A natural infinite membrane hierarchy

Henning Fernau

Universität Trier

fernau@uni-trier.de

Theorietag Automaten und Formale Sprachen
Ilmenau, September 2013

History of the paper

- K. G. Subramanian visited Trier (Grande Région) in January 2013.
- Turning his IWCIA 2012 paper to a journal paper was one of his to-dos.
- This project brought others into play, so that the list of authors is pretty long:
  Henning Fernau, Rudolf Freund, Markus L. Schmid, $\leftarrow$ present here!
  K. G. Subramanian and Petra Wiederhold
- Main result of this project:
  A natural infinite membrane hierarchy

Overview on the talk

- Array grammars with contextual rules
- Membrane mechanisms
- Infinite hierarchies: a riddle on membranes
- Combs: a solution to the riddle
- Discussions

Arrays / Pictures

A *two-dimensional array or a picture* is a function $\alpha : \mathbb{Z}^2 \to V \cup \{\#\}$ with finite and connected support $supp(\alpha)$, given by $supp(\alpha) = \{v \in \mathbb{Z}^2 | \alpha(v) \neq \#\}$.
V^{+2}: The set of all two-dimensional arrays over the alphabet V.

An example:

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

More formally, restricting on the support:
$\{((0,0),\mathtt{a}),((1,0),\mathtt{a}),((2,0),\mathtt{a}),$
$((3,0),\mathtt{a}),((4,0),\mathtt{a}),((5,0),\mathtt{a}),$
$((6,0),\mathtt{a}),((0,1),\mathtt{a}),((0,2),\mathtt{a}),$
$((0,3),\mathtt{a}),((0,4),\mathtt{a}),((0,5),\mathtt{a})\}$

Mostly, exact positions are irrelevant $\rightsquigarrow$ equivalence classes under translation.

Contextual Grammars

- a classical mechanism invented by S. Marcus about 50 years ago;
- the central idea is that of *adjoining* fitting pieces;
  "fitness" can be tested by using *selectors*;
  if fitting, the *context* can be adjoined.
- this has been adapted also to the case of arrays before.

Contextual Array Rules have the form $p = (\alpha, \beta)$, where

- α is a function defined on $U_\alpha \subset \mathbb{Z}^2$ with values in V;
- β is a function defined on $U_\beta \subset \mathbb{Z}^2$ with values in V;
- $U_\alpha \cap U_\beta = \emptyset$, U_α, U_β are finite.

(U_α, α) is called the *selector* and $\left(U_\beta, \beta\right)$ the *context* of the production (α, β);
U_α is called the *selector area*, and U_β is the *context area*.
$\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$: find an occurrence of p's selector as a sub-array in $\mathcal{C}_1$;
if then the context area is all blanks, replace it by the context to obtain $\mathcal{C}_2$.

A small example

Consider the rule: $p_1 := \begin{array}{ccc} \boxed{\mathtt{a}} & \mathtt{a} & \\ \boxed{\mathtt{a}} & \boxed{\mathtt{a}} & \boxed{\mathtt{a}} \end{array}$

Here, the selector is shown by box-framing some symbols.

Consider the array $\begin{array}{ccc} \mathtt{a} & & \\ \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} \end{array}$

The selector occurs twice, but only in the green situation, the context can be inserted.

Contextual Array Grammars are constructs $G = (V, P, A)$ 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 contextual array rules.
- $\mathcal{C}_1 \Longrightarrow_G \mathcal{C}_2$ means $\exists p \in P : \mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$.
- $\mathcal{A} \Longrightarrow_G^t \mathcal{B}$ if and only if $\mathcal{A} \Longrightarrow_G^* \mathcal{B}$ and there is no $\mathcal{C} \in V^{+2}$ with $\mathcal{B} \Longrightarrow_G \mathcal{C}$.
- $L_t(G) \;=\; \left\{\mathcal{B} \in V^{+2} \mid \mathcal{A} \Longrightarrow_G^t \mathcal{B} \text{ for some } \mathcal{A} \in A\right\}.$

