Parsing Languages of P Colony Automata

Erzsébet Csuhaj-Varjú

Eötvös Loránd University, Budapest, Hungary

Kristóf Kántor, **György Vaszil**

University of Debrecen, Hungary

Eötvös Loránd University
Faculty of Informatics

University of Debrecen
Faculty of Informatics

Motivation, background, ...

Parallel architectures, **networks**, **internet**: A **modification** of the **"classic",** imperative **programming/computing** paradigm might be interesteing.

➔A "**chemical style**" approach to the notion of **computation**.

The goal is to free algorithms from the kind of sequentiality which is the consequence of the underlying (sequential) computational model.

"Chemistry" as a metaphor

- **Information** is stored in the **structure** and the properties of **molecules**
- **Chemical reaction → information processing**

data	substances or molecules
processing	chemical reaction
algorithm	substances and their reaction laws

In a more formal setting:

- **multiset** as **data structure**
- **multiset transformation/**processing as **computation**

“Chemical” models

- **Gamma** programming **formalism** (J.P. Banatre)
- **Chemical abstract machine** (G. Boudol)
- etc.

- **Membrane systems**, P systems (G. Paun)
  - **P colonies**

P colonies

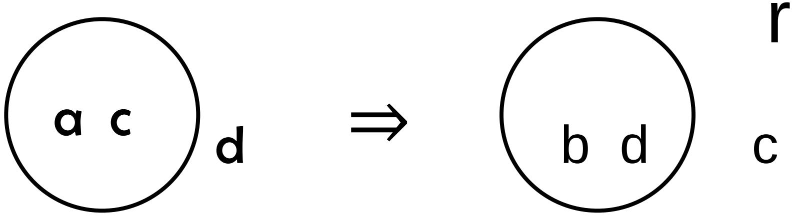
- A population of very **simple cells/computing units** in a **shared environment**:
  - **Fixed number** of objects (1, 2, 3,…) inside each cell
  - **Simple** rules (programs) for **moving** and **changing** the objects

- The objects are **exchanged** directly only between the **cells** and the **environment**

[Kelemen, Kelemenová, Paun 2004]

