Proceedings of Grammar Systems Week 2004, edited by E. Csuhaj-Varjú and Gy. Vaszil,
MTA SZTAKI, Budapest, pages 179 - 199, 2004.

Modelling Grammar Systems by Tissue P Systems Working in the Sequential Mode

Rudolf Freund, Marion Oswald

Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9, A-1040 Wien, Austria
{rudi,marion}@emcc.at

Abstract

We consider tissue P systems where rules are applied when moving through a channel from one cell to another one. In a very general manner (i.e., working on arbitrary objects as strings, arrays, graphs, etc.), these tissue P systems equipped with the sequential derivation mode allow for the representation of hybrid co-operating grammar systems using the classic basic derivation modes $*, t$ and $\leq k, = k, \geq k$, for $k \geq 1$, as well as the internally hybrid modes $(\geq k \wedge \leq \ell)$, for $k, \ell \in \mathbf{N}, k \leq \ell$, and $(t \wedge \leq k)$, $(t \wedge = k)$, $(t \wedge \geq k)$, for $k \geq 1$. Moreover, we also show how these tissue P systems working in the sequential mode allow for the simulation of random context grammars, too.

1 Introduction

For the many variants of P systems (introduced as membrane systems in [24]) investigated so far we refer the reader to [25] for a comprehensive overview as well as [28] for the actual state of research. We assume the reader to be familiar with the original definitions and explanations given for these models, as going into more details would go far beyond the scope of an introductory article to a new field of applying the ideas of membrane computing as this one is intended to be. In this paper, we consider a general model of tissue P systems that will allow us to model hybrid co-operating grammar systems when working in the sequential derivation mode, which result will be established for arbitrary object types, e.g., strings and arrays.

Co-operating distributed (CD) grammar systems first were introduced in [20] with motivations related to two-level grammars. Later, grammar systems became a vivid area of research after relating CD grammar systems with Artificial Intelligence (AI) notions [2], such as multi-agent systems or blackboard models for problem solving [22]. A first survey on grammar systems is given in [4]. The main idea of (hybrid) CD grammar systems is the co-operation of several components ("agents") on the same sentential form; non-deterministically, one component takes the sentential form, performs some derivation steps, and according to some specific stopping condition returns it back such that another component may continue the work.

There are several ways to formalize this collaboration. The following (derivation) modes have thoroughly been investigated in the literature:

- $\Longrightarrow^{\leq k}$: the component which takes the sentential form in order to work on it has to perform at most k derivation steps.
- $\Longrightarrow^{=k}$: the component ... has to perform exactly k derivation steps.
- $\Longrightarrow^{\geq k}$: the component ... has to perform at least k derivation steps.
- $\Longrightarrow^{t}$: the component ... has to perform as many derivation steps as possible.
- $\Longrightarrow^{*}$: the component ... can perform arbitrarily many derivation steps.

In CD grammar systems, all components work according to the same mode. In hybrid CD grammar systems introduced by Mitrana and Păun in [21, 23], different components may work in different modes. In a series of papers on hybrid modes in CD grammar systems, the internally hybrid modes $(\geq k \wedge \leq \ell)$, for $k, \ell \in \mathbf{N}, k \leq \ell$, and $(t \wedge \leq k)$, $(t \wedge = k)$, $(t \wedge \geq k)$, for $k \geq 1$, were considered: [7] introduced hybrid modes in CD array grammar systems as a natural specification tool for array languages, [15] investigated accepting CD grammar systems with hybrid modes, while [14] as well as [1] stressed descriptional complexity issues. In [12], results on the combination of the modes t and $\geq k$ were presented, in [13], results on the combination of the modes t with the modes $\leq k$ and $= k$ were presented. Most parts of [12, 13, 14] are contained in the report [9].

The internally hybrid modes informally can be described as follows:

- $\Longrightarrow^{(\geq k_1 \wedge \leq k_2)}$: the component which takes the sentential form in order to work on it has to perform at least k_1 and at most k_2 derivation steps.
- $\Longrightarrow^{(t \wedge \geq k)}$: the component ... has to perform as many derivation steps as possible, and at least k steps.
- $\Longrightarrow^{(t \wedge = k)}$: the component ... has to perform as many derivation steps as possible, and exactly k steps.
- $\Longrightarrow^{(t \wedge \leq k)}$: the component ... has to perform as many derivation steps as possible, and at most k steps.

Combinations $(* \wedge f)$ for $f \in \{*, t\} \cup \{ \leq k, = k, \geq k, \mid k \in \mathbf{N} \}$ are only an alternative notation of the original mode f.

The rest of the paper is organized as follows: In the next section, we start with introducing a general model for (sequential) grammars as well as for random-context grammars and ordered grammars, and then we recall some notions for string grammars and array grammars in the general setting used in this paper. In the third section, we define a general model of grammar systems covering the variants of hybrid co-operating distributed grammar systems considered above. In the fourth section, we define the general model of tissue P systems with channel rules to be used in the following section for modelling hybrid co-operating distributed grammar systems of

arbitrary types. In the sixth section, we show how random-context grammars can be simulated by tissue P systems with channel rules. Some results for the array and the string case that can be derived from the general results proved in the preceding sections are recalled in the seventh section. An outlook to future research concludes the paper.

2 Preliminary Definitions

The set of integers is denoted by $\mathbf{Z}$, the set of non-negative integers by $\mathbf{N}_0$ and the set of positive integers by $\mathbf{N}$. An *alphabet* V is a finite non-empty set of abstract *symbols*. Given V, the free monoid generated by V under the operation of concatenation is denoted by V^*; the elements of V^* are called strings, and the *empty string* is denoted by λ; $V^* \setminus \{\lambda\}$ is denoted by V^+. For more details on formal language theory we refer to [6] and [30].

2.1 Grammar schemes

In the following, we shall deal with various types of objects and grammars, hence, we first define a general model of a grammar scheme:

A *grammar scheme* G is a construct

$(O, O_T, P, \Longrightarrow_G)$ where

- O is the set of *objects*;
- $O_T \subseteq O$ is the set of *terminal* objects;
- P is a finite set of *productions*;
- $\Longrightarrow_G \subseteq O \times O$ is the *derivation relation* of G induced by the productions in P.

The derivation relation $\Longrightarrow_G$ is obtained as the union of all $\Longrightarrow_p \subseteq O \times O$, i.e., $\Longrightarrow_G := \cup_{p \in P} \Longrightarrow_p$, where each $\Longrightarrow_p$ is a relation which we assume at least to be recursive. The reflexive and transitive closure of $\Longrightarrow_G$ is denoted by $\overset{*}{\Longrightarrow}_G$.

In the following we shall consider different types of grammar schemes depending on the components of G, especially with respect to different types of productions.

Based on grammar schemes of specific types, we now define the notion of a (sequential) grammar.