A small example

$$p_1 := \begin{array}{ccc} \boxed{\mathtt{a}} & \mathtt{a} & \\ \boxed{\mathtt{a}} & \boxed{\mathtt{a}} & \boxed{\mathtt{a}} \end{array}, \quad p_2 := \begin{array}{ccc} \boxed{\mathtt{a}} & \mathtt{a} & \mathtt{a} \\ \boxed{\mathtt{a}} & \boxed{\mathtt{a}} & \mathtt{a} \end{array}, \quad p_3 := \begin{array}{cc} \boxed{\mathtt{a}} & \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}} & \mathtt{a} \\ \boxed{\mathtt{a}} & \end{array}, \quad p_4 := \begin{array}{cc} \boxed{\mathtt{a}} & \boxed{\mathtt{a}} \\ \boxed{\mathtt{a}} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} \end{array}$$

$$\begin{array}{ccc} \mathtt{a} & & \\ \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} \end{array} \Longrightarrow_{p_1} \begin{array}{ccc} \mathtt{a} & \mathtt{a} & \\ \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} \end{array} \Longrightarrow_{p_2} \begin{array}{cccc} \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{a} & \\ \mathtt{a} & \mathtt{a} & \mathtt{a} & \end{array} \Longrightarrow_{p_3} \begin{array}{cccc} \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} & \end{array} \Longrightarrow_{p_4} \begin{array}{cccc} \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{c} & \mathtt{a} & \mathtt{a} \\ \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{a} \\ & & \mathtt{a} & \mathtt{a} \end{array}$$

. . . continues . . . and stops (t-mode)

$$p_5 := \begin{array}{ccc} \boxed{\texttt{a}} & \boxed{\texttt{a}} & \boxed{\texttt{a}} \\ & \texttt{a} & \boxed{\texttt{a}} \end{array}, \quad p_6 := \begin{array}{ccc} \texttt{a} & \boxed{\texttt{a}} & \boxed{\texttt{a}} \\ \texttt{a} & \texttt{a} & \boxed{\texttt{a}} \end{array}, \quad p_7 := \begin{array}{cc} \texttt{a} & \\ \texttt{a} & \boxed{\texttt{a}} \\ \boxed{\texttt{a}} & \boxed{\texttt{a}} \end{array}, \quad p_8 := \begin{array}{cc} \texttt{a} & \boxed{\texttt{a}} \\ \boxed{\texttt{a}} & \boxed{\texttt{a}} \end{array}.$$

$$\Longrightarrow_{p_5} \begin{array}{cccc} \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{c} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ & \texttt{a} & \texttt{a} & \texttt{a} \end{array} \Longrightarrow_{p_6} \begin{array}{ccccc} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ & \texttt{a} & \texttt{c} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \end{array} \Longrightarrow_{p_8} \begin{array}{ccccc} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{c} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \end{array} \Longrightarrow_{p_7} \begin{array}{ccccc} \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{c} & \texttt{a} & \texttt{a}. \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \\ \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} & \texttt{a} \end{array}$$

Contextual Array Grammars are (in a sense) very weak.

- The simple one-dimensional array language consisting of all “lines” of thickness one over the alphabet a cannot be generated.
- Instead, we need a *stopper* at the end.
- Even with stopper(s), L-shaped arrays (thickness one, equal “arm lengths”) cannot be described.

$\rightsquigarrow$ We need some sort of control mechanism.
The IWCIA paper leads us to P systems.
This can be also seen as a kind of tree control.

Membrane computing on array structures

- The membrane structure can be described by labeled brackets.
- Given by a construct $\Pi = (V, \#, \mu, A_1, \ldots, A_m, P_1, \ldots, P_m, i_o)$,
- A_i: axiom set of membrane i.
- P_i: rule set of membrane i; each rule has a direction:
- targets *here*, *out*, *in*$_j$, $1 \leq j \leq m$, and *in*.
- i_o is the *output membrane* for the results of the computation.