P colonies

Programs made of rules for rewriting + communication

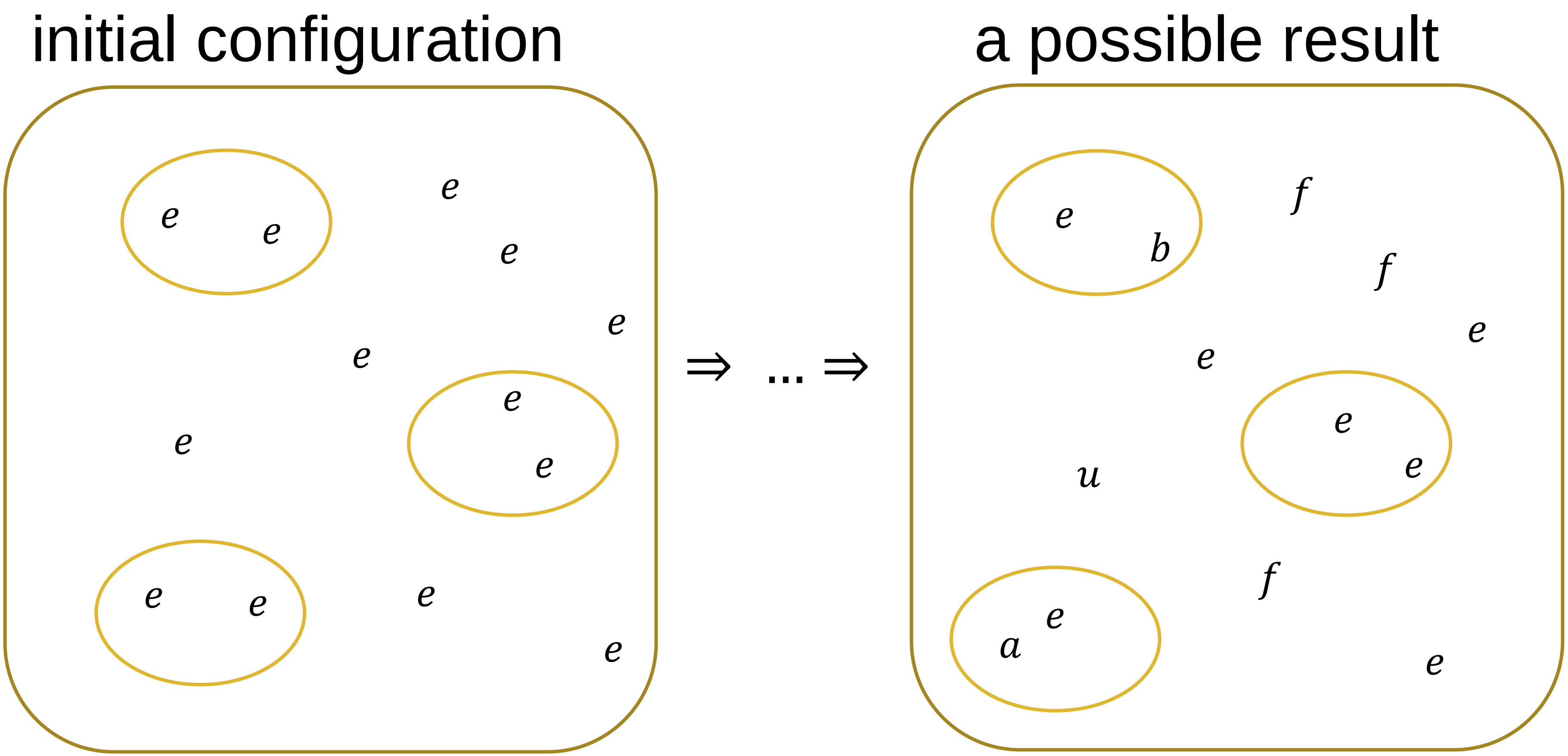


$$(a \to b, c \leftrightarrow d)$$

The computation

- Start in an **initial configuration: objects inside the cells**
- Apply a maximal set of programs in **parallel** in the cells, **halt** if no program is applicable
- The **result** of the computation:
  - **Numbers** - the **multiplicity** of certain objects found in the **environment**

The computation



Computational power

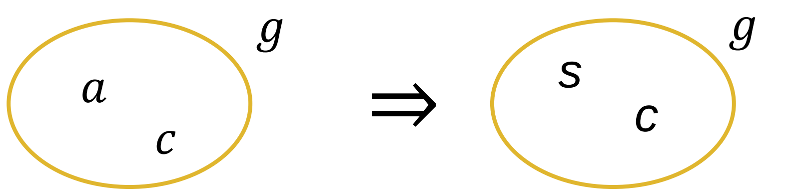
- **Variants of P colonies** can generate complex sets, most of the times **any recursively enumerable** set of numbers, **sometimes less**.

[Csuhaj-Varjú, Kelemen, Kelemenová, Paun, Vaszil 2006a]

[Ciencielová, Csuhaj Varjú, Kelemenová, Vaszil 2009]

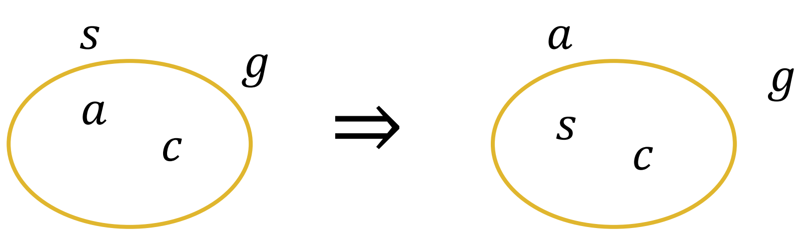
How to obtain strings – tape rules

The **application of** certain **rules** is associated with **“reading”** certain input **symbols**:



