Sigma$ to finite subsets of $Q \times \Delta^*$. Let $u = u_1 u_2 \ldots u_n$ be a word from Σ^*, where $u_i \in \Sigma$, $1 \leq i \leq n$. The set of all output words of u by M, denoted $M(u)$, is:

$$M(u) = \{w \mid w = w_1 w_2 \ldots w_n, \text{ where } w_i \in \delta(q_{i-1}, u_i), 1 \leq i \leq n,$$

$$\text{and } \delta(q_{n-1}, u_n) \in F\}.$$

If $L \subseteq \Sigma^*$ is a language, then:

$$M(L) = \bigcup_{u \in L} M(u).$$

For more informations about GSM, the reader is referred to [24].

A CD grammar system (of degree $n, n \geq 1$) is a construct

$$\Gamma = (N, \Sigma, P_1, P_2, \ldots, P_n, S),$$

where N is a (nonterminal) alphabet, Σ is a (terminal) alphabet disjoint from N, $S \in N$ and P_i are finite sets of context-free rules over $N \cup \Sigma, 1 \leq i \leq n$.

For a given P_i, the direct derivation $\Longrightarrow_{P_i}$ is defined in the usual way; we denote by $\Longrightarrow_{P_i}^{=k}, \Longrightarrow_{P_i}^{\leq k}, \Longrightarrow_{P_i}^{\geq k}, \Longrightarrow_{P_i}^{*}, \Longrightarrow_{P_i}^{t}$ a derivation in P_i consisting of exactly k steps, at most k, at least k steps, $k \geq 1$, of any number of steps and as long as possible, respectively ($x \Longrightarrow_{P_i}^{t} y$ means that $x \Longrightarrow_{P_i}^{*} y$ and there is no z such that $y \Longrightarrow_{P_i} z$).

For $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ we denote by $L_f(\Gamma)$ the language generated by Γ in the f mode, that is

$$L_f(\Gamma) = \{x \in \Sigma^* \mid S \Longrightarrow_{P_{i_1}}^{f} x_1 \Longrightarrow_{P_{i_2}}^{f} \cdots \Longrightarrow_{P_{i_m}}^{f} = x$$
$$1 \leq i_j \leq n, 1 \leq j \leq m, m \geq 1\}$$

and by $CD(f)$ the family of such languages. (Note that we do not distinguish here between systems with λ-free and with arbitrary components.)

Given a grammar system $\Gamma = (N, \Sigma, P_1, \ldots, P_n, w)$ with components $N, \Sigma, P_1, \ldots, P_n$ as above but with a string axiom $w \in (N \cup \Sigma)^*$ instead of a symbol $S \in N$, and given a natural number $s \geq 1$, a subset $Q = \{P_{i_1}, \ldots, P_{i_s}\}$ of $\{P_1, \ldots, P_n\}$ is called an s-team. For such an s-team Q and for $x, y \in (N \cup \Sigma)^*$, we write

$$x \Longrightarrow_Q y \quad \text{iff} \quad x = x_1 A_1 x_2 \ldots A_s x_{s+1}, y = x_1 y_1 x_2 \ldots y_s x_{s+1},$$
$$x_j \in (N \cup \Sigma)^*, 1 \leq j \leq s+1, A_j \to y_j \in P_{i_j}, 1 \leq j \leq s.$$

Then the relations $\Longrightarrow_Q^{=k}, \Longrightarrow_Q^{\leq k}, \Longrightarrow_Q^{\geq k}, \Longrightarrow_Q^{*}, \Longrightarrow_Q^{t}, k \geq 1$, can be defined as above, with the clarification that in the t case the derivation is correct when no more rules of any of the team components can be used.

In the sequel we recall some operations from formal languages that simulate the parallel composition of words.

The *shuffle* operation, denoted by $\sqcup\!\sqcup$, is defined recursively by:

$$au \sqcup\!\sqcup bv = a(u \sqcup\!\sqcup bv) \cup b(au \sqcup\!\sqcup v),$$

and

$$u \sqcup\!\sqcup \lambda = \lambda \sqcup\!\sqcup u = \{u\},$$

where $u, v \in \Sigma^*$ and $a, b \in \Sigma$.
The *shuffle* of two languages L_1 and L_2 is:

$$L_1 \sqcup\!\sqcup L_2 = \bigcup_{u \in L_1, v \in L_2} u \sqcup\!\sqcup v.$$

The *literal shuffle*, denoted by $\sqcup\!\sqcup_l$, is defined as follows:

$$a_1 a_2 \ldots a_n \sqcup\!\sqcup_l b_1 b_2 \ldots b_m = \begin{cases} a_1 b_1 a_2 b_2 \ldots a_n b_n b_{n+1} \ldots b_m, & \text{if } n \leq m, \\ a_1 b_1 a_2 b_2 \ldots a_m b_m a_{m+1} \ldots a_n, & \text{if } m < n, \end{cases}$$

where $a_i, b_j \in \Sigma$.

$$(u \sqcup\!\sqcup_l \lambda) = (\lambda \sqcup\!\sqcup_l u) = \{u\},$$

where $u \in \Sigma^*$.

The *balanced literal shuffle*, denoted by $\sqcup\!\sqcup_{bl}$, is defined in the next way:

$$a_1 a_2 \ldots a_n \sqcup\!\sqcup_{bl} b_1 b_2 \ldots b_m = \begin{cases} a_1 b_1 a_2 b_2 \ldots a_n b_n, & \text{if } n = m, \\ \emptyset, & \text{if } n \neq m, \end{cases}$$

where $a_i, b_j \in \Sigma$.

