Noname manuscript No.
(will be inserted by the editor)

Contextual Array Grammars and Array P Systems

Henning Fernau · Rudolf Freund · Markus L. Schmid · K. G. Subramanian · Petra Wiederhold

Received: date / Accepted: date

Abstract Contextual array grammars, with selectors not having empty cells, are considered. A P system model, called *contextual array P systems*, that makes use of array objects and contextual array rules, is introduced and its generative power for the description of picture arrays is examined. A main result of the paper is that there is a proper infinite hierarchy with respect to the classes of languages described by contextual array P systems. Such a hierarchy holds as well in the case when the selector is also endowed with the #−sensing ability.

Keywords two-dimensional arrays, array grammars, contextual array productions, array P systems

1 Introduction

Theoretical array grammar models [17,19,27] for the generation of digitized pictures are known to constitute formal syntactic methods for dealing with problems such as character recognition, cluster analysis of patterns, and other applications [5,26,24]. Several kinds of array grammar models (for example, see [9,17,

H. Fernau
FB 4 – Abteilung Informatikwissenschaften, Universität Trier, D-54296 Trier, Germany

R. Freund
Technische Universität Wien, Institut für Computersprachen, A-1040 Wien, Austria

M. L. Schmid
FB 4 – Abteilung Informatikwissenschaften, Universität Trier, D-54296 Trier, Germany

K. G. Subramanian
School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia
Tel.: +604 653 4641
Fax: +604 657 3335
E-mail: kgsmani1948@yahoo.com

P. Wiederhold
Depto. de Control Automático, Centro de Investigación y de Estudios Avanzados (CINVESTAV-IPN), Av. I.P.N. 2508, Col. San Pedro Zacatenco, México 07000 D.F., México

19,23,27] and references therein) have been proposed for generation of picture arrays in the two-dimensional plane extending the techniques of string generation in formal string language theory [20,21]. The contextual array grammars introduced and investigated by Freund et al. [8] are an interesting array counterpart of the string contextual grammars introduced by Marcus [13], and they have intensively been investigated subsequently (for example, see [4,15]). In the string case, contextual grammars are motivated from certain fundamental linguistic phenomena [15]. In contrast to Chomsky grammars [20] or Lindenmayer systems [20], in contextual grammars symbols are never modified (rewritten). Instead, strings of symbols are adjoined to the current string and the symbols once introduced remain in the finally generated string. In contextual array grammars [8], starting from an axiom array, a production of the form (s, c) where both s and c are finite patterns, can be applied if in the current array we can identify a subarray identical with s and if the places corresponding to c are empty. Then in the current array, c can be adjoined to s, thus producing a new array. In the generated language we may retain either all the arrays produced in this way or only the arrays to which no production is applicable (which corresponds to maximal derivations).

In the area of membrane computing, a computability model now known as P systems was introduced by Gheorghe Păun [14,16], inspired by the structure and the functioning of the living cells. P systems have turned out to be a rich framework for dealing with several types of computing related problems. Among different kinds of P systems, rewriting P systems, in which objects are given as finite strings over an alphabet and the evolution rules are given as context-free rewriting rules, have been investigated extensively [1,6,14,28]. The contextual way of handling string objects in P systems has been considered in [12] and contextual P systems are found to be more powerful than string contextual grammars. Extending string rewriting P systems to arrays, Ceterchi et al. [2] introduced and investigated array P systems using a context-free type of isometric array productions. Subsequently, several P system models for array generation, using either isometric or non-isometric array productions, have been considered in the literature (for example, see [22]).

In Section 3 of this paper, we first consider contextual array grammars with the restriction that both the selector and the context in a contextual array production are connected but do not have empty cells labeled by the blank symbol #. The family of picture languages generated by contextual array grammars in the maximal mode, also called t-mode, is denoted by $\mathcal{L}(cont, t)$. It is natural to introduce and examine the power of using a contextual type of rules in array P systems. Based on the contextual style of array generation considered in Section 3, in Section 4 we introduce a P system model with array objects and array contextual rules of the kind considered in Section 3, and we call this a *contextual array P system*. The family of all picture languages generated by such a system with at most m membranes is denoted by $AP_m(cont)$. We then illustrate this model by generating, in the maximal mode, a picture language called "stars with balanced arms",within both a linear and a non-linear membrane structure. In Section 5, we prove that $\mathcal{L}(cont, t) \subset AP_2(cont)$, correcting an error in the proof of this result in [25] in establishing the proper inclusion, and we also prove that $AP_2(cont) \subset AP_3(cont)$. This leads to the interesting main result of the paper on the proper infinite hierarchy of language classes described by contextual array P systems considered in Section 3. In fact, we prove that $AP_k(cont) \subset AP_{3k}(cont)$,

by considering the picture language of "combs with teeth". We conclude this work by pointing out how the gap of the hierarchy can be reduced from $3k$ to $2k$ and by observing that a corresponding infinite hierarchy can be obtained as well when we allow the $\#-$sensing feature in the contextual array rules in the sense that the selector can contain the blank symbol #, which was not allowed in the definition given in Section 3.

2 Preliminaries

We briefly recall basic notions for arrays and array contextual grammars [8]. For notions related to formal language theory we refer to [20,21], and for array grammars and two-dimensional languages we refer to [8,9].

A point in the two-dimensional digital plane $\mathbb{Z}^2$, where $\mathbb{Z}$ is the set of integers, is called a pixel. Each pixel p is identified with the unit closed square whose center is p and so, it is common to also name a pixel a square or a cell. Two pixels $p = (p_1, p_2),\ q = (q_1, q_2) \in \mathbb{Z}^2$ are called neighbors (more precisely, 8-neighbors [18]) if $\| p - q \|_{max} = \max\{|p_1 - q_1|, |p_2 - q_2|\} = 1$. The neighborhood relation defines an undirected graph on $\mathbb{Z}^2$. A subset $M \subseteq \mathbb{Z}^2$ is called connected if for any two points $p, q \in M$, there is a sequence (a path in the graph) $p = m_1, m_2, \cdots, m_k = q$ of points of M where any two consecutive points are neighbors. Interpreting the digital plane as set of squares, two squares p, q are neighbors if they intersect. The connectivity used here coincides with the well-known 8-connectivity on the two-dimensional digital plane $\mathbb{Z}^2$ [18], and with the 1-connectivity on $\mathbb{Z}^2$ considered in [8].

Now let V be a finite alphabet, and # a symbol not belonging to V. A two-dimensional array is a set of pixels labeled by the symbols of V or the so-called *blank symbol* #, where # indicates that the pixel is empty or unlabeled. Formally, a *two-dimensional array or a picture* is a function $\alpha : \mathbb{Z}^2 \to V \cup \{\#\}$ with finite and connected support $supp(\alpha)$, which is defined by $supp(\alpha) = \{v \in \mathbb{Z}^2 \mid \alpha(v) \neq \#\}$. Although an array could be given as a function defined on a proper (finite) subset of $\mathbb{Z}^2$, it can be extended to an array defined on the whole plane $\mathbb{Z}^2$, assigning the blank symbol # to any pixel not labeled by any symbol of V. For specifying an array, it is therefore sufficient to specify some finite (super)set (of) $supp(\alpha)$ and to give the label (element of V) which the function α assigns to each pixel $p \in supp(\alpha)$. The restriction of an array to its support, which is a connected finite set of pixels labeled by elements of V, is named a *picture or a two-dimensional connected finite array.*

By V^{+2} we denote the set of all non-empty connected finite picture arrays over V; the empty array is considered as the function which assigns the blank symbol to all pixels of $\mathbb{Z}^2$. An *array language* or a *picture language* is a subset of V^{+2}. For example, Fig. 1 shows an array (which is a pattern) $\mathcal{L}$ describing the letter L that has pixels labeled by a. Assuming that the pixel having label a in the corner of the letter L has coordinates $(0,0) \in \mathbb{Z}^2$, the array in Fig. 1 can be described in a formal manner by giving the label $\alpha(p) \in V = \{\mathtt{a}\}$ to each pixel p belonging to the picture as follows:

$$\mathcal{L} = \{((0,0),\mathtt{a}), ((1,0),\mathtt{a}), ((2,0),\mathtt{a}), ((3,0),\mathtt{a}), ((4,0),\mathtt{a}), ((5,0),\mathtt{a}),$$
$$((6,0),\mathtt{a}), ((7,0),\mathtt{a}), ((0,1),\mathtt{a}), ((0,2),\mathtt{a}), ((0,3),\mathtt{a}), ((0,4),\mathtt{a}), ((0,5),\mathtt{a})\}$$

```
a
a
a
a
a
a a a a a a a a
```

Fig. 1: An array representing the letter L

For any $v \in \mathbb{Z}^2$, the translation $\tau_v : \mathbb{Z}^2 \to \mathbb{Z}^2$, given by $\tau_v(w) = w + v$ for all $v \in \mathbb{Z}^2$, provides a bijection between $supp(\alpha)$ and $\tau_v(supp(\alpha))$ for any array α. Defining $\alpha'(\tau_v(p)) = \alpha(p)$ for any $p \in supp(\alpha)$, we obtain an array α' which is the translation of the array α. On the set of all arrays, the binary relation defined for any two arrays by the fact that one array is a translation of the other one, clearly is an equivalence relation. Arrays therefore can be regarded as equivalence classes of arrays with respect to translations on $\mathbb{Z}^2$. Hence, only relative positions of the symbols different from the blank symbol are essential for describing an array. In order to keep the description simple, we refer to a member of such an equivalence class as the array itself. We will use the usual pictorial method to denote an array by a figure indicating only the non-blank labels of the pixels belonging to its support, but without mentioning the coordinates of the pixels themselves. For example, the array in Fig. 1 is shown in this manner.