Let $G = (O, O_T, P, \Longrightarrow_G)$ be a grammar scheme. Then the pair (G, w) with $w \in O$ is called a *grammar, w* is the *axiom (start object).*

The *language generated by* (G, w) is the set of all terminal objects (we also assume $v \in O_T$ to be decidable for every $v \in O$) derivable from the axiom, i.e.,

$$L(G, w) = \left\{ v \in O_T \mid w \overset{*}{\Longrightarrow}_G v \right\}.$$

The family of languages generated by grammars of type X is denoted by $\mathcal{L}(X)$.

In many cases, the type X of the grammar scheme allows for the following feature:

A type X of a grammar scheme is called a *type with unit rules* if for every grammar scheme $G = (O, O_T, P, \Longrightarrow_G)$ of type X there exists a grammar scheme $G' = (O, O_T, P \cup P^+, \Longrightarrow_{G'})$ of type X such that

- $P^+ = \{p^+ \mid p \in P\}$,
- for all $x \in O$, p is applicable to x if and only if p^+ is applicable to x, and
- for all $x \in O$, the application of p^+ to x - in case p^+ is applicable to x - yields x back again.

2.2 Random-context grammars and ordered grammars

Let $G = (O, O_T, P, \Longrightarrow_G)$ be a grammar scheme of type X.

A *random-context grammar scheme* G_{RC} of type X is a construct

$$(G, P', p, f, \Longrightarrow_{G_{RC}})$$

where

- P' is a subset of P;
- p is a function assigning a set of *permitting productions* from P to each production in P';
- f is a function assigning a set of *forbidden productions* from P to each production in P';
- $\Longrightarrow_{G_{RC}}$is the derivation relation assigned to G_{RC} such that for any $x, y \in O$, $x \Longrightarrow_{G_{RC}} y$ if and only if $x \Longrightarrow y$ by some q from P' and, moreover, at least one production from $p(q)$ is applicable to x as well as no production from $f(q)$ is applicable to x.

A *random-context grammar* is a pair (G_{RC}, w), where $w \in O$ is the *axiom*. A random-context grammar (scheme) is called a *grammar (scheme) with permitting context* if $f(q) = \emptyset$ for every $q \in P'$ and a *grammar (scheme) with forbidden context* if $p(q) = \emptyset$ for every $q \in P'$. The families of languages generated by random-context grammars, by grammars with permitting context, and by grammars with forbidden context of type X are denoted by $\mathcal{L}(X\text{-}RC)$, $\mathcal{L}(X\text{-}pC)$, and $\mathcal{L}(X\text{-}fC)$, respectively. Obviously, for any arbitrary type X, $\mathcal{L}(X\text{-}yC) \subseteq \mathcal{L}(X\text{-}RC)$ for $y \in \{f, p\}$.

An *ordered grammar scheme* G_O of type X is a construct

$$((O, O_T, P, \Longrightarrow_G), <, \Longrightarrow_{G_O})$$

where $<$ is a partial order relation on the productions in P and $\Longrightarrow_{G_O}$ is the derivation relation assigned to G_O such that for any $x, y \in O$, $x \Longrightarrow_{G_O} y$ if and only if $x \Longrightarrow y$ by some q from P and, moreover, no production r with $r > q$ is applicable to x.

An *ordered grammar* is a pair (G_O, w), where $w \in O$ is the *axiom*. The family of languages generated by ordered grammars is denoted by $\mathcal{L}(X\text{-}O)$.

In the general setting used in this paper, the model of ordered grammars is very much related with the model of grammars with forbidden context:

Theorem 2.1. *For every ordered grammar G_O of type X we can construct a grammar with forbidden context G_{fC} of type X such that $L(G_O) = L(G_{fC})$, i.e., $\mathcal{L}(X\text{-}O) \subseteq \mathcal{L}(X\text{-}fC)$ for any arbitrary type X.*

Proof. Let $G = (O, O_T, P, \Longrightarrow_G)$ be a grammar scheme of type X and let $(G, <, \Longrightarrow_{G_O})$ be an ordered grammar of type X. Then we construct the grammar with forbidden context $\left(G, P, f, \Longrightarrow_{G_{fC}}\right)$ of type X by defining

$$f(q) := \{p \mid p > q\}$$

for all $q \in P$. Obviously, by this construction $L(G_O) = L(G_{fC})$. □

The reverse inclusion only holds true for grammars with forbidden context that are not using additional rules in the sets of forbidden rules:

Theorem 2.2. *For every grammar with forbidden context G_{fC} of type X with*

$$\left((O, O_T, P, \Longrightarrow_G), P, f, \Longrightarrow_{G_{fC}}\right)$$

we can construct an ordered grammar G_O of type X such that $L(G_{fC}) = L(G_O)$.

Proof. We construct the ordered grammar $((O, O_T, P, \Longrightarrow_G), <, \Longrightarrow_{G_O})$ by defining, for all $q \in P$,

$$p > q \text{ if and only if } p \in f(q).$$

Obviously, by this construction $L(G_{fC}) = L(G_O)$. □

2.3 String grammars

A *string grammar scheme* usually is defined as a construct

(N, T, P) where

- N is the alphabet of *non-terminal symbols*;
- T is the set of *terminal* symbols, $N \cap T = \emptyset$;
- P is a finite set of *productions* of the form $u \to v$ with $u \in V^+$ and $v \in V^*$, where $V := N \cup T$.

In the general notion of the preceding subsection, a string grammar scheme G now is represented as

$$((V \cup T)^*, T^*, P, \Longrightarrow_G)$$

where the derivation relation for $u \to v \in P$ is defined as usual by $xuy \Longrightarrow_{u \to v} xvy$ for all $x, y \in V^*$, thus yielding the well-known derivation relation $\Longrightarrow_G$ for the string grammar scheme G. A *string grammar* then is a pair (G, S) where $S \in V - T$ is the start symbol.

As special types of string grammars we consider string grammars with arbitrary productions, context-free productions of the form $A \to v$ with $A \in N$ and $v \in V^*$, λ-free context-free productions of the form $A \to v$ with $A \in N$ and $v \in V^+$, (right-)regular productions of the form $A \to v$ with $A \in N$ and $v \in TV \cup T$, the corresponding types of grammars denoted by $ENUM$, CF, $CF_{-\lambda}$, and REG, thus yielding the families of languages $\mathcal{L}(ENUM)$, i.e., the family of recursively enumerable languages, as well as $\mathcal{L}(CF)$, $\mathcal{L}(CF_{-\lambda})$, and $\mathcal{L}(REG)$, i.e., the families of context-free, λ-free context-free, and regular languages, respectively. Observe that the types $ENUM$, CF, and $CF_{-\lambda}$ are types with unit rules (of the form $w \to w$ for $w \to v \in P$), whereas the type REG (in the definition given above) is not a type with unit rules (therefore, we often allow regular productions to be of the general form $A \to v$ with $A \in N$ and $v \in T^*V \cup T^*$).

2.4 Arrays and array grammars

In this subsection we introduce the basic notions for n-dimensional arrays and array grammar schemes and array grammars (e.g., see [16], [29], [31]).