Combs

The language of combs with 4 teeth (where the i^{th} tooth is defined over the alphabet $\{a_i\}$, $1 \le i \le 4$) denoted by $L_{\text{comb},4}$, is the following set

$$\left\{ \begin{array}[b]{ccccccccc}
 & & & & & & & & \mathtt{X} \\
 & & & & & & & & \mathtt{b}_3 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & & \mathtt{a}_4 & & \mathtt{b}_2 \\
\mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 & \mathtt{X} & \mathtt{a}_4 & \mathtt{X} & \mathtt{b}_1
\end{array} , \;
\begin{array}[b]{ccccccccc}
 & & & & & & & & \mathtt{X} \\
 & & & & & & & & \mathtt{b}_3 \\
 & & & & & & & & \mathtt{b}_2 \\
 & & & & & & & & \mathtt{b}_1 \\
 & & & & & & & & \mathtt{b}_4 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & & \mathtt{a}_4 & & \mathtt{b}_3 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & & \mathtt{a}_4 & & \mathtt{b}_2 \\
\mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 & \mathtt{X} & \mathtt{a}_4 & \mathtt{X} & \mathtt{b}_1
\end{array} , \;
\begin{array}[b]{ccccccccc}
 & & & & & & & & \mathtt{X} \\
 & & & & & & & & \mathtt{b}_3 \\
 & & & & & & & & \mathtt{b}_2 \\
 & & & & & & & & \mathtt{b}_1 \\
 & & & & & & & & \mathtt{b}_4 \\
 & & & & & & & & \mathtt{b}_3 \\
 & & & & & & & & \mathtt{b}_2 \\
 & & & & & & & & \mathtt{b}_1 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & & \mathtt{a}_4 & & \mathtt{b}_4 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & & \mathtt{a}_4 & & \mathtt{b}_3 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & & \mathtt{a}_4 & & \mathtt{b}_2 \\
\mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 & \mathtt{X} & \mathtt{a}_4 & \mathtt{X} & \mathtt{b}_1
\end{array} , \; \ldots \right\}$$

Where is the stopper?

$L_{\textbf{comb},k}$ contains combs like in $L_{\text{comb},4}$, but with k instead of 4 teeth.

$$\begin{array}{ccccccccc}
 & & & & & & & & \mathtt{X} \\
 & & & & & & & & \mathtt{b}_3 \\
 & & & & & & & & \mathtt{b}_2 \\
 & & & & & & & & \mathtt{b}_1 \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & \ldots & \mathtt{a}_k & & \vdots \\
\vdots & & \vdots & & \vdots & \vdots & \vdots & & \\
 & & & & & & & & \mathtt{b4} \\
 & & & & & & & & \mathtt{b3} \\
\mathtt{a}_1 & & \mathtt{a}_2 & & \mathtt{a}_3 & \ldots & \mathtt{a}_k & & \mathtt{b}_2 \\
\mathtt{a}_1 & \mathtt{X} & \mathtt{a}_2 & \mathtt{X} & \mathtt{a}_3 & \ldots & \mathtt{a}_k & \mathtt{X} & \mathtt{b}_1
\end{array} .$$

A P system with $k+1$ membranes that generates the language $L_{\mathsf{comb},k}$

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

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

$$p_1 := \left(\begin{array}{c} \mathtt{b}_1 \\ \boxed{\mathtt{b}_k} \end{array}, \mathsf{in}_2 \right), \quad p_1' := (\boxed{\mathtt{X}}\ \ \mathtt{b}_1,\ \mathsf{in}_2) \text{ THE START,}$$

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

$$p_k' := \left(\begin{array}{c} \mathtt{X} \\ \boxed{\mathtt{b}_{k-1}} \end{array}, \mathsf{in}_{k+1} \right) \text{ THE END,}$$

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

Overview on the talk

- Array grammars with contextual rules
- Membrane mechanisms
- Infinite hierarchies: a riddle on membranes
- Combs: a solution to the riddle
- Discussions

Riddles for membrane computing

- Question (by Gh. Păun): Find an infinite non-universal hierarchy of languages induced by the number of membranes of the involved P systems.
- Ibarra's answer from 2003: The Number of Membranes Matters.
  His construction is based on results on multihead finite automata
  (exhibiting P system simulations via counters).
- A kind of earlier answer from 2001 by R. Freund.
- Here, we provide an alternative answer, based on well-established grammar models, using some interchange argument.