Given two arrays α, β, array β is called a *sub-array* of α if there exists a vector $v \in \mathbb{Z}^2$ such that for the translation τ_v we have that $\tau_v(supp(\beta)) \subseteq supp(\alpha)$ as well as $\beta(\tau_v(x)) = \alpha(x)$ for all $x \in supp(\alpha)$. In other words, all labeled pixels of β coincide with the corresponding labeled pixels of α when β is placed on α after a suitable translation of the pattern β.

The *diameter* of a (finite) set of (2-dimensional) vectors U is

$$\mathrm{diam}(U) := \max\{\parallel u - v \parallel \mid u, v \in U\}.$$

The diameter of an array W, denoted by $\mathrm{diam}(W)$, is the diameter of the set of underlying position vectors.

3 Contextual Array Grammars

We now recall the definition of a contextual array grammar [8], in which we restrict ourselves to the two-dimensional case and with the "selector" and the "context" being connected and labeled only by symbols from an alphabet V and not by the blank symbol #, which means the selector and the context do not have empty pixels.

Definition 1 A contextual array grammar (CAG for short) is a construct $G = (V, P, A)$ where V is an alphabet, A is a finite set of two-dimensional arrays in V^{+2} called *axioms*, and P is a finite set of rules of the form (α, β), called *contextual array productions* where

(i) α is a function defined on $U_\alpha \subset \mathbb{Z}^2$ with values in V;
(ii) β is a function defined on $U_\beta \subset \mathbb{Z}^2$ with values in V;

(iii) $U_\alpha \cap U_\beta = \emptyset$ and U_α, U_β are finite.

(U_α, α) is called the *selector* and (U_β, β) the *context* of the production (α, β); U_α is called the *selector area*, and U_β is the *context area*.

For arrays $\mathcal{C}_1, \mathcal{C}_2 \in V^{+2}$, intuitively, if in $\mathcal{C}_1$ we find a sub-array that corresponds to the selector (U_α, α), and if the places corresponding to (U_β, β) are labeled only by the blank symbol #, then we can add the context (U_β, β), thus deriving $\mathcal{C}_2$. Formally, $\mathcal{C}_2$ is called directly derivable from $\mathcal{C}_1$ by the contextual array production $p = (\alpha, \beta) \in P$ (we write $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$), if there exists a vector $v \in \mathbb{Z}^2$ such that, again denoting by τ_v the translation by v, the following is true:

- $\mathcal{C}_1(w) = \mathcal{C}_2(w) = \alpha(\tau_{-v}(w))$ for all $w \in \tau_v(U_\alpha)$,
- $\mathcal{C}_1(w) = \#$ for all $w \in \tau_v(U_\beta)$,
- $\mathcal{C}_2(w) = \beta(\tau_{-v}(w))$ for all $w \in \tau_v(U_\beta)$,
- $\mathcal{C}_1(w) = \mathcal{C}_2(w)$ for all $w \in \mathbb{Z}^2 \setminus \tau_v(U_\alpha \cup U_\beta)$.

If there exists a contextual array production $p \in P$ such that $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$, then $\mathcal{C}_2$ is called derivable from $\mathcal{C}_1$ and we write $\mathcal{C}_1 \Longrightarrow_G \mathcal{C}_2$. By $\Longrightarrow_G^*$ we denote the reflexive transitive closure of $\Longrightarrow_G$ and by $\Longrightarrow_G^t$ we denote the relation which, for arbitrary arrays $\mathcal{A}, \mathcal{B} \in V^{+2}$, is defined by $\mathcal{A} \Longrightarrow_G^t \mathcal{B}$ if and only if $\mathcal{A} \Longrightarrow_G^* \mathcal{B}$ and there is no $\mathcal{C} \in V^{+2}$ such that $\mathcal{B} \Longrightarrow_G \mathcal{C}$.

The picture array language generated by a CAG G in the t-mode is defined as follows:

$$L_t(G) = \left\{ \mathcal{B} \in V^{+2} \mid \mathcal{A} \Longrightarrow_G^t \mathcal{B} \text{ for some } \mathcal{A} \in A \right\}.$$

For a given CAG grammar G, the relation $\Longrightarrow_G^t$ corresponds to collecting only the arrays produced by blocked derivations, namely, derivations which cannot be continued. This derivation mode is known as the maximal mode or *t-mode* ("termination-mode") [3]. On the other hand, in the $*$*-mode* of derivation, all pictures derivable from an axiom are taken in the picture language generated by G. In this paper, we mainly consider the t-mode of derivation. The family of picture languages generated by contextual array grammars of the form $G = (V, P, A)$ in the t-mode will be denoted by $\mathcal{L}(cont, t)$.

The diameter of a contextual array rule p is the diameter of the union of the sets of position vectors in the selector and in the context. Alternatively, one can say that the diameter of p is the diameter of the array obtained by the union of the selector and the context arrays. The diameter of a (finite) set of arrays is the maximum of the diameters of the arrays contained in the set. The diameter of a (finite) set of contextual array rules is the maximum of the diameters of the contextual array rules contained in the set. The diameter of a contextual array grammar is the diameter of its rule set. For all the different versions of the notion of a diameter, we use the denotation diam when necessary.

We illustrate the definitions given above by an example contextual array grammar working in t-mode which generates the array language L_c consisting of all pictures of solid squares of odd side lengths $2n+1, n \geq 1$, with the label c in its "central" pixel (the pixel in row $n+1$ and column $n+1$) and all other pixels having label a. Fig. 2 shows such a 5×5 solid square with c in its central pixel.

Example 1 Let $G = (\{\mathtt{a}, \mathtt{c}\}, P, A)$ be a contextual array grammar with A containing two axiom picture arrays A_1, A_2, which are given in pictorial form in the

```
a a a a a
a a a a a
a a c a a
a a a a a
a a a a a
```

Fig. 2: A 5×5 solid square with c in its central position

following figure without mentioning the coordinates of the pixels:

$$A_1 := \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix}, \quad A_2 := \begin{matrix} \mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix}$$

Since the selector area U_α and the context area U_β are disjoint in a contextual array production, the rules can be represented by the following patterns where the pixels and symbols of the selector are indicated by enclosing them in boxes. In that way, the set of rules $P := \{p_1, p_2, \ldots, p_8\}$ is defined by

$$p_1 := \begin{matrix} \boxed{\mathtt{a}}\ \mathtt{a} \\ \boxed{\mathtt{a}}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \end{matrix}, \quad p_2 := \begin{matrix} \boxed{\mathtt{a}}\ \mathtt{a}\ \mathtt{a} \\ \boxed{\mathtt{a}}\,\boxed{\mathtt{a}}\,\mathtt{a} \end{matrix}, \quad p_3 := \begin{matrix} \boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \\ \boxed{\mathtt{a}}\ \mathtt{a} \\ \boxed{\mathtt{a}} \end{matrix}, \quad p_4 := \begin{matrix} \boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \\ \boxed{\mathtt{a}}\ \mathtt{a} \\ \mathtt{a}\ \ \mathtt{a} \end{matrix},$$

$$p_5 := \begin{matrix} \boxed{\mathtt{a}}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \\ \mathtt{a}\ \ \boxed{\mathtt{a}} \end{matrix}, \quad p_6 := \begin{matrix} \mathtt{a}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \\ \mathtt{a}\ \mathtt{a}\ \ \boxed{\mathtt{a}} \end{matrix}, \quad p_7 := \begin{matrix} \mathtt{a} \\ \mathtt{a}\ \ \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \end{matrix}, \quad p_8 := \begin{matrix} \mathtt{a}\ \ \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}}\,\boxed{\mathtt{a}} \end{matrix}.$$

A maximal (that is, t-mode) derivation in G generating a 4×4 picture array of the language L_c is shown below:

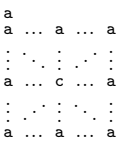
$$\begin{matrix} \mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_1} \begin{matrix} \mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_2} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_3} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_4} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a} \end{matrix}$$

$$\Longrightarrow_{p_5} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_6} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{c}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_8} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{c}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix} \Longrightarrow_{p_7} \begin{matrix} \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{c}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \\ \mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a} \end{matrix}.$$

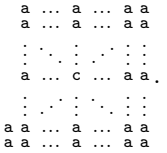
Lemma 1 *The contextual array grammar G of Example 1 describes the set of all squares over* a *with an odd number of rows and columns and with the symbol* c *in the central position.*

Proof The 3×3 axiom array A_1 belongs to the language L_c as no contextual array production can be applied to this picture. Assume that in a derivation of G we

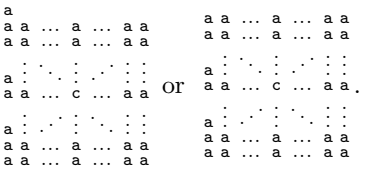
have constructed an array of the form



Then we have to apply production p_1 several times, production p_2 once, production p_3 several times, production p_4 once, production p_5 several times and then production p_6 once, which yields the following array



Now both productions p_7 and p_8 are applicable. However, no matter in which order we now apply these two productions, we will necessarily construct either



In the first case this whole procedure can be repeated, and in the second case no production is applicable any more and we terminate. Obviously, every member of L_c can be constructed in this way. □

Remark 1 The use of the language L_c of Example 1 to show the proper inclusion in Theorem 1 in [25], which is a preliminary version of this revised and enlarged version, is incorrect as L_c is generated by a contextual array grammar in the t-mode, although the statement of Theorem 1 in [25] itself holds as shown in the following Section 5.

4 Array P Systems with Contextual Array Productions

Among the different variants of P systems introduced in membrane computing, string rewriting P systems [14] which use context-free rules with target indications define string languages. Ceterchi et al. [2] introduced array P systems linking the areas of membrane computing and picture grammars and investigated its power in generating array languages. Array P systems are analogous to the string rewriting kind of P systems [14], but with picture array objects and array rewriting rules in the membrane regions of the system, with target indications *here*, *in*, *out*. It has an internal output in the sense that the result is obtained in a specified elementary membrane and has halting computations to define successful computations. We now introduce a P system that has similar features but with the difference of having contextual array productions in its membranes as in a contextual array grammar [8].