The *insertion* operation, see [7], denoted by $\longleftarrow$, is defined as:

$$u \longleftarrow v = \{u'vu'' \mid u'u'' = u, u', u'' \in \Sigma^*\}.$$

All the above operations are extended in the usual way to operations with languages.

3 Trajectories and constraints

In this section we introduce the notion of the trajectory and that of the shuffle on trajectories, and study their basic properties which are necessary in the sequel. The shuffle of two words has a natural geometrical interpretation related to latticial points in the plane (points with nonnegative integer coordinates) and with a certain "walk" in the plane defined by each trajectory.

Definition 3.1 *Consider the alphabet $V = \{r, u\}$. We say that r and u are versors in the plane: r stands for the right direction, whereas u stands for the up direction. A trajectory is an element t, $t \in V^*$.*

□

Definition 3.2 *Let Σ be an alphabet and let t be a trajectory, $t = t_1 t_2 \ldots t_n$, where $t_i \in V, 1 \leq i \leq n$. Let α, β be two words over Σ, $\alpha = a_1 a_2 \ldots a_p, \beta = b_1 b_2 \ldots b_q$, where $a_i, b_j \in \Sigma, 1 \leq i \leq p$ and $1 \leq j \leq q$.*

The shuffle of α with β on the trajectory t, denoted $\alpha \sqcup\!\sqcup_t \beta$, is defined as follows: if $|\alpha| \neq |t|_r$ or $|\beta| \neq |t|_u$, then $\alpha \sqcup\!\sqcup_t \beta = \emptyset$, else

$\alpha \sqcup\!\sqcup_t \beta = c_1 c_2 \ldots c_{p+q}$, where, if $|t_1 t_2 \ldots t_{i-1}|_r = k_1$ and $|t_1 t_2 \ldots t_{i-1}|_u = k_2$, then

$$c_i = \begin{cases} a_{k_1+1}, & \text{if } t_i = r, \\ b_{k_2+1}, & \text{if } t_i = u. \end{cases}$$

□

If T is a set of trajectories, the *shuffle of α with β on the set T of trajectories*, denoted $\alpha \sqcup\!\sqcup_T \beta$, is:

$$\alpha \sqcup\!\sqcup_T \beta = \bigcup_{t \in T} \alpha \sqcup\!\sqcup_t \beta.$$

The above operation is extended to languages over Σ, if $L_1, L_2 \subseteq \Sigma^*$, then we define

$$L_1 \sqcup\!\sqcup_T L_2 = \bigcup_{\alpha \in L_1, \beta \in L_2} \alpha \sqcup\!\sqcup_T \beta.$$

Example 3.1 *Let α and β be the words $\alpha = a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8$, $\beta = b_1 b_2 b_3 b_4 b_5$ and assume that $t = r^3 u^2 r^3 ururu$. The shuffle of α with β on the trajectory t is:*

$$\alpha \sqcup\!\sqcup_t \beta = \{a_1 a_2 a_3 b_1 b_2 a_4 a_5 a_6 b_3 a_7 b_4 a_8 b_5\}.$$

The result has the following geometrical interpretation (see Figure 1): the trajectory t defines a line starting in the origin and continuing one unit to the right or up, depending of the definition of t. In our case, first there are three units right, then two units up, then three units right, etc. Assign α on the Ox axis and β on the Oy axis of the plane. Observe that the trajectory ends in the point with coordinates $(8,5)$ (denoted by E in Figure 1) that is exactly the upper right corner of the rectangle defined by α and β, i.e., the rectangle $OAEB$ in Figure 1. Hence, the result of the shuffle of α with β on the trajectory t is nonempty. The result can be read following the line defined by the trajectory t: that is, when being in a lattice point of the trajectory, with the trajectory going right, one should pick up the corresponding letter from α, otherwise, if the trajectory is going up, then one should add to the result the corresponding letter from β. Hence, the trajectory t defines a line in the rectangle $OAEB$, on which one has "to walk" starting from the corner O, the origin, and ending in the corner E, the exit point. In each lattice point one has to follow one of the versors r or u, according to the definition of t.

Assume now that t' is another trajectory, say:

$$t' = ur^5u^3rur^2.$$

In Figure 1, the trajectory t' is depicted by a much bolder line than the trajectory t. Observe that:

$$\alpha \sqcup\!\sqcup_{t'} \beta = \{b_1a_1a_2a_3a_4a_5b_2b_3b_4a_6b_5a_7a_8\}.$$

Consider the set of trajectories, $T = \{t, t'\}$. The shuffle of α with β on the set T of trajectories is:

$$\alpha \sqcup\!\sqcup_T \beta = \{a_1a_2a_3b_1b_2a_4a_5a_6b_3a_7b_4a_8b_5, b_1a_1a_2a_3a_4a_5b_2b_3b_4a_6b_5a_7a_8\}.$$