Let $d \in \mathbf{N}$. Then a *d-dimensional array* $\mathcal{A}$ over an alphabet V is a function $\mathcal{A} : \mathbf{Z}^d \to V \cup \{\#\}$, where $shape(\mathcal{A}) = \{v \in W \mid \mathcal{A}(v) \neq \#\}$ is finite and $\# \notin V$ is called the *background* or *blank-symbol*. We usually write $\mathcal{A} = \{(v, \mathcal{A}(v)) \mid v \in shape(\mathcal{A})\}$.

The set of all d-dimensional arrays over V is denoted by V^{*d}. The *empty array* in V^{*d} with empty shape is denoted by Λ_d. Moreover, we define $V^{+d} = V^{*n} \setminus \{\Lambda_d\}$.

Let $v \in \mathbf{Z}^d$, $v = (v_1, \ldots, v_d)$. The *translation* $\tau_v : \mathbf{Z}^d \to \mathbf{Z}^d$ is defined by $\tau_v(w) = w + v$ for all $w \in \mathbf{Z}^d$, and for any array $\mathcal{A} \in V^{*d}$ we define $\tau_v(\mathcal{A})$, the corresponding d-dimensional array translated by v, by $(\tau_v(\mathcal{A}))(w) = \mathcal{A}(w - v)$ for all $w \in \mathbf{Z}^d$. The vector $(0, \ldots, 0) \in \mathbf{Z}^d$ is denoted by Ω_d.

A *d-dimensional array production* p over V is a triple $(W, \mathcal{A}_1, \mathcal{A}_2)$, where $W \subseteq \mathbf{Z}^d$ is a finite set and $\mathcal{A}_1$ and $\mathcal{A}_2$ are mappings from W to $V \cup \{\#\}$ such that $shape(\mathcal{A}_1) \neq \emptyset$. We say that the array $\mathcal{B}_2 \in V^{*d}$ is *directly derivable* from the array $\mathcal{B}_1 \in V^{*d}$ by the d-dimensional array production $(W, \mathcal{A}_1, \mathcal{A}_2)$, i.e., $\mathcal{B}_1 \Longrightarrow_p \mathcal{B}_2$, if and only if there exists a vector $v \in \mathbf{Z}^d$ such that $\mathcal{B}_1(w) = \mathcal{B}_2(w)$ for all $w \in \mathbf{Z}^d \setminus \tau_v(W)$ as well as $\mathcal{B}_1(w) = \mathcal{A}_1(\tau_{-v}(w))$ and $\mathcal{B}_2(w) = \mathcal{A}_2(\tau_{-v}(w))$ for all $w \in \tau_v(W)$, i.e., the sub-array of $\mathcal{B}_1$ corresponding to $\mathcal{A}_1$ is replaced by $\mathcal{A}_2$, thus yielding $\mathcal{B}_2$.

A *d-dimensional array grammar scheme* is a grammar

$$\left((N \cup T)^{*d}, T^{*d}, P, \Longrightarrow_G\right) \text{ where}$$

- N is the alphabet of *non-terminal symbols*;
- T is the set of *terminal* symbols, $N \cap T = \emptyset$;
- P is a finite set of *array productions* over V, $V := N \cup T$;

- $\Longrightarrow_G$ is the derivation relation induced by the array productions in P according to the explanations given above, i.e., for arbitrary $\mathcal{B}_1, \mathcal{B}_2 \in V^{*d}$, $\mathcal{B}_1 \Longrightarrow_G \mathcal{B}_2$, if and only if there exists a d-dimensional array production $p = (W, \mathcal{A}_1, \mathcal{A}_2)$ in P such that $\mathcal{B}_1 \Longrightarrow_p \mathcal{B}_2$.

A d-dimensional array production $p = (W, \mathcal{A}_1, \mathcal{A}_2)$ in P is called

- *#-context-free*, if $shape(\mathcal{A}_1) = \{\Omega_d\}$; a d-dimensional #-context-free array production $p = (W, \mathcal{A}_1, \mathcal{A}_2)$ in the following will be represented in the form $\mathcal{A}_1(\Omega_d) \to \mathcal{A}_2(\Omega_d)\{(v, \mathcal{A}_2(v)) \mid v \in U\}$ with $U = W - \{\Omega_d\}$.
- *context-free*, if it is #-context-free and $shape(\mathcal{A}_1) \subseteq shape(\mathcal{A}_2)$;
- *strictly context-free*, if it is context-free and $shape(\mathcal{A}_2) = W$;
- *regular*, if it is strictly context-free and of one of the following forms:
  1. $A \to b$, $A \in N$, $b \in T$;
  2. $A \to b\{(v, C)\}$, $A, C \in N$, $b \in T$, and v is a vector with norm 1.

A d-dimensional #-context-free array production $p = (W, \mathcal{A}_1, \mathcal{A}_2)$ in the following will be represented in the form $\mathcal{A}_1(\Omega_d) \to \mathcal{A}_2(\Omega_d)\{(v, \mathcal{A}_2(v)) \mid v \in U\}$ with $U = W - \{\Omega_d\}$.

A *d-dimensional array grammar* is a pair (G, w) grammar where G is a d-dimensional array grammar scheme and $\{(v_0, S)\}$ with $S \in N$ and $v_0 \in \mathbf{Z}^d$ is the *start array (axiom)*.

An array grammar (scheme) is said to be of type *d-$ENUMA$*, *d-#-CFA*, *d-CFA*, *d-$SCFA$*, *d-$REGA$*, respectively, if every array production in P is of the corresponding type, i.e., a d-dimensional arbitrary, #-context-free, context-free, strictly context-free or regular array production, respectively. The corresponding families of d-dimensional array languages of type X are denoted by $\mathcal{L}(X)$. $\mathcal{L}(d\text{-}ENUMA)$ is the family of recursively enumerable d-dimensional array languages. Observe that only the types *d-$ENUMA$*, *d-#-CFA*, and *d-CFA* are types with unit rules.

3 A General Model for Grammar Systems

In this section we define a general model of grammar systems covering the variants of hybrid co-operating distributed grammar systems considered in this paper.