Definition 2 A *contextual array P system* with $m \geq 1$ membranes is a construct

$$\Pi = (V, \#, \mu, A_1, \ldots, A_m, P_1, \ldots, P_m, i_o),$$

where

- V is the alphabet;
- $\#$ is the blank symbol;
- μ is a membrane structure with m membranes or regions, labeled by $1, \ldots, m$, in a one-to-one manner;
- $A_1, \ldots, A_m$ are finite sets of arrays over V associated with the m regions of μ;
- $P_1, \ldots, P_m$ are finite sets of contextual array productions over V associated with the m regions of μ; the productions have attached targets *here*, *out*, *in*, and in_j, $1 \leq j \leq m$;
- i_o is the label of a membrane called *output membrane*, which serves for collecting the results of successful computations.

The membrane structure μ of a P system with m membranes can be denoted by a well-formed expression of parentheses over the alphabet of left and right parentheses $[_i$ and $]_i$, $1 \leq i \leq m$. For example, $[_1\, [_2\,]_2\, [_3\, [_4\,]_4\,]_3\,]_1$ means that membrane 1 is the outermost membrane, which contains membranes 2 and 3, whereas membrane 3 contains another membrane 4.

A *computation step* in a contextual array P system is done as follows: for each array $\mathcal{A}$ in each region of the system, if a contextual array production p in the region can be applied to $\mathcal{A}$, then it should be applied which means that the application of a rule is sequential at the level of arrays, but maximally parallel at the level of the whole system. If more than one rule is applicable at the same time, then one is chosen in a nondeterministic way. The resulting array, if any, is placed in the region indicated by the target associated with the rule having been applied with interpreting the attached target as follows: *here* means that the array remains in the same region, *out* means that the array exits the current membrane and is placed in the immediately outer membrane if one exists (in this paper, we do not allow the target *out* to be used by a rule assigned to the skin membrane), in_j means that the array is immediately sent to the directly inner membrane with label j, and *in* means that the array is immediately sent to one of the directly inner membranes, chosen in a nondeterministic way if several such membranes exist (if no inner membrane exists, then a rule with the target indication *in* cannot be used). A computation is called *successful* if and only if it halts, which means that a configuration has been reached where no rule can be applied to the existing arrays. The result of a halting computation consists of the arrays collected in the membrane with label i_o in the halting configuration. The set of all such arrays computed or *generated* by a system Π is denoted by $\mathrm{AL}(\Pi)$. The family of all picture array languages $\mathrm{CAL}(\Pi)$ generated by systems Π as defined above, with at most m membranes, is denoted by $\mathrm{AP}_m(cont)$. The diameter of a P system Π with contextual array productions, denoted by $\mathrm{diam}(\Pi)$, is the diameter of the union of the rule sets of Π.

We shall now illustrate the definition of contextual array P systems with some examples. To this end, let $L_{\mathrm{star},4}$ be the following set of stars over the alphabet

$\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ with 4 balanced arms of even length, i. e.:

$$L_{\mathrm{star},4} := \left\{ \begin{array}{c} \mathtt{c} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \end{array}, \begin{array}{c} \mathtt{c} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \end{array}, \begin{array}{c} \mathtt{c} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \\ \mathtt{b} \end{array}, \ldots \right\}$$

Each arm is a digital version of a ray starting at the centre point (and including that point). The resulting pixels are labeled alternately by $\mathtt{a}$ and $\mathtt{b}$ starting with the centre pixel labeled by $\mathtt{a}$. As an exception, the last pixel of the north arm gets label $\mathtt{c}$. The length of an arm is defined as the total number of appearances of all labels of its pixels. So, the length of the arms of the stars of $L_{\mathrm{star},4}$ are $4, 6, 8$ and so on.

We now define a contextual array P system with a linear membrane structure of 5 membranes which describes the language $L_{\mathrm{star},4}$. Let

$$\Pi_{\mathrm{star},4} = (\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, \#, \mu, A_1, \ldots, A_5, P_1, \ldots, P_5, 5)\,,$$

where $\mu = [_1\,[_2\,[_3\,[_4\,[_5\,]_5\,]_4\,]_3\,]_2\,]_1$. The sets of axioms are defined by

$$A_1 = \left\{ \begin{array}{c} \mathtt{a} \\ \mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a} \\ \mathtt{b} \\ \mathtt{a} \end{array} \right\} \text{ and } A_2 = A_3 = A_4 = A_5 = \emptyset\,,$$

and the rules are defined by

$$P_1 := \{p_{1,1}, p_{1,2}\} := \left\{ \left(\begin{array}{c} \mathtt{b} \\ \boxed{\mathtt{a}} \\ \mathtt{b} \end{array}, \text{ in} \right), \left(\begin{array}{c} \mathtt{c} \\ \boxed{\mathtt{a}} \\ \mathtt{b} \end{array}, \text{ in} \right) \right\},$$

$$P_2 := \{p_{2,1}, p_{2,2}\} := \left\{ \left(\boxed{\mathtt{b}}\,\boxed{\mathtt{a}}\,\mathtt{b}, \text{ in} \right), \left(\boxed{\mathtt{a}}\,\boxed{\mathtt{b}}\,\mathtt{a}, \text{ out} \right) \right\},$$

$$P_3 := \{p_{3,1}, p_{3,2}\} := \left\{ \left(\begin{array}{c} \boxed{\mathtt{b}} \\ \boxed{\mathtt{a}} \\ \mathtt{b} \end{array}, \text{ in} \right), \left(\begin{array}{c} \boxed{\mathtt{a}} \\ \boxed{\mathtt{b}} \\ \mathtt{a} \end{array}, \text{ out} \right) \right\},$$

$$P_4 := \{p_{4,1}, p_{4,2}\} := \left\{ \left(\mathtt{b}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{b}}, \text{ in} \right), \left(\mathtt{a}\,\boxed{\mathtt{b}}\,\boxed{\mathtt{a}}, \text{ out} \right) \right\},$$

$$P_5 := \{p_{5,1}\} := \left\{ \left(\begin{array}{c} \mathtt{a} \\ \boxed{\mathtt{b}} \\ \boxed{\mathtt{a}} \end{array}, \text{ out} \right) \right\}.$$

Intuitively, in a computation of the contextual array P system $\Pi_{\mathrm{star},4}$, the array moves back and forth between membrane 1 and membrane 5, without changing directions in between. On the way down from membrane 1 to membrane 5, every arm is extended by the symbol $\mathtt{b}$ and on the way back moving up every arm is extended by the symbol $\mathtt{a}$. The purpose of the special symbol $\mathtt{c}$ is to stop the whole computation.

Lemma 2 $\mathrm{AL}(\Pi_{star,4}) = L_{star,4}$.

Proof It can easily be seen that every computation starts with either $p_{1,1}$ or $p_{1,2}$ and then $p_{2,1}, p_{3,1}, p_{4,1}$ necessarily have to follow, and after having applied these rules, membrane 5 either contains the array

$$U := \begin{array}{c} \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{b a b a b a b} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array} \text{ or the array } U' := \begin{array}{c} \texttt{c} \\ \texttt{a} \\ \texttt{b} \\ \texttt{b a b a b a b} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array},$$

depending on whether initially $p_{1,1}$ or $p_{1,2}$ has been used. If U' has been obtained in this way, then the computation stops and $U' \in L_{\text{star},4}$. If, on the other hand, membrane 5 contains U, then the rules $p_{5,1}, p_{4,2}, p_{3,2}, p_{2,2}$ must be applied which move the array back to membrane 1 and change it to

$$U'' := \begin{array}{c} \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a b a b a b a b a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \end{array}.$$

Now, the same arguments apply again, which means that either the array

$$W := \begin{array}{c} \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{b a b a b a b a b a b a b} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array} \text{ or the array } W' := \begin{array}{c} \texttt{c} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{b a b a b a b a b a b a b} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array}$$

appears in membrane 5 and again $W' \in L_{\text{star},4}$. Thus, inductively applying these arguments, we have proved that $\Pi_{\text{star},4}$ can only produce arrays of $L_{\text{star},4}$ and, obviously, all arrays of $L_{\text{star},4}$ can be produced in this way, which in sum implies $L_{\text{star},4} = \text{AL}(\Pi_{\text{star},4})$. □

We note that it is possible, for every $k \geq 1$, to generalize the language $L_{\text{star},4}$ to a language $L_{\text{star},k}$ which contains stars like those in $L_{\text{star},4}$, but with k arms instead of 4 arms. There are several ways of how this can be done; in the following, we sketch one possibility. We wish to point out that the following constructions rely on the digitization of rays, which is a well-known procedure (for example, see Chapter 9 of [11]). In order to describe stars with 8 instead of 4 arms, we could simply add a north-west, a north-east, a south-west and a south-east arm, i. e.,

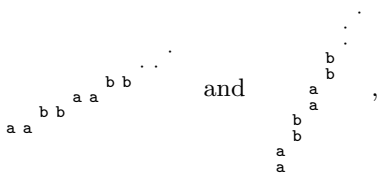
$$L_{\text{star},8} = \left\{ \begin{array}{c} \texttt{b} \quad \texttt{c} \quad \texttt{b} \\ \texttt{a} \quad \texttt{a} \quad \texttt{a} \\ \texttt{b} \quad \texttt{c} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \\ \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \\ \texttt{b b b} \quad \texttt{b b b} \quad \texttt{b b b} \\ \texttt{b a b a b a b}, \texttt{b a b a b a b a b a b}, \texttt{b a b a b a b a b a b a b}, \ldots \\ \texttt{b b b} \quad \texttt{b b b} \quad \texttt{b b b} \\ \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \\ \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \\ \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \quad \texttt{a} \\ \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \quad \texttt{b} \\ \texttt{a} \quad \texttt{a} \quad \texttt{a} \\ \texttt{b} \quad \texttt{b} \quad \texttt{b} \end{array} \right\}.$$

Thus, in order to obtain a star with 8 arms, we extend a star with 4 arms by adding a new arm with a gradient of 1 to the north-eastern area of the star and then we copy it by mirroring it on the vertical and horizontal axis of symmetry of

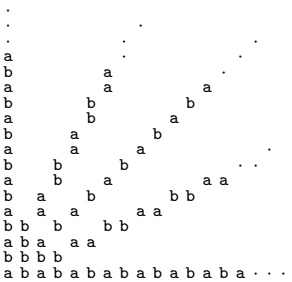
the star. Each of the four new arms is a diagonal pixel path which is the digitization of a ray of gradient 1 or -1 starting at the centre point of the star. The labeling of the pixels of the diagonal arms alternately by a and b starting at the centre pixel can be done easily since the centre pixel already has the label a and any two of the 8 arms intersect only at this centre.