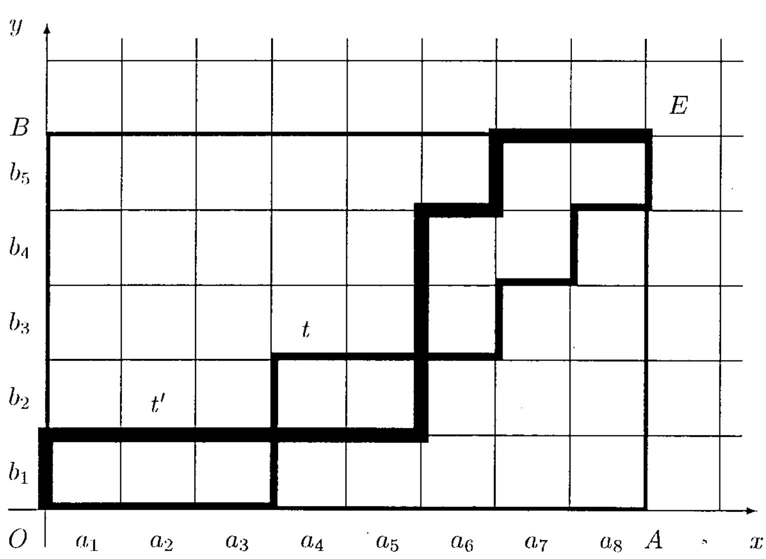

Figure 1

Remark 3.1 *One can easily observe that the following known operations for the parallel composition of words are particular cases of the operation of shuffle on trajectories.*

1. *Let T be the set $T = \{r, u\}^*$. Then for the shuffle operation $\sqcup\!\sqcup$, $\sqcup\!\sqcup_T = \sqcup\!\sqcup$.*
2. *Assume that $T = (ru)^*(r^* \cup u^*)$. Note that in this case $\sqcup\!\sqcup_T = \sqcup\!\sqcup_l$, the literal shuffle.*
3. *Consider $T = (ru)^*$. Then $\sqcup\!\sqcup_T = \sqcup\!\sqcup_{bl}$, where $\sqcup\!\sqcup_{bl}$ is the balanced literal shuffle.*
4. *Define $T = r^*u^*r^*$ and note that $\sqcup\!\sqcup_T = \longleftarrow$, where $\longleftarrow$ refers to the the insertion operation.*
5. *Assume that $T = r^*u^*$. It follows that $\sqcup\!\sqcup_T = \cdot$, where $\cdot$ is the catenation operation.*
6. *Consider $T = u^*r^*$ and observe that $\sqcup\!\sqcup_T = ^\circ$, where $^\circ$ denotes the anti-catenation operation.*

□

The following two theorems are representation results for the languages of the form $L_1 \sqcup\!\sqcup_T L_2$. We omit their rather straightforward proofs.

Theorem 3.1 *For all languages L_1 and L_2, $L_1, L_2 \subseteq \Sigma^*$, and for all sets T of trajectories, there exist a gsm M and two letter-to-letter morphisms g and h such that*

$$L_1 \sqcup\!\sqcup_T L_2 = M(h(L_1) \sqcup\!\sqcup g(L_2) \sqcup\!\sqcup T).$$

Our next theorem is a variant of Theorem 3.1.

Theorem 3.2 *For all languages L_1 and L_2, $L_1, L_2 \subseteq \Sigma^*$, and for all sets T of trajectories, there exist a morphism φ and two letter-to-letter morphisms g and h, $g : \Sigma \longrightarrow \Sigma_1^*$ and $h : \Sigma \longrightarrow \Sigma_2^*$ where Σ_1 and Σ_2 are two copies of Σ, and a regular language R, such that*

$$L_1 \sqcup\!\sqcup_T L_2 = \varphi((h(L_1) \sqcup\!\sqcup g(L_2) \sqcup\!\sqcup T) \cap R).$$

4 Constraints and CD grammar systems

Now we consider only CD grammar systems with two components. Moreover, we assume that the rules of each component have distinct labels. The case of CD grammar systems with more than two components can be easily obtained as a generalization.

Let $\Gamma = (N, \Sigma, S, P_1, P_2)$ be a CD grammar system with two components and let $T \subseteq \{r, u\}^*$ be a set of trajectories. *The constraint language generated by* Γ is the set of all words $w \in \Sigma^*$ such that w can be generated by Γ following a trajectory from T, i.e. the components P_1 and P_2 are used according to a trajectory $t \in T$. Whenever r does occur in t the component P_1 is used, otherwise, if u does occur in t, then the component P_2 is used.

Additionaly, one may consider constraint languages associated to each component. These languages are shuffled on the set T of trajectories.

Example 4.1 *Let* $\Gamma = (N, \Sigma, S, P_1, P_2)$ *be the following CD grammar system:* $N = \{S, X\}$, $\Sigma = \{a, b, c\}$,

$$P_1 = \{p_1 : S \longrightarrow aS,\ p_2 : X \longrightarrow cX,\ p_3 : X \longrightarrow \lambda\}$$

$$P_2 = \{q_1 : S \longrightarrow bS,\ q_2 : S \longrightarrow X\}.$$

The constraint language associated to the component P_1 *is* $L_1 = \{p_1^n p_2^n p_3 \mid n \geq 1\}$ *and the constraint language associated to the component* P_2 *is* $L_2 = \{q_1^n q_2 \mid n \geq 1\}$. *The set of trajectories is* $T = \{r^n u^{n+1} r^{n+1} \mid n \geq 1\}$. *The constraint language associated to the CD grammar system* Γ *is*

$$L_1 \sqcup\!\sqcup_T L_2 = \{p_1^n q_1^n q_2 p_2^n p_3 \mid n \geq 1\}.$$

One can easily verify that the language generated by the CD grammar system Γ *with the above constraints is:*

$$L(\Gamma) = \{a^n b^n c^n \mid n \geq 1\}.$$

□

Note that the language generated is non-context-free, but also the set T of trajectories is a non-context-free language. However we will see in the next section that this language can be generated also using only context-free constraints.