Let

$$G = (O, O_T, P, \Longrightarrow_G)$$

be a grammar scheme of arbitrary type X. A *hybrid co-operating distributed grammar scheme* G_{HCD} of type X is a construct

$$\left(G, P_1, ..., P_n, \Longrightarrow_{G_{HCD}}^{(f_1,...,f_n)}\right) \text{ where}$$

predicate	definition
$Q(=k,m,i,y)$	$m = k$
$Q(\leq k,m,i,y)$	$m \leq k$
$Q(\geq k,m,i,y)$	$m \geq k$
$Q(*,m,i,y)$	$m \geq 0$
$Q(t,m,i,y)$	$\neg\exists z(y \Rightarrow_i z)$
$Q((f_1 \wedge f_2),m,i,y)$	$Q(f_1,m,i,y) \wedge Q(f_2,m,i,y)$

Table 1: Definition of predicate Q for $k \in \mathbf{N}$ and $f_1, f_2 \in B$

- $P_i \subseteq P$, $1 \leq i \leq n$, $n \geq 1$, are *tables* of productions;
- $\Longrightarrow_{G_{HCD}}^{(f_1,\ldots,f_n)}$ is the derivation relation for G_{HCD}, which is defined as follows: For two objects $x, y \in O$ define $x \Longrightarrow_i y$ if and only if y can be derived from x by applying a production from P_i according to the derivation relation $\Longrightarrow_G$; $x \overset{m}{\Longrightarrow}_i y$ stands for a derivation in m, $m \geq 0$, such derivation steps. Now define the *classic basic (derivation) modes*$B = \{ *, t \} \cup \{ \leq k, = k, \geq k \mid k \in \mathbf{N} \}$ and let $D = B \cup \{ (\geq k \wedge \leq \ell) \mid k, \ell \in \mathbf{N}, k \leq \ell \} \cup \{ (t \wedge \leq k), (t \wedge = k), (t \wedge \geq k) \mid k \in \mathbf{N} \}$. For $f \in D$, we first define a predicate Q (see Table 1) and then the relation $\Longrightarrow_i^f$ by $x \Longrightarrow_i^f y$ if and only if $\exists m \geq 0 : (x \overset{m}{\Longrightarrow}_i y \wedge Q(f,m,i,y))$. For two arbitrary objects $x, y \in O$, now $x \Longrightarrow_{G_{HCD}}^{(f_1,\ldots,f_n)} y$ if and only if for some i, $1 \leq i \leq n$, $x \Longrightarrow_i^{f_i} y$.

A *hybrid co-operating distributed grammar system* G_{HCD} (HCD grammar system) of type X is a pair (G_{HCD}, w) where $w \in O$ is the axiom. The language *generated* by a HCD grammar system is defined as:

$$L(G_{HCD}) := \Big\{ u \in O_T \mid w \Longrightarrow_{i_1}^{f_{i_1}} w_1 \ldots \Longrightarrow_{i_m}^{f_{i_m}} w_m = u \\ 1 \leq i_j \leq n, 1 \leq j \leq m, m \geq 1 \}$$

If $F \subseteq D$, then the family of languages generated by HCD grammar systems of type X with degree at most n, each component working in one of the modes contained in F, is denoted by $\mathcal{L}(HCD_n, X, F)$. In a similar way, we write $\mathcal{L}(HCD_*, X, F)$ when the number of components is not restricted. If F is a singleton $\{f\}$, we simply write $\mathcal{L}(HCD_n, X, f)$, where $n \in \mathbf{N} \cup \{*\}$; in that case, the HCD grammar system is called a CD grammar system, and we also write $L_f(G)$ instead of $L(G)$ to denote the language generated by the CD grammar system G_{CD} in the mode f.

Observe that in the string case usually a hybrid co-operating distributed grammar system $G_{HCD} = \left((V^*, T^*, P, \Longrightarrow_G), P_1, ..., P_n, \Longrightarrow_{G_{HCD}}^{(f_1,\ldots,f_n)}, w \right)$ is written in the form

$$(V - T, T, (P_1, f_1), ..., (P_n, f_n), w)$$

for some $w \in V - T$.

4 A General Model of Tissue P Systems with Channel Rules

In this section we define the general model of tissue P systems in which we are going to represent the different variants of grammar systems considered in this paper; it is based on the model of tissue-like P systems with channel states introduced in [17] and works in the sequential derivation mode (see [18] and [19]).

Let

$$G = (O, O_T, P, \Longrightarrow_G)$$

be a grammar scheme of arbitrary type X. *A tissue P system* (of degree $m \geq 1$) *with channel rules (tP system* for short) Π of type X $(-, 0, +)$ is a construct

$$\left(G, m, W, syn, \left(R_{(i,j)}\right)_{(i,j)\in syn}, i_o\right),$$

where

- m is the number of *cells* assumed to be labelled with $1, 2, \ldots, m$;
- $W \subseteq O^m$ are m strings over O representing the *initial finite multisets* of objects present in the m cells of the system (we here do not consider communication with the environment);
- $syn \subseteq \{(i, j) \mid i, j \in \{1, 2, \ldots, m\}\}$ is the set of *links* (also called *synapses* or *channels*, between two cells);
- $R_{(i,j)}$ is of the form R, $+R$ or $-R$, where $R \subseteq P$ is a finite set of *rules* (we say that $R_{(i,j)}$ is associated with the channel $(i, j) \in syn$); syn and $\left(R_{(i,j)}\right)_{(i,j)\in syn}$ represent the *connection graph* of Π;
- $i_o \in \{1, 2, \ldots, m\}$ is the *output* cell.

The computation starts with the configuration specified by W. In each time unit, a computation step takes place; in the *sequential derivation mode* ($\Longrightarrow_{\Pi}^{sequ}$) we first choose a synapse (i, j) such that,

R if $R_{(i,j)} = R$ and the application of a rule from R applied to some object x in cell i yields y, then x is removed from cell i and y is added in cell j, or

$+R$ if $R_{(i,j)} = +R$ and if each rule from R can be applied to some object x in cell i, then this object x can move from cell i to cell j remaining unchanged, or

$-R$ if $R_{(i,j)} = -R$ and if no rule from R can be applied to x, then this object x can move from cell i to cell j remaining unchanged.

Observe that we also allow $R = \emptyset$, i.e., a transition via $-\emptyset$ is always possible (and a transition via $+\emptyset$ is never possible and therefore makes no sense). The reflexive and transitive closure of the derivation relation $\Longrightarrow_{\Pi}^{sequ}$ is denoted by $\overset{*}{\Longrightarrow}_{\Pi}^{sequ}$. If none of the variants of a derivation step described above is possible for any of the synapses, then the derivation stops (*halts*), yet in contrast to many other variants of (tissue) P systems we do not only consider terminal objects in a halting computation as results, instead the results of a computation are described by the terminal objects from O_T present in cell i_o at any step during an arbitrary computation, i.e., the *language generated by* Π *in the sequential derivation mode* is defined as

$$L(\Pi, sequ) = \left\{ u \in O_T \mid W \overset{*}{\Longrightarrow}_{\Pi}^{f} U, u = pr_{i_0}(U) \right\};$$

the configuration of Π can be described by a vector whose components are the contents of all m cells i; pr_i is a projection yielding the contents of cell i. The family of languages generated by tP systems of type $X(-,0,+)$ is denoted by $\mathcal{L}(tP, X(-,0,+))$.