The general construction of (labeled) stars with k arms, where k is a power of 2, follows the idea of iteration of above: each star of $k = 2^l$ arms of length n is constructed from a star having $k = 2^{l-1}$ arms of length n. Also the idea of mirroring all new arms constructed in the first quadrant on the vertical and horizontal axis of symmetry of the star can be applied in general. So, we need only to explain how to construct the first quadrant of a star.

In order to construct a star with 16 arms from a star with 8 arms, in the first quadrant we have to add 2 arms, corresponding to the ray with gradient $\frac{1}{2}$ and the ray with the reciprocal gradient 2. Similarly, for obtaining a star of $32 = 2^5$ arms, in the first quadrant we add $4 = 2^2$ new rays with gradients $\frac{1}{4}$, $\frac{3}{4}$ and (the reciprocals) $\frac{4}{3}$, 4. In the next iteration, for constructing a star of $64 = 2^6$ arms, in the first quadrant we add $8 = 2^3$ new rays with gradients $\frac{1}{8}$, $\frac{3}{8}$, $\frac{5}{8}$, $\frac{7}{8}$ and (the reciprocals) $\frac{8}{7}$, $\frac{8}{5}$, $\frac{8}{3}$, 8. For example, in order to construct stars with 16 arms, the arm with a gradient of $\frac{1}{2}$ and the arm with a gradient of 2 could look like

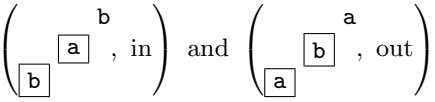


which results in stars with first quadrants of the following form:



The stars of a language $L_{\mathrm{star},k}$, where k is not a power of two, can be constructed like the stars of $L_{\mathrm{star},2^l}$, $l := \lceil \log(k) \rceil$, with the only difference, that we simply produce only k of the l possible arms and omit the other $2^l - k$ arms.

Now in order to define a P system that describes the language $\Pi_{\mathrm{star},k}$, $k \geq 1$, we can modify $\Pi_{\mathrm{star},4}$ in a straightforward way. Instead of a linear structure of 5 membranes we use a linear structure of $k + 1$ membranes, where every inner membrane is responsible for one of the arms just as it is the case for $\Pi_{\mathrm{star},4}$. For example, if membrane i is responsible for the arm with gradient 1, then we have to add the rules



to P_i. In a similar way, for the arms with gradient $\frac{1}{2}$ and 2, we need the rules

$$\left(\begin{matrix} & \mathtt{b}\,\mathtt{b} \\ \boxed{\mathtt{a}}\,\boxed{\mathtt{a}} & \end{matrix}, \text{ in}\right), \left(\begin{matrix} & \mathtt{a}\,\mathtt{a} \\ \boxed{\mathtt{b}}\,\boxed{\mathtt{b}} & \end{matrix}, \text{ out}\right) \text{ and}$$

$$\left(\begin{matrix} \mathtt{b} \\ \mathtt{b} \\ \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}} \end{matrix}, \text{ in}\right), \left(\begin{matrix} \mathtt{a} \\ \mathtt{a} \\ \boxed{\mathtt{b}} \\ \boxed{\mathtt{b}} \end{matrix}, \text{ out}\right), \text{ respectively.}$$

We wish to point out here that the diameter of the P systems $\Pi_{\mathrm{star},k}$, $k \geq 1$, grows with the number of arms. Intuitively, this is due to the fact that the rules of $\Pi_{\mathrm{star},k}$ distinguish the arms by their gradient and, since the arms are represented in a discrete way, a growing number of arms require larger and larger portions of the arms to be observed in order to determine their gradient.

In the example of $\Pi_{\mathrm{star},4}$ elaborated above, we use the linear structure of the P system in order to implement two phases of the computation, which are repeated in an alternate manner. In the first phase, we move down the membrane structure and extend every arm by exactly one $\mathtt{b}$, and in the second phase we move up the membrane structure and extend every arm by exactly one $\mathtt{a}$ and so on. An alternative approach is to first extend the first arm by $\mathtt{ba}$, then extend the second arm by $\mathtt{ba}$ and so on until all arms are extended by these two symbols and then this whole procedure starts over again. This idea is implemented in the following P system with a non-linear membrane structure.

Let

$$\Pi'_{\mathrm{star},4} = (\{\mathtt{a},\mathtt{b},\mathtt{c}\}, \#, \mu, A_1, \ldots, A_5, P_1, \ldots, P_5, 5)\,,$$

where $\mu = [_1\,[_2\,]_2\,[_3\,]_3\,[_4\,]_4\,[_5\,]_5\,]_1$. The sets of axioms are defined by

$$A_1 = \left\{ \begin{matrix} & \mathtt{a} & \\ \mathtt{b} & \mathtt{b} & \mathtt{b} \\ & \mathtt{a} & \end{matrix} \right\} \text{ and } A_2 = A_3 = A_4 = A_5 = \emptyset\,,$$

and the rules are defined by

$$\begin{aligned}
P_1 &:= \{p_{1,1}, p_{1,2}, \ldots, p_{1,5}\} \\
&:= \left\{ \left(\begin{matrix} \mathtt{b} \\ \boxed{\mathtt{a}} \\ \boxed{\mathtt{b}} \end{matrix}, \text{ in}_2 \right), \left(\begin{matrix} \mathtt{c} \\ \boxed{\mathtt{a}} \\ \boxed{\mathtt{b}} \end{matrix}, \text{ in}_2 \right), \left(\boxed{\mathtt{b}}\,\boxed{\mathtt{a}}\,\mathtt{b}, \text{ in}_3\right), \right. \\
&\qquad \left. \left(\begin{matrix} \boxed{\mathtt{b}} \\ \boxed{\mathtt{a}} \\ \mathtt{b} \end{matrix}, \text{ in}_4 \right), \left(\mathtt{b}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{b}}, \text{ in}_5\right) \right\}, \\
P_2 &:= \{p_2\} := \left\{ \left(\boxed{\mathtt{a}}\,\boxed{\mathtt{b}}\,\mathtt{a}, \text{ out}\right) \right\}, \\
P_3 &:= \{p_3\} := \left\{ \left(\begin{matrix} \boxed{\mathtt{a}} \\ \boxed{\mathtt{b}} \\ \mathtt{a} \end{matrix}, \text{ out} \right) \right\}, \\
P_4 &:= \{p_4\} := \left\{ \left(\mathtt{a}\,\boxed{\mathtt{b}}\,\boxed{\mathtt{a}}, \text{ out}\right) \right\}, \\
P_5 &:= \{p_5\} := \left\{ \left(\begin{matrix} \mathtt{a} \\ \boxed{\mathtt{b}} \\ \boxed{\mathtt{a}} \end{matrix}, \text{ out} \right) \right\}.
\end{aligned}$$

In a computation of $\Pi'_{\text{star},4}$, we successively move from membrane 1 to the inner membranes and back again. When we move from membrane 1 to membrane i, we extend arm $i-1$ (which, in the previous step, has been extended by symbol a) by the symbol b, and when we move back again to membrane 1, we extend the next arm, i. e., arm i, by symbol a.

Lemma 3 $\text{AL}(\Pi'_{star,4}) = L_{star,4}$.

Proof It can easily be seen that every computation starts with either $p_{1,1}$ or $p_{1,2}$ and then $p_2, p_{1,3}, p_3, p_{1,4}, p_4, p_{1,5}$ necessarily follow and after applying these rules, membrane 5 either contains the array

$$U := \begin{array}{c} \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{babababab} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array} \quad \text{or the array} \quad U' := \begin{array}{c} \texttt{c} \\ \texttt{a} \\ \texttt{b} \\ \texttt{babababab} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array},$$

depending on whether initially $p_{1,1}$ or $p_{1,2}$ has been used. If U' has been obtained in this way, then the computation halts and $U' \in L_{\text{star},4}$. If, on the other hand, membrane 5 contains U, then rule p_5 must be applied which moves the array back to membrane 1 and changes it to

$$U'' := \begin{array}{c} \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{babababab} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array}.$$

Now, the same argument applies again, which means that either the array

$$W := \begin{array}{c} \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{bababababababab} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array} \quad \text{or the array} \quad W' := \begin{array}{c} \texttt{c} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{bababababababab} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \\ \texttt{a} \\ \texttt{b} \end{array},$$

appears in membrane 5 and again $W' \in L_{\text{star},4}$. Thus, $\Pi_{\text{star},4}$ can only produce arrays of $L_{\text{star},4}$ and, obviously, all arrays of $L_{\text{star},4}$ can be produced in this way, which implies $L_{\text{star},4} = \text{AL}(\Pi'_{\text{star},4})$. □

Obviously, the P system $\Pi'_{\text{star},4}$ can also be extended to P systems $\Pi'_{\text{star},k}$, $k \geq 1$, which generate the languages $L_{\text{star},k}$.