5 Regular and context-free trajectories

It is well known that the shuffle of two regular languages is a regular language. Moreover, given two finite automata A_1 and A_2 one can effectively find a finite automaton A such that $L(A) = L(A_1) \sqcup\!\sqcup L(A_2)$.

The following theorem provides a characterization of those sets of trajectories T for which $L_1 \sqcup\!\sqcup_T L_2$ is a regular language, whenever L_1, L_2 are regular languages.

Theorem 5.1 *Let T be a set of trajectories, $T \subseteq \{r, u\}^*$. The following assertions are equivalent:*

(i) for all regular languages L_1, L_2, $L_1 ⊔⊔_T L_2$ is a regular language.

(ii) T is a regular language.

Proof. $(i) \Rightarrow (ii)$ Assume that $L_1 = r^*$ and $L_2 = u^*$ and note that $L_1 ⊔⊔_T L_2 = T$. It follows that T is a regular language.

$(ii) \Rightarrow (i)$ Assume that T is a regular language. Consider two regular languages L_1, L_2. Without loss of generality, we may assume that L_1 and L_2 are over the same alphabet Σ. Let $A_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ be a finite deterministic automaton such that $L(A_i) = L_i$, $i = 1, 2$. Also, let $A_T = (Q_T, \{r, u\}, \delta_T, q_0^T, F_T)$ be a finite deterministic automaton such that $L(A_T) = T$.

We define a finite nondeterministic automaton $A = (Q, \Sigma, \delta, Q_0, F)$ such that $L(A) = L_1 ⊔⊔_T L_2$. Informally, A, on an input $w \in \Sigma^*$, simulates nondeterministically A_1 or A_2 and from time to time changes the simulation from A_1 to A_2 or from A_2 to A_1. Each change determines a transition in A_T as follows: a change from A_1 to A_2 is interpreted as u and a change from A_2 to A_1 is interpreted as r. The input w is accepted by A iff A_1, A_2 and A_T accept.

Formally, $Q = Q_1 \times Q_T \times Q_2$, $Q_0 = \{(q_0^1, q_0^T, q_0^2)\}$, $F = F_1 \times F_T \times F_2$. The definition of δ is:

$$\delta((q_1, d, q_2), a) = \{(\delta_1(q_1, a), \delta_T(d, r), q_2), (q_1, \delta_T(d, u), \delta_2(q_2, a))\},$$

where, $q_1 \in Q_1, d \in Q_T, q_2 \in Q_2, a \in \Sigma$.

One can easily verify that $L(A) = L_1 ⊔⊔_T L_2$ and hence $L_1 ⊔⊔_T L_2$ is a regular language.

□

Next theorem gives a similar result as Theorem 5.1, but for context-free sets of trajectories.

Theorem 5.2 *Let T be a set of trajectories, $T \subseteq \{r, u\}^*$. The following assertions are equivalent:*

(i) for all regular languages L_1, L_2, $L_1 ⊔⊔_T L_2$ is a context-free language.

(ii) T is a context-free language.

Proof. $(i) \Rightarrow (ii)$ Assume that $L_1 = r^*$ and $L_2 = u^*$ and note that $L_1 ⊔⊔_T L_2 = T$. Therefore T is a context-free language.

$(ii) \Rightarrow (i)$ Assume that T is a context-free language. Consider two regular languages L_1, L_2. Without loss of generality, we may assume that L_1 and L_2 are over the same alphabet Σ. Let $A_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$ be a finite deterministic automaton such that $L(A_i) = L_i$, $i = 1, 2$. Also, let $P_T = (Q_T, \Gamma_T, \{r, u\}, \delta_T, q_0^T, Z_T, F_T)$ be a pushdown automaton such that $L(P_T) = T$.

We define a pushdown automaton $P = (Q, \Gamma, \Sigma, \delta, Q_0, Z, F)$ such that $L(P) = L_1 ⊔⊔_T L_2$. Informally, P, behaves as the automaton A from the proof of Theorem 5.1, except that on the second component of the states, P simulates the pushdown automaton P_T. That is, on an input $w \in \Sigma^*$, P simulates nondeterministically A_1 or A_2 and from time to time changes the simulation from A_1 to A_2 or from A_2 to A_1. Each change determines a transition in P_T as follows: a change from A_1 to A_2 is interpreted as u and a change from A_2 to A_1 is interpreted as r. The input w is accepted by P iff A_1, A_2 and P_T accept.

Formally, $Q = Q_1 \times Q_T \times Q_2$, $Q_0 = \{(q_0^1, q_0^T, q_0^2)\}$, $F = F_1 \times F_T \times F_2$, $\Gamma = \Gamma_T$, $Z = Z_T$. The definition of δ is:

$$\delta((q_1, d, q_2), a, X) = \cup_{(s,\alpha)\in\delta_T(d,r,X)}((\delta_1(q_1, a), s, q_2), \alpha)\cup$$

$$\cup_{(s',\alpha')\in\delta_T(d,u,X)}((q_1, s', \delta_2(q_2, a), \alpha')\}$$

where, $q_1 \in Q_1$, $d \in Q_T$, $q_2 \in Q_2$, $a \in \Sigma$, $X \in \Gamma$, $\alpha \in \Gamma^*$.