If no synapse $R_{(i,j)} \in syn$ in $\Pi = \left(G, m, W, syn, (R_{(i,j)})_{(i,j)\in syn}, i_o\right)$ is of the form $+R$, then we call Π a tP system of type $X(-,0)$, if no synapse $R_{(i,j)} \in syn$ in Π is of the form $-R$, then we call Π a tP system of type $X(0,+)$, and if all synapses are of the form R only, then we call Π a tP system of type $X(0)$. The families of languages generated by tP systems of type $X(-,0)$, $X(+,0)$, and $X(0)$ are denoted by $\mathcal{L}(tP, X(-,0))$, $\mathcal{L}(tP, X(0,+))$, and $\mathcal{L}(tP, X(0))$, respectively.

5 Tissue P Systems and Grammar Systems

In this section we give a purely structural proof for the main result saying that grammar systems of arbitrary type can be modelled by tissue P systems of the same type working in the sequential derivation mode.

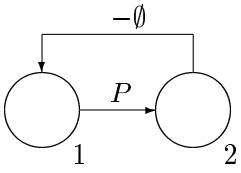

Figure 1: $*$-mode, $= 1$-mode, ≥ 1-mode, $\leq k$-mode

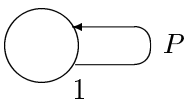

Figure 2: $*$-mode in extended variant

Theorem 5.1. *For every HCD grammar system G_{HCD} of type X we can construct a tP system of type $X(-,0)$ working in the sequential derivation mode such that*

$L(G_{HCD}) = L(\Pi, sequ)$, i.e., $\mathcal{L}(HCD_*, X, F) \subseteq \mathcal{L}(tP, X(-,0))$ *for any arbitrary non-empty set of derivation modes* F.

Proof (sketch). Let $G = (O, O_T, P, \Longrightarrow_G)$ be a grammar scheme of type X and let

$$G_{HCD} = \left(G, P_1, ..., P_n, \Longrightarrow_{G_{HCD}}^{(f_1,...,f_n)}, w\right)$$

be a HCD grammar system of type X. Then we construct the corresponding tP system

$$\Pi = (G, m, W, syn, (R_{(i,j)})_{(i,j)\in syn}, i_o)$$

as follows:

We take cell 1 as starting point of our derivations as well as the output cell i_0, too, i.e.

- $W = (w, \emptyset, ..., \emptyset)$ and
- $i_0 = 1$.

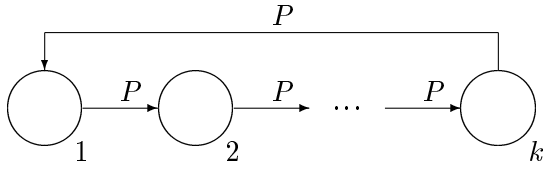

Figure 3: $= k$-mode

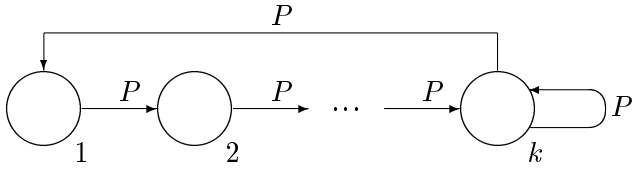

Figure 4: $\geq k$-mode

For each pair (P_i, f_i), $1 \leq i \leq n$, we now construct a number of cells together with the corresponding synapses; instead of giving formal descriptions, we describe the structure of the necessary cells and their synapses by illustrative figures; observe that the "starting point" of all simulations is cell 1.

The derivation modes $*$, $= 1$, ≥ 1, ≤ 1, and even $\leq k$ for $k \geq 2$ have the same effect and can be simulated in Π as described in Figure 1, i.e., for a pair (P, f) with $f \in \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$ we only need a simple loop and only one additional cell (as a technical detail we mention that we need to go back with a synapse having assigned $-\emptyset$ because each synapse may carry only one $-R$ or R; we could avoid this by allowing finite sets of such objects assigned to a synapse, which would yield the even simpler solution depicted in Figure 2.

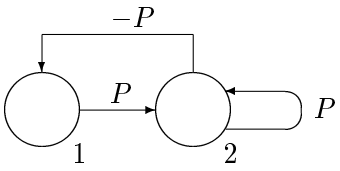

Figure 5: t-mode

For a pair (P, f) with $f \in \{= k \mid k \geq 2\}$ we need a simple loop with $k-1$ additional cells and the structure as depicted in Figure 3.

For a pair (P, f) with $f \in \{\geq k \mid k \geq 2\}$ we again need a simple loop with $k-1$ additional cells as well as an additional loop at cell k and the structure as depicted in Figure 4.

A pair (P, t) with the t-mode needs only one additional cell, but a nontrivial synapse with $-P$ as depicted in Figure 5.

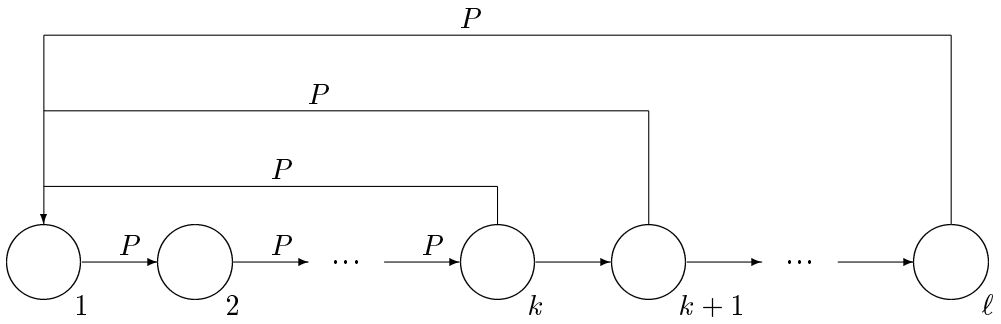

Figure 6: $(\geq k \wedge \leq \ell)$-mode, $k < l$

Suitable simulations of the internally hybrid modes $(\geq k \wedge \leq \ell)$, for $k, \ell \in \mathbf{N}, k < \ell$ (observe that for $k = l$ the derivation mode $(\geq k \wedge \leq \ell)$ coincides with the mode $= k$) and $(t \wedge \leq k)$, $(t \wedge = k)$, $(t \wedge \geq k)$, for $k \geq 1$, are illustrated in Figures 6, 7, 8, and 9.