$$(a \xrightarrow{T} s;\ c \to c)$$

Reading an *s* with a **rewriting tape rule**



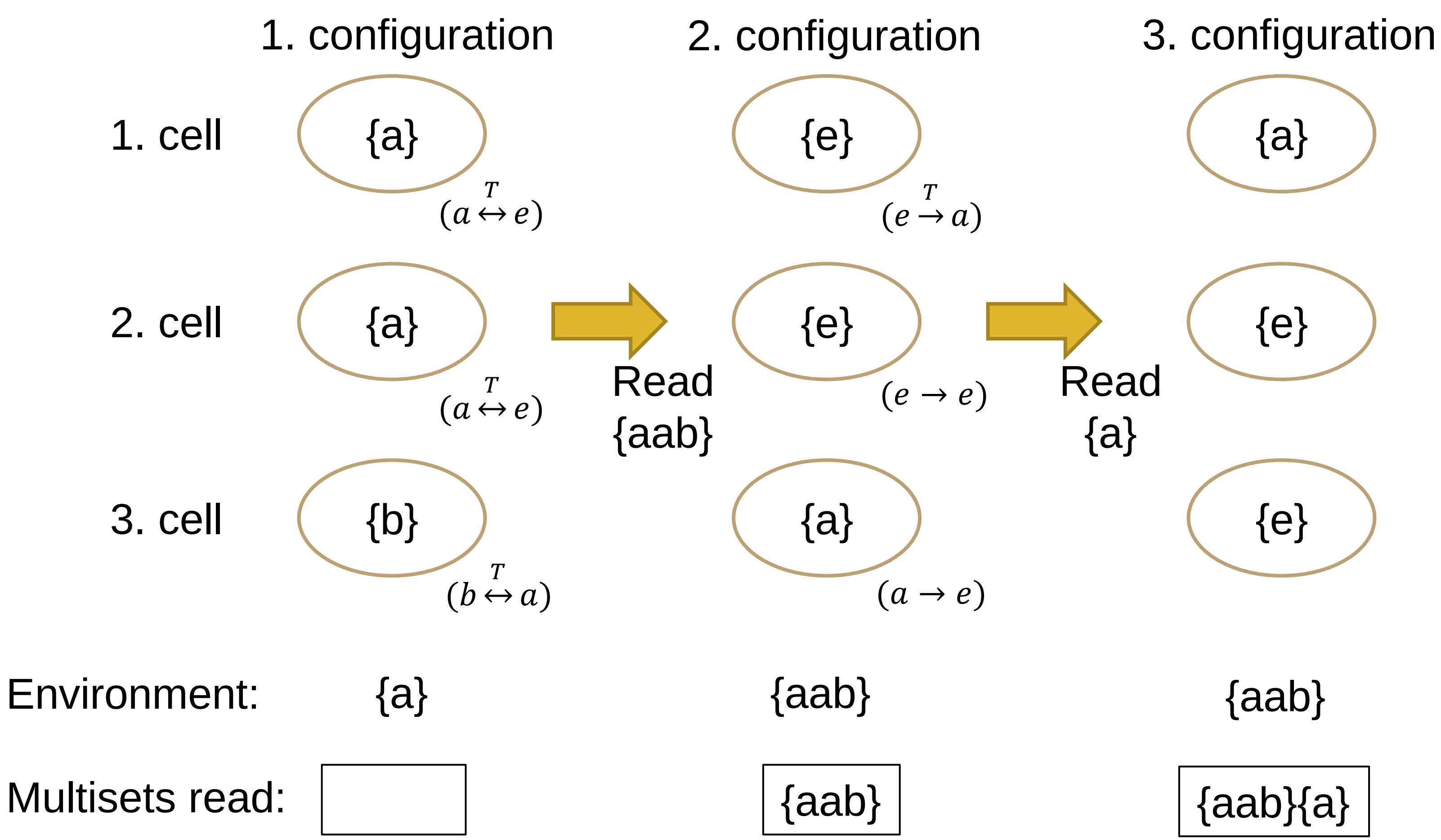
$$(s \overset{T}{\leftrightarrow} a;\ c \to c)$$