Additionally,

$$\delta((q_1, d, q_2), \lambda, X) = \cup_{(s,\alpha)\in\delta_T(d,\lambda,X)}((q_1, s, q_2), \alpha)$$

where, $q_1 \in Q_1$, $d \in Q_T$, $q_2 \in Q_2$, $X \in \Gamma$, $\alpha \in \Gamma^*$.

One can verify that $L(P) = L_1 ⊔⊔_T L_2$ and hence $L_1 ⊔⊔_T L_2$ is a context-free language. □

Theorem 5.3 *Let T be a set of trajectories, $T \subseteq \{r, u\}^*$ such that T is a regular language.*

(i) If L_1 is a context-free language and if L_2 is a regular language, then $L_1 ⊔⊔_T L_2$ is a context-free language.

(ii) If L_1 is a regular language and if L_2 is a context-free language, then $L_1 ⊔⊔_T L_2$ is a context-free language.

Proof. The proof is similar with the proof of Theorem 5.2. For the case (i) the pushdown automaton is simulated on the first component of the states, whereas for the case (ii) the pushdown automaton is simulated on the third component of the states. □

Alternative proofs for Theorems 5.1 - 5.3 can be obtained using Theorem 3.1 or Theorem 3.2.

From Theorems 5.1 - 5.3 we obtain the following corollary:

Corollary 5.1 *Let L_1, L_2 and T, $T \subseteq \{r, u\}^*$ be three languages.*

(i) if all three languages are regular languages, then $L_1 ⊔⊔_T L_2$ is a regular language.

(ii) if two languages are regular languages and the third one is a context-free language, then $L_1 ⊔⊔_T L_2$ is a context-free language.

6 Fairness

Fairness is a property of the parallel composition of processes that, roughly speaking, says that each action of a process is performed with not too much delay with respect to performing actions from another process. That is, the parallel composition is "fair" with both processes that are performed.

Definition 6.1 *Let $T \subseteq \{r, u\}^*$ be a set of trajectories and let n be an integer, $n \geq 1$. T has the n-fairness property iff for all $t \in T$ and for all t' such that $t = t't''$ for some $t'' \in \{r, u\}^*$, it follows that:*

$$| \, |t'|_r - |t'|_u \, | \leq n.$$

□

This means that all trajectories from T are contained in the region of the plane bounded by the line $y = x - n$ and the line $y = x + n$, see Figure 2, for $n = 4$.

Example 6.1 *The balanced literal shuffle ($\sqcup\!\sqcup_b$) has the n-fairness property for all n, $n \geq 1$.*

The following operations: shuffle ($\sqcup\!\sqcup$), catenation ($\cdot$), insertion ($\longleftarrow$) do not have the n-fairness property for any n, $n \geq 1$.

For instance, note that the catenation means shuffle on the set T of trajectories, where (see also Remark 3.1, 5.):

$$T = r^* u^* = \{r^i u^j \mid i, j \geq 0\}.$$

Therefore,

$$\{| \, |t'|_r - |t'|_u| \, \| \, t't'' \in T \text{ for some } t''\} = \{| \, i - j \, \| \, i, j \geq 0\}.$$

Because the values $| \, i - j \, |$, where $i, j \geq 0$, cannot be bounded by any fixed constant n, $n \geq 1$, it follows that the catenation is not n-fair for any $n \geq 1$.

A similar argument is valid to prove that shuffle and insertion operations do not have the n-fairness property for any n, $n \geq 1$.

Definition 6.2 *Let n be a fixed number, $n \geq 1$. Define the language F_n by:*

$$F_n = \{t \in V^* \mid \, | \, |t'|_r - |t'|_u \, | \leq n, \text{ for all } t' \text{ such that } t = t't'', t'' \in V^*\}.$$

□

Remark 6.1 *Note that a set T of trajectories has the n-fairness property if and only if $T \subseteq F_n$.*

□

We omit the straightforward proof of the following statement.

Proposition 6.1 *For every n, $n \geq 1$, the language F_n is a regular language.*

Corollary 6.1 *Let T be a set of trajectories. If T is a context-free or a simple matrix language and n is a fixed number, $n \geq 1$, then it is decidable whether or not T has the n-fairness property.*

Proof. It is easy to observe that for the above families of languages the problem if a language from a family is contained in a regular language is a decidable problem. Hence, from Proposition 6.1, this corollary follows. □

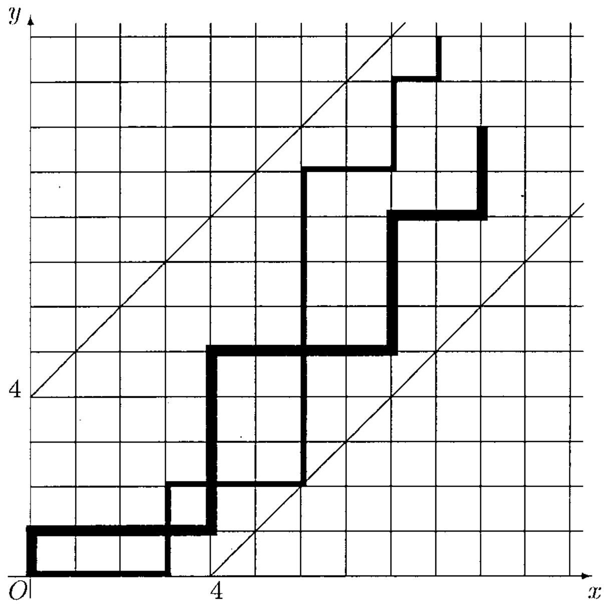

Figure 2

Comment. For a context-free language $T \subseteq V^*$ it is decidable whether or not there exists a non-negative integer n, such that T has the n-fairness property, see [19]. However, in general it is an open problem for what families $\mathcal{L}$ of languages it is decidable this problem.