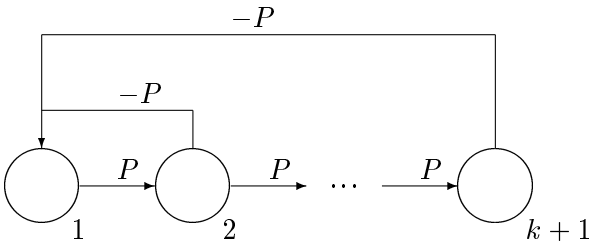

Figure 7: $(t \wedge \leq k)$-mode

In a depictive way, all the figures explained above show how the tP system Π can easily be constructed from the given HCD grammar system G_{HCD} of type X. Moreover, from the given construction we immediately infer $L(G_{HCD}) = L(\Pi, sequ)$; again we should like to stress that this result holds true for any arbitrary type X.□

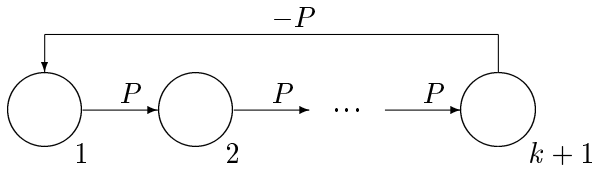

Figure 8: $(t\wedge = k)$-mode

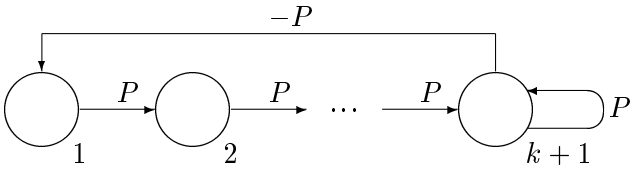

Figure 9: $(t\wedge \geq k)$-mode

Now let X be a type with unit rules. Then except for the t-mode and the hybrid modes containing the t-mode, tP systems of type $X\,(0)$ are sufficient:

Corollary 5.2. *For every HCD grammar system* G_{HCD}

$$G_{HCD} = \left((O, O_T, P, \Longrightarrow_G), P_1, ..., P_n, \Longrightarrow_{G_{HCD}}^{(f_1,...,f_n)}, w \right)$$

of a type X *with unit rules such that* $\{f_1, ..., f_n\}$ *is a subset of*

$$\{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\} \cup \{(\geq k\wedge \leq \ell) \mid k, \ell \in \mathbf{N}, k \leq \ell\}$$

we can construct a tP system of type $X\,(0)$ *working in the sequential derivation mode such that* $L\,(G_{HCD}) = L\,(\Pi, sequ)$, *i.e.,*

$$\mathcal{L}\,(HCD_*, X, F) \subseteq \mathcal{L}\,(tP, X\,(0))$$

for any arbitrary non-empty set of derivation modes

$$F \subseteq \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\} \cup \{(\geq k\wedge \leq \ell) \mid k, \ell \in \mathbf{N}, k \leq \ell\}.$$

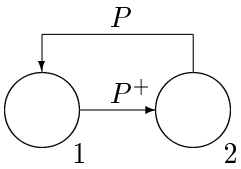

Figure 10: non-t-modes with unit rules

Proof (sketch). In the construction of the tP system in the proof of Theorem 5.1, the $*$-mode now can be simulated as shown in Figure 10, where P^+ is the set of unit rules corresponding to the set of rules P. □

For more restricted sets of derivation modes, the result of the preceding corollary even holds for arbitrary types X; moreover, we only need one synapse as shown in Figure 11:

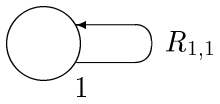

Figure 11: $*$-mode with unit rules

Corollary 5.3. *For every HCD grammar system* G_{HCD}

$$G_{HCD} = \left((O, O_T, P, \Longrightarrow_G), P_1, ..., P_n, \Longrightarrow_{G_{HCD}}^{(f_1,...,f_n)}, w \right)$$

of an arbitrary type X *such that*

$$\{f_1, ..., f_n\} \subseteq \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$$

we can construct the tP system

$$\Pi = \left((O, O_T, P, \Longrightarrow_G), 1, W, \{1\}, R_{(1,1)}, 1 \right)$$

of type $X\,(0)$ *with* $R_{(1,1)} = \cup_{i=1}^{n} P_i$ *working in the sequential derivation mode such that* $L\,(G_{HCD}) = L\,(\Pi, sequ)$*; hence,*

$$\mathcal{L}\,(HCD_*, X, F) = \mathcal{L}\,(X) \subseteq \mathcal{L}\,(tP, X\,(0))$$

for any arbitrary non-empty set of derivation modes

$$F \subseteq \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}.$$

For the inverse inclusion relations we just mention that the tP systems investigated in this section exactly characterize the corresponding variants of HCD grammar systems when having the specific connection graphs, i.e., the structures of synapses and their connections, as described above, the other cells always being arranged around the output cell, which also carries the only input object.

6 Tissue P Systems and Random-Context Grammars

To prove our next result in the general case, we need channel rules $R_{(i,j)}$ of the form $+R$ (which allows us to check for permitting context or to check for competence as this is called by Jürgen Dassow in [5]).

Theorem 6.1. *For every random-context grammar* G_{RC} *of type* X *we can construct a tP system* Π *of type* $X\,(-, 0, +)$ *working in the sequential derivation mode such that* $L(G_{RC}) = L(\Pi, sequ)$*, i.e.,* $\mathcal{L}\,(X\text{-}RC) \subseteq \mathcal{L}\,(tP, X\,(-, 0, +))$.

Proof. Let $G = (O, O_T, P, \Longrightarrow_G)$ be a grammar scheme of type X and let

$$G_{RC} = (G, P', p, f, \Longrightarrow_{G_{RC}})$$

be a random-context grammar of type X with $P' = \{q_1, ..., q_n\}$. Then we construct the corresponding tP system

$$\Pi = (G, 2n+1, W, syn, (R_{(i,j)})_{(i,j)\in syn}, 1)$$

by using the simulation described in Figure 12 for every production q_i in P', i.e.:

- $syn = \{(1, i+1), (i+1, 2i+1), (2i+1, 1) \mid 1 \leq i \leq n\}$;
- $R_{(1,i+1)} = +p(q_i)$, $1 \leq i \leq n$;
- $R_{(i+1,2i+1)} = -f(q_i)$, $1 \leq i \leq n$;
- $R_{(2i+1,1)} = \{q_i\}$, $1 \leq i \leq n$.

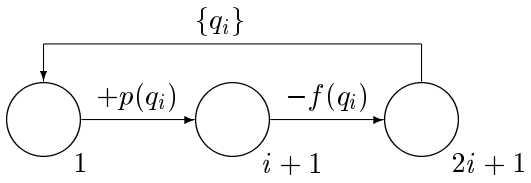

Figure 12: Π simulating G_{RC}

Observe that even in case $f(q_i) = \emptyset$ the simulation still works correctly. In the case $p(q_i) = \emptyset$ we have to omit one cell and the channel rule $+p(q_i)$ (see Figur 13); another possibility is to set $p(q_i) := \{q_i\}$ instead, which is always possible, too. Obviously, by this construction $L(G_{RC}) = L(\Pi, sequ)$; again we should like to stress that this result holds true for any arbitrary type X. □

The following corollaries for grammars with forbidden context G_{fC} and grammars with permitting context G_{pC} are immediate consequences of the construction given in the proof of the preceding theorem:

Corollary 6.2. *For any arbitrary type X, $\mathcal{L}(X\text{-}fC) \subseteq \mathcal{L}(tP, X(-, 0))$.*

Proof. The simulation now works as depicted in Figure 13, i.e., we construct the corresponding tP system

$$\Pi = (G, n+1, W, syn, (R_{(i,j)})_{(i,j)\in syn}, 1)$$

with

- $syn = \{(1, i+1), (i+1, 1) \mid 1 \leq i \leq n\}$;
- $R_{(1,i+1)} = -f(q_i)$, $1 \leq i \leq n$;
- $R_{(i+1,1)} = \{q_i\}$, $1 \leq i \leq n$. □

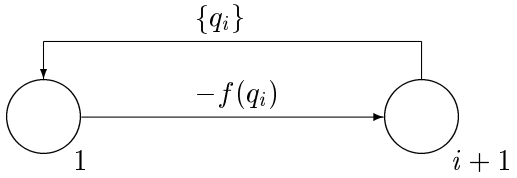

Figure 13: Π simulating G_{fC}

Corollary 6.3. *For any arbitrary type* X, $\mathcal{L}(X\text{-}pC) \subseteq \mathcal{L}(tP, X(0,+))$.