We wish to emphasize that in the example P systems defined above, we use a number of k membranes (ignoring the skin membrane, which we always have) in order to synchronize the growth of k arms of the stars. It seems intuitively clear that such a synchronization of k independent parts of the arrays is only possible if we have enough membranes and it seems unlikely that the languages $L_{\text{stars},k}$ can be described by a P system with less than k membranes (in addition to the skin membrane). We shall formally prove this intuition in the next section.

5 A Proper Hierarchy of Language Classes Described by Contextual Array P Systems

In this section, we prove that the expressive power of contextual array P systems strictly increases with the number of membranes, i. e., we prove a proper hierarchy with respect to the classes of languages described by contextual array P systems. Before we present this infinite hierarchy, we first compare the expressive power of contextual array grammars in the maximal mode with the expressive power of contextual array P systems with only 2 membranes. It turns out that the latter device is strictly more powerful, which shows that the contextual way of handling array objects in P systems increases the generative power of the contextual way of rewriting arrays in the t-mode. After this, we compare the class of languages generated by contextual array P systems with 2 membranes and the class of languages generated by contextual array P systems with 3 membranes.

5.1 Contextual array P systems with few membranes

Theorem 1 $\mathcal{L}(cont, t) \subset \mathrm{AP}_2(cont)$.

Proof The inclusion can be seen as follows: Let L be an array language in $\mathcal{L}(cont, t)$ generated by a CAG G in the maximal mode. We construct a contextual array P system Π with only one membrane which is also the output membrane, containing all the rules of G, each with attached target *here*, and the same sets of axioms as G. It is clear that Π generates exactly the arrays of L.

The proper inclusion is seen by considering the picture language L_l consisting of picture arrays describing the shape L with each pixel labeled by `a` except for the pixel in the uppermost position of the vertical arm and both the arms having equal length (which here means equal number of pixels) of at least three. A member of L_l is shown in Fig. 3.

```
b
a
a
a
a a a a a
```

Fig. 3: A picture describing the shape L.

We prove that the language L_l cannot be generated by any CAG in the maximal mode. To this end, we assume that there exists a CAG G, such that $L_t(G) = L_l$. Next, we note that since G operates in the maximal mode, it cannot contain any rule p which is applicable to an element $w \in L_l$. This is due to the fact that if such a rule p exists, then w cannot be the result of a maximal derivation, since p is applicable to w; thus, w cannot be derived with respect to G. This directly implies that G does not contain a rule of form $\boxed{\mathtt{a}}\boxed{\mathtt{a}}\cdots\boxed{\mathtt{a}}\,\mathtt{a}\,\mathtt{a}\cdots\mathtt{a}$ or $\mathtt{a}\,\mathtt{a}\cdots\mathtt{a}\boxed{\mathtt{a}}\boxed{\mathtt{a}}\cdots\boxed{\mathtt{a}}$. Obviously, without such rules, not all elements of L_l can be generated by G in the t-mode. This concludes the proof that the language L_l cannot be generated by any CAG in the maximal mode.

On the other hand, the following contextual array P system Π_l with 2 membranes can generate L_l. Let $\Pi_l = (\{\mathtt{a}, \mathtt{b}\}, \#, \mu, A_1, A_2, P_1, P_2, 2)$ where $\mu = [_1 [_2]_2]_1$ and $A_1 := \{ \begin{smallmatrix} \mathtt{a} \\ \mathtt{a}\ \mathtt{a} \end{smallmatrix} \}$, $A_2 := \emptyset$. The rules are defined as follows:

$$P_1 := \{p_1\} := \left\{ \left(\boxed{\mathtt{a}}\, \boxed{\mathtt{a}}\, \mathtt{a}, in \right) \right\},$$

$$P_2 := \{p_{2,1}, p_{2,2}\} := \left\{ \left(\begin{smallmatrix} \mathtt{a} \\ \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}} \end{smallmatrix}, out \right), \left(\begin{smallmatrix} \mathtt{b} \\ \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}} \end{smallmatrix}, here \right) \right\}.$$

Starting from the axiom array $\begin{smallmatrix} \mathtt{a} \\ \mathtt{a}\ \mathtt{a} \end{smallmatrix}$ in membrane 1, the rule p_1 is applied adjoining the context $\mathtt{a}$ to the selector $\mathtt{a}\,\mathtt{a}$ and then the resulting array $\begin{smallmatrix} \mathtt{a} \\ \mathtt{a}\ \mathtt{a}\ \mathtt{a} \end{smallmatrix}$ is sent to membrane 2 due to the target indication *in* in p_1. Note that there is no initial array in membrane 2. If the rule $p_{2,1}$ is applied in membrane 2, then the context $\mathtt{a}$ will be adjoined resulting in the array $\begin{smallmatrix} \mathtt{a} \\ \mathtt{a} \\ \mathtt{a}\ \mathtt{a}\ \mathtt{a} \end{smallmatrix}$ which is sent back to membrane 1, due to the target indication *out* in rule $p_{2,1}$. Now the rule p_1 in membrane 1 can be applied and the resulting array is sent to membrane 2 again. The process can be repeated. If rule $p_{2,2}$ is applied in membrane 2, then the uppermost pixel is "filled" with $\mathtt{b}$ and the picture remains in membrane 2 due to the target indication *here*. Note that the number of symbols in both the horizontal and the vertical arms of the resulting L shaped picture will be the same so that the arms are of equal length (of at least three). Hence, only arrays from L_l can be produced and all arrays of L_l can be produced in this way as well. □

Next, we show that contextual array P systems with 3 membranes are more powerful than contextual array P systems with 2 membranes.

Theorem 2 $\mathrm{AP}_2(cont) \subset \mathrm{AP}_3(cont)$.

Proof The inclusion follows from the definition. The proper inclusion can be seen by considering the picture language L_t consisting of pictures in the shape of the letter `T` with the pixels in the horizontal arm labeled by `a` except the leftmost pixel which is labeled by `b` and the pixels in the vertical arm labeled by `a` and `b` alternately. A member of L_t is shown in Fig. 4. The number of symbols in the picture in the upper horizontal row is $2n + 3, n \geq 1$, while in the vertical column it is $2n + 1, n \geq 1$ (thereby counting the symbol at the intersection of both lines twice).

```
b a a a a a a
    b
    a
    b
    a
```

Fig. 4: A picture describing the shape T.

We prove that the language L_t cannot be generated by any contextual array P system with only 2 membranes. To this end, we assume that there exists a contextual array P system Π with a membrane structure of $[_1 [_2]_2]_1$. In the following, a rule of form $(\boxed{\mathtt{a}}\, \boxed{\mathtt{a}} \ldots \boxed{\mathtt{a}}\, \mathtt{a}\, \mathtt{a} \ldots \mathtt{a}, T)$, $T \in \{in, out, here\}$, is a *right extension rule* and a rule of form $(\mathtt{a}\, \mathtt{a} \ldots \mathtt{a}\, \boxed{\mathtt{a}}\, \boxed{\mathtt{a}} \ldots \boxed{\mathtt{a}}, T')$, $T' \in \{in, out, here\}$, is a *left extension rule*. It can easily be seen that in a computation of a sufficiently large

array of L_t, there is a final phase of the computation in which every applied rule extending the horizontal arm to the left or to the right must be a left or right extension rule, respectively. Moreover, we can assume that in this phase of the computation, at least one left or right extension rule is applied. We shall now modify such a computation in the following way. First, we carry out the computation up to the very last application of a left or right extension rule and, without loss of generality, we assume that this is a left extension rule p with target indication $T \in \{in, out, here\}$. If $T = here$, then we can simply repeat this rule and finish with the remainder of the original computation, which produces an array that is not a member of L_t. In the following, we assume that $T \in \{in, out\}$ and $p \in P_l$, where $l \in \{1, 2\}$, i. e., the rule p changes the array from membrane l to membrane l', where $l' \in \{1, 2\}$, $l \neq l'$. If in P_l there is a right extension rule q with the same target indication T, then we can use q instead of p and finish with the remainder of the original computation or, if q has target indication $here$, then we can use q once, before we apply p and then finish with the remainder of the original computation (note that q must be applicable, due to our assumption that the produced array is large enough and the application of p is the very last application of a left or right extension rule in the whole computation). In both cases, we produce an array that is not a member of L_t. Now if we want to rule out this possibility of a right extension rule in P_l, then we have to assume that there is a right extension rule in $P_{l'}$ with target indication $T' \in \{in, out, here\}$, since otherwise no right extension rule would exist. We can now proceed with the computation by applying rule p and therefore moving the array to membrane l'. If $T' = here$, then we can apply q once and then finish with the remainder of the original computation, which produces an array that is not a member of L_t. If, on the other hand, $T' \in \{in, out\}$, then we can apply q and move back to membrane l, where again p is applicable. Hence, we can enter a loop of alternately applying rules p and q and then proceeding with the original computation. In this loop, the horizontal arm may be equally grown to both the left and to the right, but the southbound arm is not synchronized; thus, an array is produced that is not a member of L_t. This shows that $L_t \notin \mathrm{AP}_2(cont)$.