Now we use the fairness concept in connection with CD grammar systems. Let Γ be a CD grammar system,

$$\Gamma = (N, \Sigma, P_1, P_2, \ldots, P_m, S).$$

Assume that the components of Γ are labelled, such that P_i has the label e_i, $1 \leq i \leq m$. Let E be the set of labels, $E = \{e_1, e_2, \ldots, e_m\}$.

In order to extend the notion of fairness for the general case of CD grammar systems with m components, $m \geq 2$, firstly we define the notion of a m-trajectory. A m-trajectory is an element $t \in E^*$, i.e., a word over the m letters alphabet $\{e_1, e_2, \ldots, e_m\}$.

Definition 6.3 *Let $T \subseteq E^*$ be a set of m-trajectories and let n be an integer, $n \geq 1$. T has the n-fairness property iff for all $t \in T$ and for all t' such that $t = t't''$ for some $t'' \in E^*$, it follows that:*

$$| \, |t'|_{e_i} - |t'|_{e_j} \, | \leq n,$$

for all $1 \leq i, j \leq m$.

A CD grammar Γ has the n-fairness property iff for all terminal derivations

$$S \Longrightarrow_{e_{i_1}} w_1 \Longrightarrow_{e_{i_2}} w_2 \Longrightarrow_{e_{i_3}} \dots \Longrightarrow_{e_{i_k}} w_k$$

the corresponding trajectory $e_{i_1} e_{i_2} \dots e_{i_k}$ has the n-fairness property.

A language L is n-fair, $n \geq 1$, iff there exists a CD grammar system Γ with the n-fairness property such that $L(\Gamma) = L$.

□

Theorem 6.1 *If a language L can be generated by a CD grammar system Γ such that Γ has the n-fairness property for some $n \geq 1$, then L can be generated by a CD grammar system Γ' in the $\leq k$ mode of derivation.*

The converse is not true.

Proof. Observe that for $k = n$ the CD grammar system Γ has the property that $L_{\leq k}(\Gamma) = L$. Hence, one can simply define the CD grammar system Γ' as being Γ.

The converse is not true since a CD grammar system Γ can generate terminal strings in the $\leq k$ mode, just by alternating two of its components and without using the other components. Thus, such a derivation is not n-fair for any n. □

Theorem 6.2 *There exists a non-context-free and semilinear language L such that:*

(i) L can be generated by a CD grammar system in the t mode of derivation.

(ii) L cannot be generated by any n-fair CD grammar system for any $n \geq 1$.

Proof. (*i*) Let L be the following non-context-free and semilinear language:

$$L = \{a^i b^i c^j \mid 1 \leq i \leq j\}.$$

Let Γ be the following CD grammar system:

$$\Gamma = (N, \Sigma, P_1, P_2, P_3, P_4, S),$$

where: $N = \{S, X, X', Y, Y', Y'', Z), \Sigma = \{a, b, c\}$, and the components:

$$P_1 = \{S \longrightarrow XY, S \longrightarrow X'Y', S \longrightarrow X'Y''\},$$

$$P_2 = \{X \longrightarrow aX'b, Y \longrightarrow cY', Y \longrightarrow cY''\},$$

$$P_3 = \{X' \longrightarrow aXb, Y' \longrightarrow cY, Y'' \longrightarrow cZ\},$$

$$P_4 = \{X' \longrightarrow ab, Y'' \longrightarrow c, Y'' \longrightarrow cY''\}.$$

One can easily verify that $L_t(\Gamma) = L$.

(*ii*) In order to prove this statement, assume by contrary that L can be generated by a CD grammar system Γ that has the n-fairness property for some $n \geq 1$.

Clearly, Γ must have a component, say α, that increases the number of occurrences of the symbol a at least by one. Similarly, Γ should have a component, say γ, that increase the number of occurrences of the symbol c by s. Assume that s is the maximum of the number of c symbols that can be produced by a component when it is applied only once.

Since the CD grammar system Γ is n-fair for some fixed $n \geq 1$, it follows that after each n consecutive steps in a derivation the number of occurrences of the symbol a is increased with at least 1 and the number of occurrences of the symbol c with at most $(n-1)s$.

Therefore, if a terminal derivation has length p, where $p = nq + r$, such that $0 \leq r < n$, then the derived word has at least q occurrences of the symbol a and at most $q(n-1)s + rs$ occurrences of the symbol c.

Assume that this derivation produces the terminal word $w = a^i b^i c^j$. Note that $q < i$ and that $j < q(n-1)s + rs < q(n-1)s + ns$. Therefore $j < i(n-1)s + ns$. Note that n and s are fixed constants.

It follows that the CD grammar system Γ cannot generate words $a^i b^i c^j$ with $j \geq i(n-1)s + ns$. This contradicts our assumption that $L(\Gamma) = L$. □

Comment. The above theorem is similar with another, well-known result from the theory of CD grammar-systems, see [3]. The derivation mode $= k$ gives also some idea of fairness. However, it is known that the language

$$L = \{a^{2^n} \mid n \geq 1\}$$

can be generated by a CD grammar system in the t mode, but L cannot be generated by any CD grammar system in the mode $= k$.