A natural infinite membrane hierarchy

$\mathsf{AP}_k(cont)$: The array languages that can be obtained with k membranes with contextual rules.

Lemma: For any $k \geq 1$, $L_{\text{comb},k} \in \mathsf{AP}_{k+1}(cont)$.

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

Corollary: The hierarchy of array languages

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

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

Proving the Main Theorem

Lemma 1: W.l.o.g., there is one axiom of one membrane that is

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

Idea: If this was not the case, then there is no way of synchronizing thre number of teeth and the length of the last tooth. Of course, the back of the comb could be generated step by step, but this could be shortcut by the suggested axiom.

If there are more and other axioms, this does not change the following arguments.

Proving the Main Theorem

Lemma 2: Let Π be some P system with contextual array rules that generates $L_{\text{comb},k}$. Then, each rule of Π can be classified into two groups:

- Either, a rule is $\mathtt{X}$-sensing, meaning that it contains $\mathtt{X}$ in its selector;
- 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 one-dimensional shape for the rightmost tooth of the comb.

Idea: A non-$\mathtt{X}$-sensing rule cannot synchronize between different teeth.

Proving the Main Theorem

Let Π be a P system and let p be a rule of Π.
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 3: Any P system Π with m membranes has at most $3m - 2$ different rule types. If Π has no rules labelled $here$, then it has rules of at most $2m - 2$ different types.

Proof: It is well-known that any undirected tree on 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 labelled out or in_j has one of these types. Rules labelled $here$ correspond to loops at some nodes. Hence, altogether Π cannot have more than $3m - 2$ different rule types.

Proving the Main Theorem

Lemma 4: No P system Π with m membranes can produce $L_{\text{comb},3m}$.

Proof: Assume the contrary, so there exists some P system Π with m membranes that can produce $L_{\text{comb},3m}$. We can assume that Π is in the normal form described in Lemmas 1 and 2. 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 $A_h \in L_{\text{comb},3m}$ whose first tooth contains exactly h occurrences of $\mathtt{a}_1$.
As $\mathsf{AL}(\Pi) = L_{\text{comb},3m}$ 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 4, Π contains at most $3m-2$ rule types. By the pigeon-hole principle, among the rules $p_1, \dots, p_{3m-1}$, there must be two of the same type, say, p_I and p_J.

Reconsider now the mentioned computation. W.l.o.g., 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 $L_{\text{comb},3m}$, as the I^{th} tooth is now shorter than h, while the J^{th} tooth will be longer than h.

Improving the gap in the Main Theorem

Remark: No P system Π with m membranes can produce $L_{\text{comb},2m}$, by making the following Claim:

> In the discussion of the non-`X` sensing rules working on tooth 1 through k in the proof of Lemma 4, none of these rules has the target indication $here$.

Open Problem: Close the gap perfectly!
Prove that $\mathsf{AP}_1(cont) \subsetneq \mathsf{AP}_2(cont) \subsetneq \mathsf{AP}_3(cont) \subsetneq \mathsf{AP}_4(cont) \cdots$
Notice: Ibarra's hierarchy is not dense, either!

We could only show:
Lemma 5: $\mathsf{AP}_1(cont) \subsetneq \mathsf{AP}_2(cont) \subsetneq \mathsf{AP}_3(cont)$.
Separating examples are L- or T-shaped arrays, resp.

Discussions

- The choice of the regulation mechanism was kind of arbitrary.
- Similar results holds for graph-controlled contextual array grammars of various types.
- More generally, restricting the number of vertices in a graph-control might be an interesting measure of descriptional complexity, related but not identical to the number of productions.

Thanks for your attention !

Theoretische

Informatik Trier