Next, we show that there exists a contextual array P system Π_t with three membranes which generates L_t:

Let $\Pi_t = (\{a, b\}, \#, \mu, A_1, A_2, A_3, P_1, P_2, P_3, 3)$ where $\mu = [_1\,[_2\,[_3\,]_3\,]_2\,]_1$. The sets of axioms are defined by $A_1 := \{\begin{smallmatrix}\mathtt{a}\,\mathtt{a}\,\mathtt{a}\\ \mathtt{b}\end{smallmatrix}\}$ and $A_2 = A_3 = \emptyset$. The rules are as follows:

$$P_1 := \{p_1\} := \left\{\left(\boxed{\mathtt{a}}\,\boxed{\mathtt{a}}\,\mathtt{a}, in\right)\right\},$$

$$P_2 := \{p_{2,1}, p_{2,2}\} := \left\{\left(\begin{smallmatrix}\boxed{\mathtt{a}}\\ \boxed{\mathtt{b}}\\ \mathtt{a}\end{smallmatrix}, in\right), \left(\begin{smallmatrix}\boxed{\mathtt{b}}\\ \boxed{\mathtt{a}}\\ \mathtt{b}\end{smallmatrix}, out\right)\right\},$$

$$P_3 := \{p_{3,1}, p_{3,2}\} := \left\{\left(\mathtt{a}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{a}}, out\right), \left(\mathtt{b}\,\boxed{\mathtt{a}}\,\boxed{\mathtt{a}}, here\right)\right\}.$$

Starting from the axiom array $\begin{smallmatrix}\mathtt{a}\,\mathtt{a}\,\mathtt{a}\\ \mathtt{b}\end{smallmatrix}$ in membrane 1, the rule p_1 is applied adjoining the context $\mathtt{a}$ to the right of the selector $\mathtt{a}\,\mathtt{a}$ extending the right arm by one pixel and then the resulting array $\begin{smallmatrix}\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\\ \mathtt{b}\end{smallmatrix}$ is sent to membrane 2 due to the target indication in in p_1. If the rule $p_{2,1}$ is applied in membrane 2, then the context $\mathtt{a}$ will be adjoined extending the vertical arm by one pixel, and the resulting array $\begin{smallmatrix}\mathtt{a}\,\mathtt{a}\,\mathtt{a}\,\mathtt{a}\\ \mathtt{b}\\ \mathtt{a}\end{smallmatrix}$ is sent to membrane 3, due to the target indication in in rule $p_{2,1}$. If the rule

$p_{3,1}$ in membrane 3 is applied now, this extends the left arm by one pixel resulting in the array $\begin{smallmatrix} & & & & \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ & & & & & & \mathtt{b} & \\ & & & & & & \mathtt{a} & \end{smallmatrix}$, which is sent again to membrane 2. The only applicable rule now is $p_{2,2}$, which extends the vertical arm by one pixel with the resulting array $\begin{smallmatrix} \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ & & \mathtt{b} & \\ & & \mathtt{a} & \\ & & \mathtt{b} & \end{smallmatrix}$ being sent to membrane 1, and the process can be repeated. If rule $p_{3,2}$ is applied in membrane 2, then the leftmost pixel in the horizontal arm is filled with label `b` and the picture remains in output membrane 3 due to the target indication *here*. Obviously, this array is a member of L_t, and every array of L_t can be generated in such a way. This observation concludes the proof. □

5.2 Comb languages

We now define a class of array languages which shall be used in order to establish the infinite hierarchy of language classes given by contextual array P systems. The language of combs with 4 teeth (where the i^{th} tooth is defined over the alphabet $\{a_i\}$, $1 \leq i \leq 4$), denoted by $L_{\text{comb},4}$, is the following set:

$$\left\{ \begin{array}{ccccccccccccccccccccc}
 & & & & & & & & & & & & & & & & & & \mathtt{X} & & \\
 & & & & & & & & & & & & & & & & & & \mathtt{b}_3 & & \\
 & & & & & & & & & & \mathtt{X} & & & & & & & & \mathtt{b}_2 & & \\
 & & & & & & & & & & \mathtt{b}_3 & & & & & & & & \mathtt{b}_1 & & \\
 & & & & & & \mathtt{X} & & & & \mathtt{b}_2 & & & & & & & & \mathtt{b}_4 & & \\
 & & & & & & \mathtt{b}_3 & & & & \mathtt{b}_1 & & & & & & & & \mathtt{b}_3 & & \\
\mathtt{a}_1 & \mathtt{a}_2 & \mathtt{a}_3 & \mathtt{a}_4 & \mathtt{b}_2 & , & & & & & \mathtt{b}_4 & , & & & & & & & \mathtt{b}_2 & , \ \ldots . \\
\mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 & \mathtt{X} & \mathtt{a}_4 & \mathtt{X} & \mathtt{b}_1 & & \mathtt{b}_3 & & & & & & & & \mathtt{b}_1 & & \\
 & & & & & & & & & \mathtt{a}_1 & \mathtt{a}_2 & \mathtt{a}_3 & \mathtt{a}_4 & & & & & \mathtt{b}_4 & & \\
 & & & & & & & & & \mathtt{a}_1 & \mathtt{a}_2 & \mathtt{a}_3 & \mathtt{a}_4 & & & & & \mathtt{b}_2 & & \\
 & & & & & & & & & \mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 & \mathtt{X} & \mathtt{a}_4 & \mathtt{X} & \mathtt{b}_1 & &
\end{array} \right\}$$

In a similar way, for every $k \geq 1$, the language $L_{\text{comb},k}$ can be defined as containing combs like in $L_{\text{comb},4}$, but with k instead of 4 teeth, i. e., $L_{\text{comb},k}$ contains arrays of the form

$$\begin{array}{ccccccc}
 & & & & & & \mathtt{X} \\
 & & & & & & \mathtt{b}_{k-1} \\
 & & & & & & \vdots \\
 & & & & & & \mathtt{b}_1 \\
 & & & & & & \vdots \\
\mathtt{a}_1 & \mathtt{a}_2 & \mathtt{a}_3 & \ldots & \mathtt{a}_k & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \mathtt{b}_k \\
 & & & & & & \vdots \\
\mathtt{a}_1 & \mathtt{a}_2 & \mathtt{a}_3 & \ldots & \mathtt{a}_k & & \mathtt{b}_2 \\
\mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 \ \ldots \ \mathtt{a}_k & \mathtt{X} & \mathtt{b}_1
\end{array} .$$

We shall now define, for every $k \geq 1$, a P system with $k+1$ membranes that generates the language $L_{\text{comb},k}$. Let

$$\Pi_{\text{comb},k} = (\{\mathtt{a}_1, \ldots \mathtt{a}_k, \mathtt{b}_1, \ldots, \mathtt{b}_k, \mathtt{X}\}, \#, \mu, A_1, \ldots, A_{k+1}, P_1, \ldots, P_{k+1}, 1)$$

where $\mu = [_1 \, [_2 \,]_2 \, [_3 \,]_3 \, \ldots \, [_{k+1} \,]_{k+1} \,]_1$. The sets of axioms are defined by

$$A_1 = \{ \, \mathtt{a}_1 \ \mathtt{X} \ \mathtt{a}_2 \ \mathtt{X} \ \mathtt{a}_3 \ \ldots \ \mathtt{a}_k \ \mathtt{X} \, \} \text{ and } A_2 = A_3 = \ldots = A_{k+1} = \emptyset \, ,$$

and the rules are defined by

$$P_1 := \{p_1, p_1', p_i, p_k' \mid 2 \le i \le k\} \text{ and } P_i := \{q_i\}, 2 \le i \le k+1, \text{ where}$$

$$p_1 := \left(\begin{matrix} \mathtt{b}_1 \\ \boxed{\mathtt{b}_k} \end{matrix}, in_2 \right),$$

$$p_1' := \left(\boxed{\mathtt{X}}\, \mathtt{b}_1, in_2 \right),$$

$$p_i := \left(\begin{matrix} \mathtt{b}_i \\ \boxed{\mathtt{b}_{i-1}} \end{matrix}, in_{i+1} \right), \text{ for every } i,\ 2 \le i \le k,$$

$$p_k' := \left(\begin{matrix} \mathtt{X} \\ \boxed{\mathtt{b}_{k-1}} \end{matrix}, in_{k+1} \right),$$

$$q_i := \left(\begin{matrix} \mathtt{a}_{i-1} \\ \boxed{\mathtt{a}_{i-1}} \end{matrix}, out \right), \text{ for every } i,\ 2 \le i \le k+1.$$

Lemma 4 *For every $k \ge 1$, $L(\Pi_{comb,k}) = L_{comb,k}$.*

Proof It can be easily seen that every computation starts with p_1', q_2, p_2, q_3, p_3, ..., q_k and after applying these rules, membrane 1 contains the array

$$\begin{array}{ccccccc} & & & & & & \mathtt{b}_{k-1} \\ & & & & & & \vdots \\ \mathtt{a}_1 & \mathtt{a}_2 & \cdots & \mathtt{a}_{k-1} & & & \mathtt{b}_2 \\ \mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \cdots & \mathtt{a}_{k-1} & \mathtt{X}\ \mathtt{a}_k\ \mathtt{X} & \mathtt{b}_1 \end{array}.$$

Now, the only applicable rules are p_k or p_k' which both move the array to membrane $k+1$. Then rule q_{k+1} is applied, which moves the array back to membrane 1 and the array is now either of form

$$W := \begin{array}{ccccccc} & & & & & & \mathtt{b}_k \\ & & & & & & \mathtt{b}_{k-1} \\ & & & & & & \vdots \\ \mathtt{a}_1 & \mathtt{a}_2 & \cdots & \mathtt{a}_{k-1} & \mathtt{a}_k & & \mathtt{b}_2 \\ \mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \cdots & \mathtt{a}_{k-1} & \mathtt{X}\ \mathtt{a}_k\ \mathtt{X} & \mathtt{b}_1 \end{array} \quad \text{or} \quad W' := \begin{array}{ccccccc} & & & & & & \mathtt{X} \\ & & & & & & \mathtt{b}_{k-1} \\ & & & & & & \vdots \\ \mathtt{a}_1 & \mathtt{a}_2 & \cdots & \mathtt{a}_{k-1} & \mathtt{a}_k & & \mathtt{b}_2 \\ \mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \cdots & \mathtt{a}_{k-1} & \mathtt{X}\ \mathtt{a}_k\ \mathtt{X} & \mathtt{b}_1 \end{array},$$

depending on whether rule p_k or p_k' has been used. We note that $W' \in \Pi_{\mathrm{comb},k}$ and if W has been produced, then exactly the same arguments can be applied again. Thus, $\Pi_{\mathrm{comb},k}$ can only produce arrays of $L_{\mathrm{comb},k}$ and, obviously, all arrays of $L_{\mathrm{comb},k}$ can be produced in this way, which implies $L_{\mathrm{comb},k} = L(\Pi_{\mathrm{comb},k})$. □

Remark 2 The contextual rules in the P systems $\Pi_{\mathrm{comb},k}$, $k \ge 1$, have some interesting properties:

- No rule is #-sensing, i.e., any rule contains neither a selector with a blank symbol # nor a context with a blank symbol #.
- With the previously mentioned restriction, all rules are of smallest possible size, i.e., both the selector and the context contain exactly one symbol.
- Moreover, all rules are connected in the strongest possible sense, i.e., the selector and the context positions are either vertical or horizontal neighbors.