Reading an *s* with a **communication tape rule**

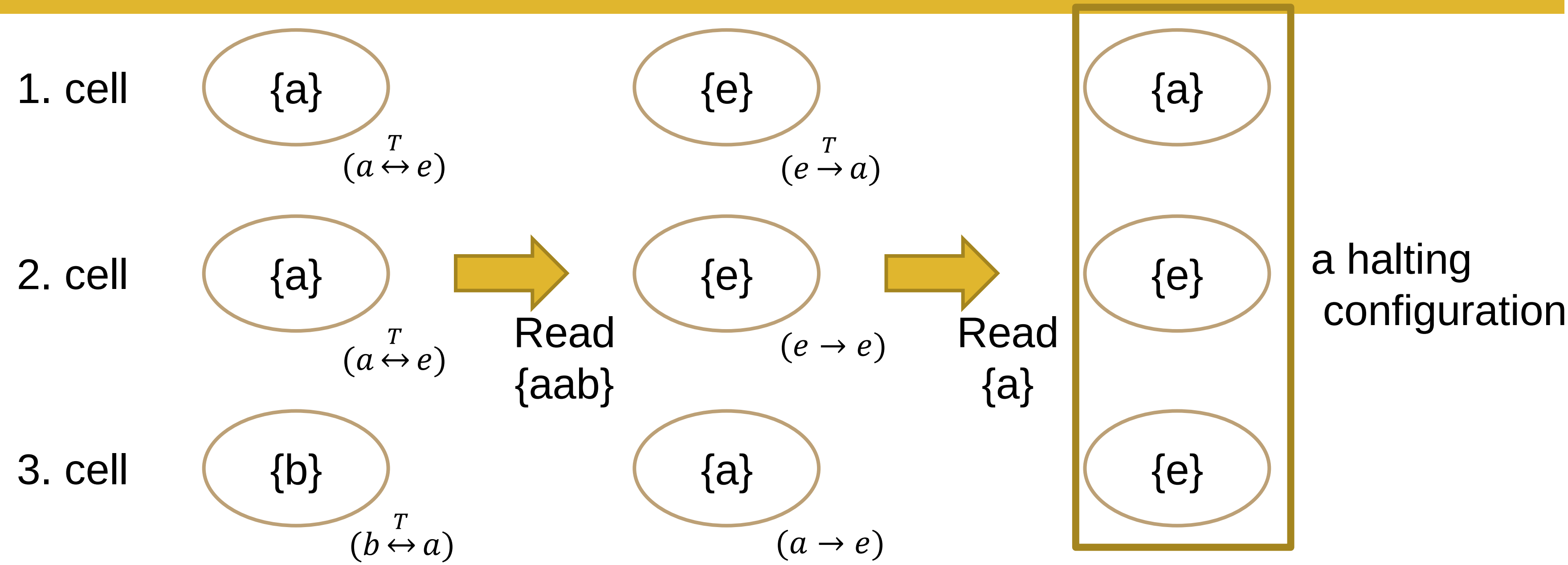
Generalized P colony automata

- A **maximal set** of programs is chosen, tape rules and non-tape rules together
- The chosen tape rules might "read" several **different symbols:**
  - A **multiset** is read in one **computational step**
  - A **sequence of multisets** is read during a **computatio**n

Computation and rules – small example



The accepted strings, the input mapping



Environment: {a} {aab} {aab}

Multisets read: [ ] [{aab}] [{aab}{a}]

If *f(aab)={00,1}*, *f(a)={1}*,…. then *f({aab}{a})={00,1}{1}*, that is, 001 and 11 belong to the language

Parsing - Reconstructing the string generation/acceptance process

The reconstruction should be deterministic, like for CF grammars: LR(k) grammars, LL(k) grammars

- For P colony automata?

For example...

A grammar:

S -> aB | bA
A -> a | aS | bAA
B -> b | bS | aBB

	aabb#
	S#

	aabb#
S_1	aB#

a	abb#
S_1a	B#

a	abb#
S_1aB_3	aBB#

aa	bb#
S_1aB_3a	BB#

aa	bb#
$S_1aB_3aB_1$	bB#
$S_1aB_3aB_2$	bSB#

aab	b#
$S_1aB_3aB_1b$	B#
$S_1aB_3aB_2b$	SB#

aab	b#
$S_1aB_3aB_1bB_1$	b#
$S_1aB_3aB_1bB_2$	bS#
$S_1aB_3aB_2bS_2$	bAB#

aabb	#
$S_1aB_3aB_1bB_1b$	#
$S_1aB_3aB_1bB_2b$	S#
$S_1aB_3aB_2bS_2b$	AB#

aabb#	
$S_1aB_3aB_1bB_1b\#$	

For example...

- An LL(1) grammar:

$$S \to aB$$
$$B \to b \mid aBb$$

	aabb#	→		aabb#	→	a	abb#	→
	S#		S_1	aB#		S_1a	B#	

a	abb#	→	aa	bb#	→	aa	bb#	→
S_1aB_2	aBb#		S_1aB_2a	Bb#		$S_1aB_2aB_1$	bb#	

aab	b#	→	aabb	#	→	aabb#	
$S_1aB_2aB_1b$	b#		$S_1aB_2aB_1bb$	#		$S_1aB_2aB_1bb\#$	

As we have seen

- $\{a^n b^n \mid n>1\}$ is an LL(1) language

But it is also clear:

- $\{a^n b^n \mid n>1\} \cup \{a^n c^n \mid n>1\}$ is **not** an LL(k) language for any k

How to apply the idea in P colonies?