Proof. The simulation now works as depicted in Figure 14, i.e., we construct the corresponding tP system

$$\Pi = (G, n+1, W, syn, (R_{(i,j)})_{(i,j)\in syn}, 1)$$

with

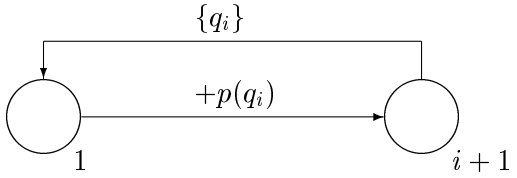

Figure 14: Π simulating G_{pC}

- $syn = \{(1, i+1), (i+1, 1) \mid 1 \le i \le n\}$;
- $R_{(1,i+1)} = +p(q_i)$, $1 \le i \le n$;
- $R_{(i+1,1)} = \{q_i\}$, $1 \le i \le n$.

As a special technical proof detail we mention that without loss of generality we may assume $q_i \in p(q_i)$. □

If the underlying type X is a type with unit rules, we do not need channel rules $R_{(i,j)}$ of the form $+R$ as is shown in Figure 15 and Figure 16 - observe that in case of $p(q_i) = \emptyset$ we have to take $p(q_i)^+ := \{q_i\}^+$ ($= \{q_i{}^+\}$); hence, we obtain the following results:

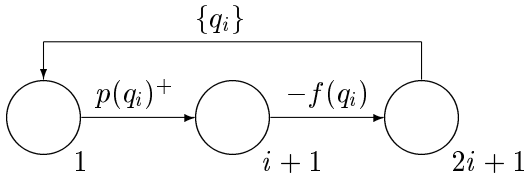

Figure 15: Π simulating G_{RC} with unit rules

Corollary 6.4. *For any type* X *with unit rules,* $\mathcal{L}(X\text{-}RC) \subseteq \mathcal{L}(tP, X(-,0))$.

Corollary 6.5. *For any type* X *with unit rules,* $\mathcal{L}(X\text{-}pC) \subseteq \mathcal{L}(tP, X(0))$.

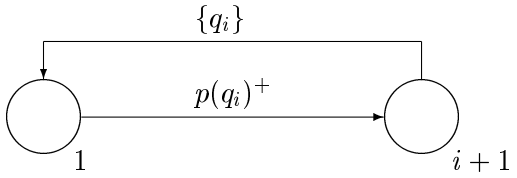

Figure 16: Π simulating G_{pC} with unit rules

7 Tissue P Systems Working on Strings and Arrays

Based on the general result presented in the preceding section, we can immediately infer the corresponding results for the types of string and array grammars defined in the second section.

For example, in the string case we cite the following two results from Theorem 19 and Corollary 21 in [13]:

Corollary 7.1. *If* $\emptyset \neq F \subseteq \{ =k, \geq k, (\geq k \wedge \leq \ell), (t\wedge \geq k) \mid 2 \leq k \leq \ell \}$, *then*

$$\mathcal{L}(HCD_*, CF, F \cup \{(t\wedge = 1)\}) = \mathcal{L}(ENUM).$$

Corollary 7.2. *Let* $\emptyset \neq F \subseteq \{*, t\} \cup \{ \leq k, =k, \geq k, (t\wedge \geq k) \mid k \geq 1 \} \cup \{ (\geq k \wedge \leq \ell) \mid k, \ell \geq 1, k \leq \ell \}$. *Then, for every* $m \geq 2$,

$$\mathcal{L}(HCD_*, CF, F \cup \{(t\wedge = m)\}) = \mathcal{L}(ENUM).$$

Obviously, from these results we immediately obtain the following one due to Theorem 5.1:

Corollary 7.3. *For every recursively enumerable string language* L *we can construct a tP system* Π *of type* CF *such that* $L(\Pi, sequ) = L$, *i.e.,*

$$\mathcal{L}(ENUM) = \mathcal{L}(tP(-, 0), CF).$$

The structure of the connection graph of the tP system Π in Corollary 7.3 depends on the set of derivation modes F we take according to Corollary 7.1 or Corollary 7.2.

Moreover, from Corollary 5.3 we immediately obtain the following result:

Corollary 7.4. *Let* $\emptyset \neq F \subseteq \{*, =1, \geq 1\} \cup \{ \leq k \mid k \geq 1 \}$. *Then*

$$\mathcal{L}(HCD_*, X, F) = \mathcal{L}(X)$$

for $X \in \{ENUM, CF, CF_{-\lambda}\} \cup \{d\text{-}ENUMA, d\text{-}\#\text{-}CFA, d\text{-}CFA \mid d \geq 1\}$.

Several other results proved in the preceding sections carry over to the case of array grammar systems of various types (e.g., see [3]), too. Following the results elaborated in [8] and [11], we even get applications of tP systems of types 2-(#-)CFA in the area of character recognition, e.g., see [10].

A generalization of the two-dimensional case to arbitrary dimensions of a result proved in [16] shows that, for any arbitrary dimension $d \geq 1$,

$$\mathcal{L}(d\text{-}ENUMA) = \mathcal{L}(d\text{-}\#\text{-}CFA\text{-}O);$$

according to Theorem 2.1,

$$\mathcal{L}(d\text{-}\#\text{-}CFA\text{-}O) \subseteq \mathcal{L}(d\text{-}\#\text{-}CFA\text{-}fC);$$

due to Corollary 6.2,

$$\mathcal{L}(d\text{-}\#\text{-}CFA\text{-}fC) \subseteq \mathcal{L}(tP, d\text{-}\#\text{-}CFA(-,0)).$$

In sum, we get the following result:

Corollary 7.5. *For any arbitrary dimension* $d \geq 1$,

$$\mathcal{L}(d\text{-}ENUMA) = \mathcal{L}(d\text{-}\#\text{-}CFA\text{-}O) =$$
$$\mathcal{L}(d\text{-}\#\text{-}CFA\text{-}fC) = \mathcal{L}(tP, d\text{-}\#\text{-}CFA(-,0)).$$

Without proof we should like to mention that for $d \in \{1,2,3\}$ we could even show that

$$\mathcal{L}(d\text{-}ENUMA) = \mathcal{L}(tP, d\text{-}\#\text{-}CFA(0)).$$

8 Summary and Future Research

We have shown that hybrid co-operating distributed grammar systems of arbitrary types equipped with arbitrary classic basic derivation modes as well as internally hybrid derivation modes can be simulated by tissue P systems with channel rules of the corresponding types when working in the sequential derivation mode and with only specific structures of the underlying connection graph. Moreover, we have shown how random-context grammars can be simulated by tissue P systems, too.