From Lemma 4, we can directly conclude the following:

Lemma 5 *For any $k \ge 1$, $L_{comb,k} \in \mathrm{AP}_{k+1}(cont)$.*

5.3 An infinite membrane hierarchy result

Next, we show that, for every $k \geq 1$, every contextual array P system that is able to describe $L_{\mathrm{comb},k}$ needs at least $\lceil \frac{k}{3} \rceil$ membranes:

Theorem 3 *For any $k \geq 1$, $L_{comb,k} \notin \mathrm{AP}_{\lfloor \frac{k}{3} \rfloor}(cont)$.*

From this main result of this section, we can immediately conclude:

Corollary 1 *The hierarchy of array languages*

$$\mathrm{AP}_1(cont) \subseteq \mathrm{AP}_2(cont) \subseteq \mathrm{AP}_3(cont) \subseteq \cdots$$

is infinite and, for every $k \geq 1$, $\mathrm{AP}_k(cont) \subsetneq \mathrm{AP}_{3k}(cont)$.

Unfortunately, we could not show a separation result like

$$\mathrm{AP}_k(cont) \subsetneq \mathrm{AP}_{k+1}(cont)$$

for any number k of membranes.

In order to prove Theorem 3, we need a kind of normal form of our rules:

Lemma 6 *Let Π be some P system with contextual array rules that generates $L_{comb,k}$. Then, each rule of Π can be classified into two groups:*

- *Either, a rule is* `X`*-sensing, meaning that it contains the symbol* `X` *in its selector;*
- *or, it is* `X`*-producing, meaning that it contains the symbol* `X` *in its context;*
- *or, a rule is of the form* $\begin{array}{c} \mathtt{a}_i \\ \vdots \\ \mathtt{a}_i \\ \boxed{\mathtt{a}_i} \\ \vdots \\ \boxed{\mathtt{a}_i} \end{array}$ *for some $i \leq k$, or some similar vertical one-dimensional shape for the rightmost tooth of the comb.*

Proof Assume that a rule is neither `X`-sensing nor `X`-producing. As its shape is 1-connected and as #-sensing is excluded, it has to be applied to one of the teeth, which leads to the form described in the third item. □

Especially for any derivation D that generates a large array from $L_{\mathrm{comb},k}$, there is a point T_D in the derivation after which no `X` can be sensed anymore, as Π is finite. We are going to apply the pigeon-hole principle in the following. In order to do so, the following lemma is helpful. In its formulation, we use the following notion: Let Π be a P system and let p be a rule of Π. Then, the *type* of rule p is specified by the pair of membranes (m_1, m_2), where m_1 is the membrane p belongs to and m_2 is the membrane where the processing will continue (as indicated by the target command in_j, *out* or *here*).

Lemma 7 *Any P system Π with m membranes has at most $3m-2$ different rule types. If Π has no rules labeled here, then it has rules of at most $2m-2$ different types.*

Proof It is well-known that any undirected tree with m nodes has $m-1$ edges. Clearly, the membrane structure of Π corresponds to such a tree. Each edge of this tree may be used in two directions (by the *out* or in_j labels); this gives (at most) $2m-2$ different types, and each rule labeled *out* or in_j has one of these types. Rules labeled *here* correspond to loops at some nodes. Hence, altogether Π cannot have more than $3m-2$ different rule types. □

Lemma 8 *No P system Π with m membranes can produce $L_{comb,3m}$.*

Proof Assume the contrary, so there exists some P system Π of diameter $d < \infty$ and m membranes that can produce $L_{\mathrm{comb},3m}$. We can assume that Π is in the normal form described in Lemma 6. Let x be the greatest diameter of any X-sensing rule in Π, and y be the greatest diameter of any axiom in Π. Let $h > x+y$ and consider the array $A_h \in L_{\mathrm{comb},3m}$ whose first tooth contains exactly h occurrences of $\mathtt{a}_1$. As $\mathrm{AL}(\Pi) = L_{\mathrm{comb},3m}$ by our assumption, there is some halting computation that outputs A_h. Let us discuss the derivation of A_h in this computation in the following. By the choice of h, for the i^{th} tooth, there must be a rule p_i of vertical one-dimensional shape that extends the i^{th} tooth in that derivation. By Lemma 7, Π contains at most $3m-2$ rule types. By the pigeon-hole principle, among the rules $p_1, \ldots, p_{3m-1}$, there must be two of the same type, say, p_I and p_J. Reconsider now the mentioned computation. At a certain step, one of the rules p_I or p_J was applied last. Without loss of generality, assume that p_I was applied later than p_J was ever applied. However, as also p_J has been applied (before that time), it is also applicable at that point of time, so we could replace one application of p_I by one of p_J to get a new terminating computation that will output an array that is not in $L_{\mathrm{comb},3m}$, as the I^{th} tooth is now shorter than h, while the J^{th} tooth will be longer than h. □

Lemma 8 concludes the proof of Theorem 3 and hence of Corollary 1.

Remark 3 We could have improved slightly on the claimed hierarchy gap, proving that no P system Π with m membranes can produce $L_{\mathrm{comb},2m}$, by making the following Claim:

> In the discussion of the non-$\mathtt{X}$ sensing rules working on teeth 1 through k in the proof of Lemma 8, none of these rules has the target indication *here*.

Namely, assume that some rule p_i of vertical one-dimensional shape that extends the i^{th} tooth in the derivation that we consider has target indication *here*. Then, after this rule has been applied in the derivation, it is still applicable, so it can be applied an arbitrary number of times. This would enable us to produce arrays that are not in the comb language. □

Lemma 7 would then yield the mentioned slightly stronger result, showing $\mathrm{AP}_k(cont) \subsetneq \mathrm{AP}_{2k}(cont)$.

Remark 4 In his book [16], Gheorghe Păun raised the question whether there exists a non-universal model of a membrane computing system which induces an infinite hierarchy on the number of membranes. The first, even dense, infinite hierarchy on linear membrane structures was established by Rudolf Freund in [7], with applying rules over a one-letter alphabet, but also using membrane dissolution. For static membrane structures, the first infinite hierarchy was presented

by Oscar Ibarra in [10]. Our hierarchy would offer an alternative solution to this problem. Like the one that we presented in this paper, Ibarra's hierarchy is not a dense one. Another similarity is that both hierarchies are built on "counter ideas". While Ibarra employs known results on counter (or multi-head finite-state machine) languages hierarchies, our comb teeth could be viewed as counters with some linear layout. While Ibarra's hierarchy also holds for unary alphabets, the one presented in this paper needs arbitrarily large alphabets. However, we believe that we can modify the comb language construction like in the star example to overcome this limitation.

We would like to mention that our hierarchy result does not crucially depend on certain aspects of the definition of P systems. For instance, it also holds if the final output is sent out to the environment, or if no specific output membrane is given at all, i.e., all "terminal arrays" would then be collected into the generated array language. Likewise, the result itself does not depend on another intricacy in the definition of array grammars, namely the aspect of #-sensing rules, as we will discuss now.

5.4 #-sensing contextual array P systems

We conclude this section by taking a closer look at #-sensing contextual array grammars and #-sensing contextual array P systems, i. e., the selectors of the contextual rules can contain the blank symbol #. We first observe that, for every $k \geq 1$, the language $L_{\mathrm{comb},k}$ can be easily generated by a #-sensing contextual array P system with only two membranes: in the first membrane a rule of form

$$\left(\begin{array}{ccccccccc} \mathtt{a}_1 & & \mathtt{a}_2 & & \cdots & & \mathtt{a}_k & & \\ \boxed{\mathtt{a}_1} & \boxed{\#} & \boxed{\mathtt{a}_2} & \boxed{\#} & \cdots & \boxed{\#} & \boxed{\mathtt{a}_k} & , & \mathit{in} \end{array} \right)$$

is used in order to extend every tooth in one step and then in the second membrane, we use a rule

$$\left(\begin{array}{c} \mathtt{b}_k \\ \vdots \\ \mathtt{b}_2 \\ \mathtt{b}_1 \\ \boxed{\mathtt{b}_k} \end{array} \right. \begin{array}{l} \\ \\ , \mathit{out} \\ \\ \\ \end{array} \left. \right) \quad \text{or} \quad \left(\begin{array}{c} \mathtt{X} \\ \vdots \\ \mathtt{b}_2 \\ \mathtt{b}_1 \\ \boxed{\mathtt{b}_k} \end{array} \right. \begin{array}{l} \\ \\ , \mathit{here} \\ \\ \\ \end{array} \left. \right)$$

in order to extend the $(k+1)^{\text{th}}$ tooth and move back to the first membrane or to extend the $(k+1)^{\text{th}}$ tooth and stop the computation, respectively. However, if we slightly change the languages $L_{\mathrm{comb},k}$, $k \geq 1$, then we can obtain a class of languages that allows us to prove a hierarchy result for #-sensing contextual array P systems.