Informally:

The **next *k* symbols** of the not-yet-generated part of the **string** to be obtained **determines the cells and the programs** to be applied in the next computational step.

More formally

Let $\mathrm{FIRST}_k(U) = \{pref_k(u) \in \Sigma^* \mid u \in U\}$

Consider **two computations** from configuration c_s

$$c_s \xRightarrow{P_{c_s}} c_{s+1} \xRightarrow{P_{c_{s+1}}} \dots \xRightarrow{P_{c_{s+m}}} c_{s+m+1}, \text{ and } c_s \xRightarrow{P'_{c_s}} c'_{s+1} \xRightarrow{P_{c'_{s+1}}} \dots \xRightarrow{P_{c'_{s+m'}}} c'_{s+m'+1}$$

with **input** sequences:

$$u_{c_s} u_{c_{s+1}} \dots u_{c_{s+m}} \text{ and } u'_{c_s} u_{c'_{s+1}} \dots u_{c'_{s+m'}}$$

$$w \in f(u_{c_s}) f(u_{c_{s+1}}) \dots f(u_{c_{s+m}}) \text{ and } w' \in f(u'_{c_s}) f(u_{c'_{s+1}}) \dots f(u_{c'_{s+m'}})$$

The **genPCol automaton is LL(k)** if $P_{c_s} \neq P'_{c_s}$ implies

$$\mathrm{FIRST}_k(w) \cap \mathrm{FIRST}_k(w') = \emptyset.$$

Example with 1 symbol lookahead

$P =$

$\langle e \to b, a \overset{T}{\leftrightarrow} e \rangle$	$\langle e \to \boldsymbol{c}, a \overset{T}{\leftrightarrow} e \rangle$	$\langle e \to \boldsymbol{d}, \boldsymbol{c} \overset{T}{\leftrightarrow} b \rangle$	$\langle e \to \boldsymbol{g}, \boldsymbol{f} \overset{T}{\leftrightarrow} b \rangle$
$\langle e \to e, b \overset{T}{\leftrightarrow} a \rangle$	$\langle e \to f, a \overset{T}{\leftrightarrow} e \rangle$	$\langle b \to c, d \overset{T}{\leftrightarrow} e \rangle$	$\langle b \to \boldsymbol{f}, f \overset{T}{\leftrightarrow} e \rangle$

$F =$

$\left\{ \begin{matrix} (v, ce), \\ (v, fe) \end{matrix} \middle| \; v \in V^*, \; b \notin v \right\}$

Possible computation:

$(ea) \overset{(a)}{\Longrightarrow} a\ (eb) \overset{(b)}{\Longrightarrow} b\ (ea) \overset{(a)}{\Longrightarrow} ab\ (eb) \overset{(b)}{\Longrightarrow} bb\ (ea)$

$\overset{(a)}{\Longrightarrow} bba\ (ce) \overset{(c)}{\Longrightarrow} bac\ (bd) \overset{(d)}{\Longrightarrow} bacd\ (ce) \overset{(c)}{\Longrightarrow} cadc\ (bd) \overset{(d)}{\Longrightarrow} cadcd\ (ce)$

$$L(\Pi, f_{perm}) = L(\Pi, f_{TRANS}) = \{a\} \cup \{(ab)^n a (cd)^n \mid n \geq 1\} \cup \{(ab)^n a (fg)^n \mid n \geq 1\}$$

Thus:

L= $\{a\} \cup \{(ab)^n a(cd)^n \mid n \geq 1\} \cup \{(ab)^n a(fg)^n \mid n \geq 1\}$ is an **LL(1) P colony automata** language, although it is **not** generated by any context-free **LL(*k*) grammar** for any *k*.

We can state:

There are CF languages in $\mathcal{L}_X(genPCol, LL(1))$, $X \in \{perm, TRANS\}$ which are not in $\mathcal{L}(CF, LL(k))$ for any $k \geq 1$.

Thank you.

- The work of **E. Csuhaj-Varjú** was supported in part by the National Research, Development and Innovation Office of Hungary, NKFIH, grant no. K 120558.
- The work of **K. Kántor** and **Gy. Vaszil** was supported in part by the National Research, Development and Innovation Office of Hungary, NKFIH, grant no. K 120558 and also by the construction EFOP-3.6.3-VEKOP-16-2017-00002, a project financed by the European Union, co-financed by the European Social Fund.