Theorem 6.2 provides an example of a language that can be generated by a CD grammar system in mode t, but it cannot be generated by any n-fair CD grammar system for any $n \geq 1$.

7 Parallelization of CD grammar systems

In the following we shall deal with parallelization of languages using shuffle on trajectories.

The *parallelization of a problem* consists in decomposing the problem in subproblems, such that each subproblem can be solved by a processor, i.e., the subproblems are solved in parallel and, finally, the partial results are collected and assembled in the answer of the initial problem by a processor. Solving problems in this way increases the time efficiency. It is known that not every problem can be parallelized. Also, no general methods are known for the parallelization of problems.

Here we formulate the problem in terms of languages and shuffle on trajectories, and present some examples.

The *parallelization of* a language L consists in finding languages L_1, L_2 and T, $T \subseteq V^*$, such that $L = L_1 ⧢_T L_2$ and moreover, the complexity of L_1, L_2 and T is in some sense smaller than the complexity of L. In the sequel the complexity of a language L refers to the Chomsky class of L, i.e., regular languages are less complex than context-free languages that are less complex than context-sensitive languages.

One can easily see that every language L, $L \subseteq \{a,b\}^*$ can be written as $L = a^* ⧢_T b^*$ for some set T of trajectories. However, this is not a parallelization of L since the complexity of T is the same with the complexity of L.

In view of Corollary 5.1 there are non-context-free languages L such that $L = L_1 ⧢_T L_2$ for some context-free languages L_1, L_2 and T. Moreover, one of those three languages can be even a regular language. Note that this is a parallelization of L.

As a first example we consider the non-context-free language $L \subseteq \{a,b,c\}^*$, $L = \{w \mid\mid w \mid_a = \mid w \mid_b = \mid w \mid_c\}$.

Consider the languages: $L_1 \subseteq \{a,b\}^*$, $L_1 = \{u \mid\mid u \mid_a = \mid u \mid_b\}$, $L_2 = c^*$ and $T = \{t \mid\mid t \mid_r = 2 \mid t \mid_u\}$.

One can easily verify that $L = L_1 ⧢_T L_2$. Moreover, note that L_1 and T are context-free languages, whereas L_2 is a regular language. Hence this is a parallelization of L. As a consequence of Corollary 5.1 one cannot expect a significant improvement of this result, for instance to have only one context-free language and two regular languages in the decomposition of L.

Now we consider the case of CD grammar systems. Next example shows how one can define context-free constraints to generate a non-context-free language.

Example 7.1 *Let $\Gamma = (N, \Sigma, S, P_1, P_2)$ be the following CD grammar system: $N = \{S\}$, $\Sigma = \{a,b,c\}$,*

$$P_1 = \{p_1 : S \longrightarrow aS,\ p_2 : S \longrightarrow bS\}$$

$$P_2 = \{q_1 : S \longrightarrow cS,\ q_2 : S \longrightarrow \lambda\}.$$

The constraint language associated to the component P_1 is $L_1 = \{p_1^n p_2^n \mid n \geq 1\}$ and the constraint language associated to the component P_2 is $L_2 = \{q_1^n q_2 \mid n \geq 1\}$. The set of trajectories is $T = \{r^{2n} u^{n+1} \mid n \geq 1\}$. The constraint language associated to the CD grammar system Γ is

$$L_1 ⧢_T L_2 = \{p_1^n p_2^n q_1^n q_2 \mid n \geq 1\}.$$

One can easily verify that the language generated by the CD grammar system Γ with the above constraints is:

$$L(\Gamma) = \{a^n b^n c^n \mid n \geq 1\}.$$

□

Each set T of trajectories from the above examples concerning CD grammar systems does not have the fairness property. However it is not known if the language $L = \{a^n b^n c^n \mid n \geq 1\}$ can be generated by a CD grammar system using constraints with the fairness property.

8 Conclusions

We considered the notion of trajectory in connection with CD grammar systems. The use of trajectories in the theory of CD grammar systems offers some new possibilities to investigate this area. The concept of fairness can be introduced at the level of the components of a CD grammar system, at the level of the productions of a CD grammar system or at the level of the teams used in the derivations of a CD grammar system. For the case of teams, one should put labels to all possible teams and consider as valid only those derivations that follow trajectories from a certain, fixed set of trajectories. Mixed fairness constraints are also possible. For a given CD grammar system, the valid derivations can be defined as being those derivations that satisfy a certain fairness constraint at the level of components and another fairness constraint at the level of productions, etc. Therefore, this framework offers a great flexibility in modelling the fairness phenomenon with respect to CD grammar systems.

Fairness is a natural property of a CD grammar system and it leads to new interesting properties. For instance, it is not known the generative power of the CD grammar systems that use constraints with the fairness property.

There are different natural variants of the fairness property. The fairness property can be considered also with respect to only a part (a fixed subset) of the components (or of productions or teams) of a CD grammar system.

The fairness property can be relaxed or modified in other, different ways. For instance one can consider the restriction that there exists a fixed $n \geq 1$ such that in any terminal derivation, in any n consecutive steps of it, each component does occur at least once, but it does not occur more than k times, where k is a fixed number.

The interrelations between the fairness property and the generative modes t, $= k$, $\leq k$ and $*$ are subjected for further research.