For every $k \geq 1$, let $\widehat{L}_{\mathrm{comb},k}$ be the language that contains combs as in $L_{\mathrm{comb},k}$, but between the i^{th} and $(i+1)^{\text{th}}$ tooth of the comb there is an arbitrary number of occurrences of the symbol $\mathtt{X}_i$. For example,

$$\begin{array}{ccccccccccccccccc} & & & & & & & & & & & & & & & & \mathtt{X} \\ & & & & & & & & & & & & & & & & \mathtt{b}_3 \\ & & & & & & & & & & & & & & & & \mathtt{b}_2 \\ & & & & & & & & & & & & & & & & \mathtt{b}_1 \\ & & & & & & & & & & & & & & & & \mathtt{b}_4 \\ \mathtt{a}_1 & & & & \mathtt{a}_2 & & \mathtt{a}_3 & & & & & & \mathtt{a}_4 & & & & \mathtt{b}_3 \\ \mathtt{a}_1 & & & & \mathtt{a}_2 & & \mathtt{a}_3 & & & & & & \mathtt{a}_4 & & & & \mathtt{b}_2 \\ \mathtt{a}_1 & \mathtt{X}_1 & \mathtt{X}_1 & \mathtt{X}_1 & \mathtt{a}_2 & \mathtt{X}_2 & \mathtt{a}_3 & \mathtt{X}_3 & \mathtt{X}_3 & \mathtt{X}_3 & \mathtt{X}_3 & \mathtt{X}_3 & \mathtt{a}_4 & \mathtt{X}_4 & \mathtt{X}_4 & \mathtt{X}_4 & \mathtt{b}_1 \end{array}$$

is a member of $\widehat{L}_{\mathrm{comb},4}$. Next, we show that every language $\widehat{L}_{\mathrm{comb},k}$ can be generated by a contextual array P system $\widehat{\Pi}_{\mathrm{comb},k}$ with $k+2$ membranes. To this end, we modify $\Pi_{\mathrm{comb},k}$ by adding a new outermost membrane with rules

$$\left(\boxed{\mathtt{a}_i}\ \mathtt{X}_i, here\right), \left(\boxed{\mathtt{X}_i}\ \mathtt{a}_{i+1}, here\right), \left(\boxed{\mathtt{X}_i}\ \mathtt{X}_i, here\right), \text{ for every } i, 1 \le i \le k-1,$$
$$\left(\boxed{\mathtt{a}_k}\ \mathtt{X}_k, here\right), \left(\boxed{\mathtt{X}_k}\ \mathtt{X}_k, here\right), \left(\boxed{\mathtt{a}_k}\ \mathtt{X}_k, in\right) \text{ and } \left(\boxed{\mathtt{X}_k}\ \mathtt{X}_k, in\right).$$

These rules turn the axiom, which is now $\mathtt{a}_1$, into a one-dimensional array

$$\mathtt{a}_1\,\mathtt{X}_1\,\mathtt{X}_1\ \ldots\ \mathtt{X}_1\,\mathtt{a}_2\,\mathtt{X}_2\,\mathtt{X}_2\ \ldots\ \mathtt{X}_2\,\mathtt{a}_3\ \ldots\ \mathtt{a}_k\,\mathtt{X}_k\,\mathtt{X}_k\ \ldots\ \mathtt{X}_k$$

and move this array into the next membrane, which is the outermost membrane of the contextual array P system $\Pi_{\mathrm{comb},k}$. From now on, the P system $\widehat{\Pi}_{\mathrm{comb},k}$ works in a similar way as $\Pi_{\mathrm{comb},k}$; thus, a member of $\widehat{L}_{\mathrm{comb},k}$ is generated.

We can now adapt the proof of Theorem 3 to #-sensing contextual array P systems. To this end, we note that apart from possible sensing of some blank symbols # in non-X-sensing rules, the normal form of Lemma 6 is still valid. In Lemma 8, we only have to take care of selecting an array whose teeth are at a distance large enough of each other to prevent any sensing of neighboring teeth in the derivation.

6 Conclusion

We have introduced P systems with array objects and contextual array productions of the kind considered in [8], but with no pixel in any selector in the rules having # as a label, and we have shown that the P system model has more generative power than the contextual array grammars working in the maximal mode themselves have in generating two-dimensional arrays. Moreover, the expressive power of contextual array P systems strictly increases with the number of membranes, thereby giving a proper hierarchy with respect to the classes of languages described by contextual array P systems. A corresponding result also holds when we allow the #−sensing ability in the selector. But it remains open whether such hierarchy results hold from k to $k+1$, for any number k of membranes.

Acknowledgements

K.G. Subramanian gratefully acknowledges partial support for this research from a FRGS grant No. 203/PKOMP/6711267 of the Ministry of Higher Education, Malaysia, and support by the project "Universität der Großregion — UniGR" that in particular enabled his visit at the University of Trier. Markus L. Schmid was partially supported by a scholarship of the German Academic Exchange Service (DAAD) for returning scientists. Petra Wiederhold is grateful to CONACYT Mexico for partial support from grant No. CB-2011-01-166223. We appreciate the efficient work of the referees..

References

1. Bottoni, P., Labella, A., Martín-Vide, C., Păun, G.: Rewriting P systems with conditional communication. In: W. Brauer, H. Ehrig, J. Karhumäki, A. Salomaa (eds.) Formal and Natural Computing, *LNCS*, vol. 2300, pp. 325–353. Springer (2002)
2. Ceterchi, R., Mutyam, M., Păun, G., Subramanian, K.G.: Array-rewriting P systems. Natural Computing **2**(3), 229–249 (2003)
3. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, G. (eds.): Grammar Systems: A Grammatical Approach to Distribution and Cooperation. Gordon and Breach Science Publishers, Inc. Newark, NJ, USA (1994)
4. Ehrenfeucht, A., Păun, G., Rozenberg, G.: Contextual grammars and formal languages. In: G. Rozenberg, A. Salomaa (eds.) Handbook of Formal Languages, Volume II, pp. 237–293. Berlin: Springer (1997)
5. Fernau, H., Freund, R.: Bounded parallelism in array grammars used for character recognition. In: P. Perner, P. Wang, A. Rosenfeld (eds.) Advances in Structural and Syntactical Pattern Recognition (Proceedings of the SSPR'96), *LNCS*, vol. 1121, pp. 40–49. Springer (1996)
6. Ferretti, C., Mauri, G., Paun, G., Zandron, C.: On three variants of rewriting P systems. Theoretical Computer Science **301**(1-3), 201–215 (2003)
7. Freund, R.: Special variants of P systems inducing an infinite hierarchy with respect to the number of membranes. EATCS Bulletin **75**, 209–219 (2001)
8. Freund, R., Păun, G., Rozenberg, G.: Chapter 8: Contextual array grammars. In: K.G. Subramanian, K. Rangarajan, M. Mukund (eds.) Formal Models, Languages and Applications, *Series in Machine Perception and Articial Intelligence*, vol. 66, pp. 112–136. World Scientic (2007)
9. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: G. Rozenberg, A. Salomaa (eds.) Handbook of Formal Languages, Volume III, pp. 215–267. Berlin: Springer (1997)
10. Ibarra, O.H.: On membrane hierarchy in P systems. Theoretical Computer Science **334**(1-3), 115–129 (2005)
11. Klette, R., Rosenfeld, A. (eds.): Digital Geometry. Elsevier (2004)
12. Krithivasan, K., Mutyam, M.: Contextual P systems. Fundamenta Informaticae **49**(1-3), 179–189 (2002)
13. Marcus, S.: Contextual grammars. Rev. Roum. Math. Pures Appl. **14**, 1525–1534 (1969)
14. Paun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1), 108–143 (2000)
15. Păun, G.: Marcus contextual grammar. Studies in Linguistics and Philosophy. Dordrecht: Kluwer Academic Publishers (1997)
16. Păun, G.: Computing with Membranes: An Introduction. Springer, Berlin (2002)
17. Rosenfeld, A. (ed.): Picture Languages. Academic Press, Inc. Orlando, FL, USA (1979)
18. Rosenfeld, A., Kak, A.C. (eds.): Digital picture processing. Academic Press, Inc. Orlando, USA (1982)
19. Rosenfeld, A., Siromoney, R.: Picture languages – a survey. Languages of Design **1**, 229–245 (1993)
20. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages (3 volumes). Springer (1997)
21. Salomaa, A.K.: Formal Languages. Academic Press (1973)
22. Subramanian, K.G.: P systems and picture languages. In: J.O. Durand-Lose, M. Margenstern (eds.) Machines, Computations, and Universality, 5th International Conference, MCU, *LNCS*, vol. 4664, pp. 99–109. Springer (2007)
23. Subramanian, K.G., Ali, R.M., Geethalakshmi, M., Nagar, A.K.: Pure 2D picture grammars and languages. Discrete Applied Mathematics **157**(16), 3401–3411 (2009)
24. Subramanian, K.G., Revathy, L., Siromoney, R.: Siromoney array grammars and applications. International Journal of Pattern Recognition and Artificial Intelligence **3**, 333–351 (1989)
25. Subramanian, K.G., Venkat, I., Wiederhold, P.: A P system model for contextual array languages. In: R.P. Barneva, V.E. Brimkov, J.K. Aggarwal (eds.) Combinatorial Image Analysis - 15th International Workshop, IWCIA, *LNCS*, vol. 7655, pp. 154–165. Springer (2012)
26. Wang, P.S.P.: An application of array grammars to clustering analysis for syntactic patterns. Pattern Recognition **17**, 441–451 (1984)

27. Wang, P.S.P.: Array Grammars, Patterns and Recognizers. World Scientific Publishing Co., Singapore (1989)
28. Zandron, C., Ferretti, C., Mauri, G.: Two normal forms for rewriting P systems. In: M. Margenstern, Y. Rogozhin (eds.) Machines, Computations, and Universality, Third International Conference, MCU, *LNCS*, vol. 2055, pp. 153–164. Springer (2001)