This paper has to be seen as a starting point for future research only. Many technical details remain for being investigated in a more precise way. Moreover, we have not considered other models of grammar systems, for example, parallel communicating grammar systems, as has been done by Gheorghe Păun, see [26].

Acknowledgements

The authors acknowledge many interesting and inspiring discussions with Francesco Bernardini, Erzsébet Csuhaj-Varjú, Jürgen Dassow, and Gheorghe Păun in the areas of grammar systems and membrane computing, especially during the Grammar Systems Week having taken place in Budapest at the beginning of July 2004 as well as IST-2001-32008 project "MolCoNet".

References

[1] H. Bordihn, M. Holzer: On a hierarchy of languages generated by cooperating distributed grammar systems. *Inform. Process. Lett.* **69**, 2 (1999) 59–62

[2] E. Csuhaj-Varjú, J. Dassow: On cooperating/distributed grammar systems. *J. Inf. Process. Cybern. EIK*, 26, 1/2 (1990) 49–63

[3] J. Dassow, R. Freund, Gh. Păun: Cooperating array grammar systems. *Int. Journ. of Pattern Recognition and Artificial Intelligence* **9**, 6 (1995) 1–25

[4] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun: *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach, London (1994)

[5] J. Dassow: On cooperating distributed grammar systems with competence based start and stop conditions. In: E. Csuhaj-Varjú, Gy. Vaszil (eds.): Prel. Proceedings of Grammar Systems Week 2004, MTA SZTAKI, Budapest (2004) 146–151

[6] J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory.* Springer-Verlag, Berlin (1989)

[7] H. Fernau, R. Freund: Bounded parallelism in array grammars used for character recognition. In: P. Perner, P. Wang, A. Rosenfeld (eds.): *Proceedings of Structural and Syntactical Pattern Recognition SSPR'96*, Lecture Notes in Computer Science **1121**. Springer, Berlin (1996) 40–49

[8] H. Fernau, R. Freund: Accepting array grammars with control mechanisms. In: Gh. Păun, A. Salomaa (eds.): *New Trends in Formal Languages*, Lecture Notes in Computer Science **1218**. Springer, Berlin (1997) 95–118

[9] H. Fernau, R. Freund, M. Holzer: External versus internal hybridization for cooperating distributed grammar systems. Technical Report TR 185–2/FR–1/96, Technische Universität Wien, Austria (1996)

[10] H. Fernau, R. Freund, M. Holzer: Character recognition with k-head finite array automata. In: A. Amin et al. (eds.): *Proceedings of Structural and Syntactical Pattern Recognition SSPR'98*, Lecture Notes in Computer Science **1451**. Springer, Berlin (1998) 282–291

[11] H. Fernau, R. Freund, M. Holzer: Regulated array grammars of finite index. In: Gh. Păun, A. Salomaa (eds.): *Grammatical Models of Multi-Agent Systems.* Gordon and Breach, London (1999) 157–181 (Part I) and 284–296 (Part II)

[12] H. Fernau, R. Freund, M. Holzer: Hybrid modes in cooperating distributed grammar systems: internal versus external hybridization. *Theoretical Computer Science* **259** (2001) 405–426

[13] H. Fernau, R. Freund, M. Holzer: Hybrid modes in cooperating distributed grammar systems: combining the t-mode with the modes $\leq k$ and $= k$. *Theoretical Computer Science* **299** (2003) 633–662

[14] H. Fernau, R. Freund, M. Holzer: Bounding resources in cooperating distributed grammar systems. In: S. Bozapalidis (ed.): *Proc. of the 3rd Int. Conf. Developments in Language Theory.* Aristotle University of Thessaloniki (1997) 261–272

[15] H. Fernau, M. Holzer: Accepting multi-agent systems II. *Acta Cybernetica* 12 (1996) 361–379

[16] R. Freund: Control mechanisms on #-context-free array grammars. In: Gh. Păun (ed.): *Mathematical Aspects of Natural and Formal Languages.* World Scientific Publ., Singapore (1994) 97–137

[17] R. Freund, Gh. Păun, M.J. Pérez Jiménez: Tissue-like P systems with channel states. In: [27] (2004) 206–223

[18] R.Freund: Asynchronous P systems. In: *Proceedings of WMC 5,* Milano, Italy, June 2004 (2004)

[19] R.Freund: Asynchronous P systems on arrays and strings. Accepted for presentation at *DLT 2004* (2004)

[20] R. Meersman, G. Rozenberg: Cooperating grammar systems. In: *Proceedings of Mathematical Foundations of Computer Science MFCS'78*, Lecture Notes in Computer Science **64**. Springer, Berlin (1978) 364–374

[21] V. Mitrana: Hybrid cooperating/distributed grammar systems. *Computers and Artificial Intelligence* **12**, 1 (1993) 83–88

[22] N. J. Nilsson: *Principles of Artificial Intelligence.* Springer, Berlin (1982)

[23] Gh. Păun: On the generative capacity of hybrid CD grammar systems. *J. Inf. Process. Cybern. EIK* **30**, 4 (1994) 231–244

[24] Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences* **61,** 1 (2000) 108–143, and TUCS Research Report 208 (1998) (`http://www.tucs.fi`)

[25] Gh. Păun: *Membrane Computing: an Introduction.* Springer, Berlin (2002)

[26] Gh. Păun: Grammar systems vs. membrane computing: a preliminary approach. In: E. Csuhaj-Varjú, Gy. Vaszil (eds.): *Prel. Proc. of Grammar Systems Week 2004*, MTA Sztaki, Budapest (2004) 225–245

[27] Gh. Păun, A. Riscos Nuñez, A. Romero Jiménez, F. Sancho Caparrini (eds.): Dept. of Computer Sciences and Artificial Intelligence, *Univ. of Sevilla Tech. Report 01/2004*, Second Week on Membrane Computing, Sevilla, Spain, Feb. 2-7 (2004)

[28] The P Systems Web Page: `http://psystems.disco.unimib.it/`

[29] A. Rosenfeld: *Picture Languages.* Academic Press, Reading, MA (1979)

[30] A. Salomaa, G. Rozenberg (eds.): *Handbook of Formal Languages.* Springer, Berlin (1997)

[31] P. S.-P. Wang (ed.): *Array Grammars, Patterns and Recognizers.* World Scientific Series in Computer Science **18.** World Scientific Publ., Singapore (1989)