A more general approach, based on geometric considerations, can be considered. Assume that we fix two regions A and B in a many dimensional space (the number of dimensions is equal with the number of versors that encode the trajectories). The regions A and B are not necessarily disjoint. A derivation is considered valid iff the associated trajectory is contained in the region A but it avoids the region B. Note that this approach is an extensions of the notion of fairness depicted in Figure 2. There the region A is the band of the plane bounded by the lines $y = x + 4$ and $y = x - 4$ whereas the region B is empty or any region outside of A.

The idea behind this considerations is also the existence of non-critical sections (devices) described by the region A and of critical sections (devices) described by the region B.

Another important problem is the problem of parallelization of languages, i.e., to express a language as the shuffle of two (or more) other languages over a certain set (or sets) of trajectories. The possibility of decomposing a language as the parallel composition of other, less complex, languages is of theoretical but especially of practical interest. This problem leads to the possibility to perform the parsing operation or other operations, by a parallel machine.

It is an open problem to decide for a given language L (L defined using a CD grammar system) whether or not there exist two languages L_1 and L_2 and a set T of trajectories, such that $L = L_1 \sqcup\!\sqcup_T L_2$.

The problem of parallelization of languages opens new connections between CD grammar systems and the theory of parallel computation.

References

[1] A. Atanasiu and V. Mitrana, "The modular grammars", *Intern. J. Computer Math.*, 30 (1989), 101-122.

[2] E. Csuhaj-Varju and J. Dassow, "On cooperating distributed grammar systems", *J. Inform. Proc. Cybern. EIK*, 26 (1990), 49-63.

[3] E. Csuhaj-Varju, J. Dassow, J. Kelemen and Gh. Păun, *Grammar Systems*, Gordon and Breach, 1993.

[4] J. S. Golan, A. Mateescu and D. Vaida, "Semirings and Parallel Composition of Processes ", Journal of Automata, Languages and Combinatorics, 1 (1996) 3, 199 - 217.

[5] L. Guo, K. Salomaa and S. Yu, *Synchronization Expressions and Languages*, The University of Western Ontario London, Dept. of Comp. Sci., Technical Report 368, 1993.

[6] T. Harju, M. Lipponen and A. Mateescu, "Flatwords and Post Correspondence Problem", *Theoretical Computer Science*, TCS, 161 (1996) 93 - 108.

[7] L. Kari, *On insertion and deletion in formal languages*, PhD Thesis, University of Turku, Turku, Finland, 1991.

[8] L. Kari, A. Mateescu, G. Păun and A. Salomaa, "Teams in cooperating grammar systems", *J. Expt. Theor. Artif. Intell.*, 7(1995) 347-359.

[9] J. Kelemen and A. Kelemenova, "A subsumption arhitecture for generative symbol systems", *Cybernetics and Systems Research '92*, Proc. 11th European Meeting Cybern. Syst. Res. (R. Trappl, ed.), World Scientific, 1992, 1529-1536.

[10] A. Kelemenova and E. Csuhaj-Varju, "Languages of colonies", *2nd Intern. Coll. Words, Languages, Combinatorics*, Kyoto, 1992.

[11] H. C. M. Kleijn and G. Rozenberg, "A study in parallel rewriting systems", *Inform. Control*, 44 (1980), 134-163.

[12] M. Kudlek and A. Mateescu, "Distributed Catenation and Chomsky Hierarchy", FCT'95, Dresden, 1995, Lecture Notes in Computer Science, LNCS 965, Springer-Verlag, 1995, 313-322.

[13] M. Kudlek and A. Mateescu, "Rational and Algebraic Languages with Distributed Catenation", DLT'95, Magdeburg, 1995, in *Developments in Language Theory II*, eds. J. Dassow, G. Rozenberg and A. Salomaa, World Scientific, Singapore, 1996, 129-138.

[14] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, 1986.

[15] A. Mateescu, "On (Left) Partial Shuffle", *Results and Trends in Theoretical Computer Science*, LNCS 812, Springer-Verlag, (1994) 264-278.

[16] A. Mateescu, G. Rozenberg and A. Salomaa, "Shuffle on Trajectories: Syntactic Constraints", TUCS Technical Report, 41, 1996.

[17] A. Mateescu and A. Salomaa, "Formal Languages: an Introduction and a Synopsis", Chapter 1, in *Handbook of Formal Languages*, eds. G. Rozenberg and A. Salomaa, Springer-Verlag, 1997, 1-40.

[18] A. Mateescu and A. Salomaa, "Parallel Composition of Words with Reentrant Symbols", TUCS Technical Report, 15, 1996.

[19] A. Mateescu, K. Salomaa and S. Yu, "Decidability of Fairness for Context-Free Languages", to appear in Proc. of DLT'97, Thessaloniki, July, 1997.

[20] H. A. Maurer, G. Rozenberg and E. Welzl, "Using String Languages to Describe Picture Languages", *Information and Control*, 3, 54, (1982) 155-185.

[21] R. Meersman and G. Rozenberg, "Cooperating grammar systems", *Proc. MFCS '78 Symp., LNCS 64*, Springer-Verlag, 1978, 364-376.

[22] P. H. Nii, "Blackboard systems", in *The Handbook of AI*, vol. 4 (A. Barr, P. R. Cohen, E. A. Feigenbaum, eds.), Addison-Wesley, 1989.

[23] Gh. Păun and L. Sântean, "Parallel communicating grammar systems: the regular case", *Ann. Univ. Buc., Series Matem.-Inform.* 38 (1989), 55-63.

[24] G. Rozenberg and A. Salomaa,(Eds.), *Handbook of Formal Languages*, Springer, 1997.

[25] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.