P Colony Automata with LL(k) conditions

Erzsébet Csuhaj-Varjú[1], Kristóf Kántor[2], György Vaszil[2]

[1]Eötvös Loránd University, Budapest

[2]University of Debrecen

- LL(k) grammars

- P colonies,
  P colony automata,
  generalized P colony automata

- LL(k) P colony automata

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#	
$S_1aB_3aB_1bB_1b\#$	

- An LL(1) grammar:

S –> aB
B –> b | aBb

	aabb#
	S#

→

	aabb#
S_1	aB#

→

a	abb#
S_1a	B#

→

a	abb#
S_1aB_2	aBb#

→

aa	bb#
S_1aB_2a	Bb#

→

aa	bb#
$S_1aB_2aB_1$	bb#

→

aab	b#
$S_1aB_2aB_1b$	b#

→

aabb	#
$S_1aB_2aB_1bb$	#

→

aabb#	
$S_1aB_2aB_1bb\#$	

As we have seen

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

It is also known:

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

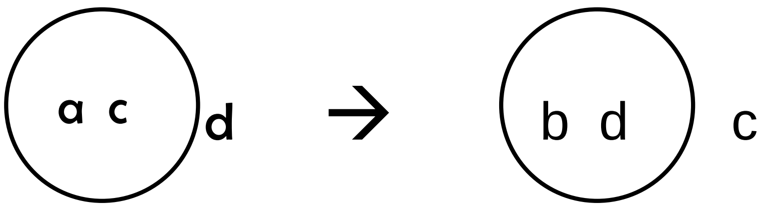
P colonies

- A population of very **simple cells** 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**

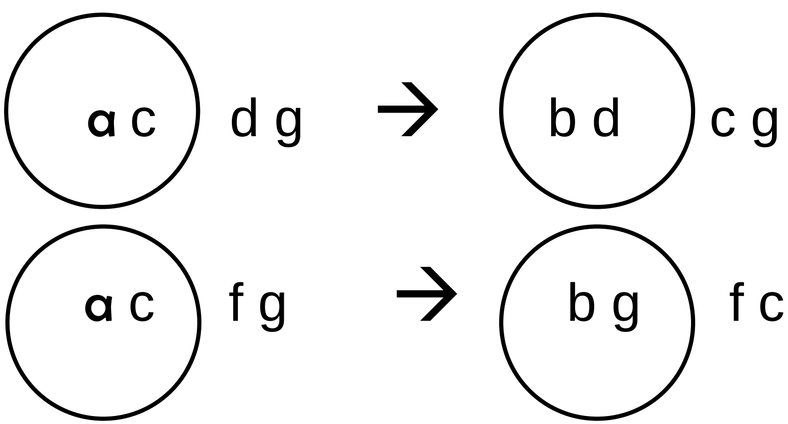
[Kelemen, Kelemenová, Paun 2004]

P colonies



rewriting + communication

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



rewriting + checking communication

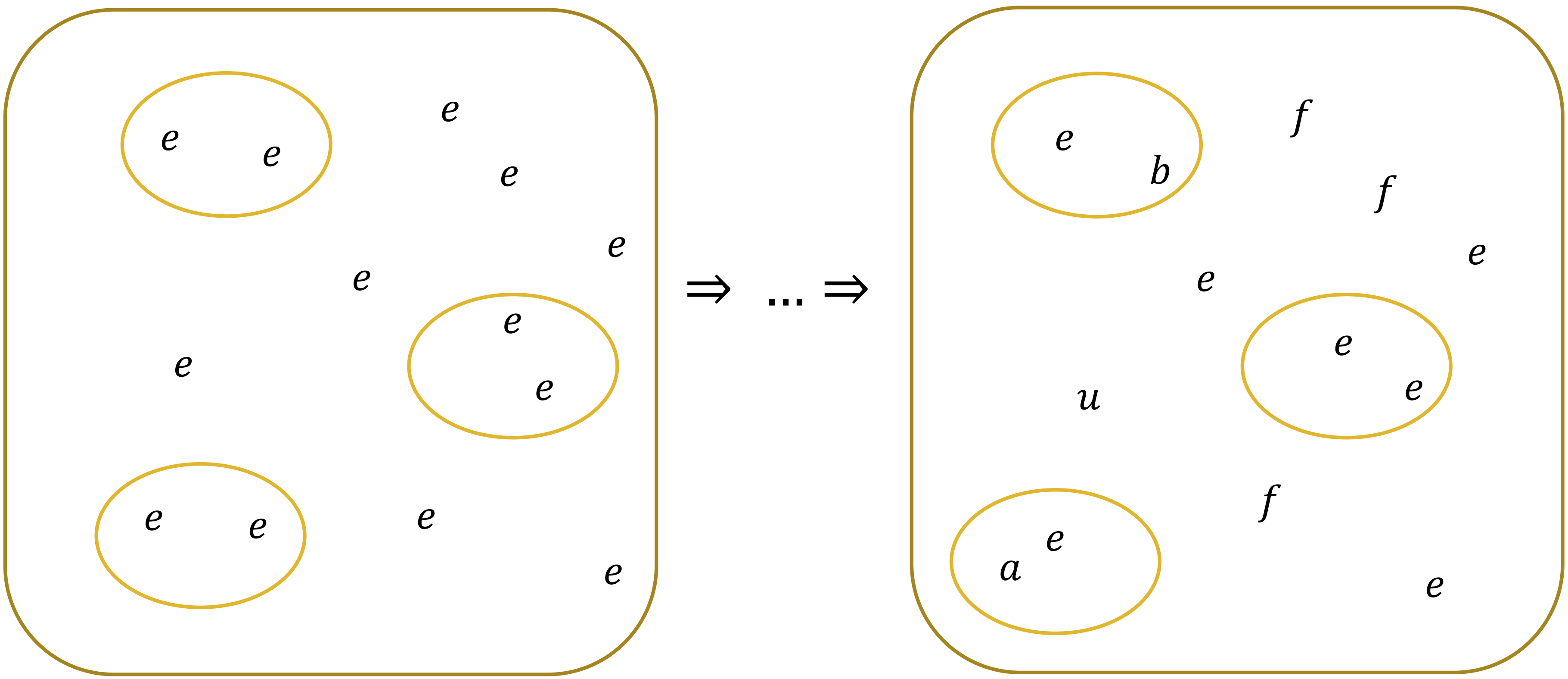
$$(a \to b, c \leftrightarrow d / c \leftrightarrow g)$$

The computation

- Start in an **initial configuration**
- Apply (a maximal set of) programs in **parallel** in the cells, **halt** if no program is applicable
- The **result** is the **number** of the **multiplicities** of certain objects found in the **environment**

The computation

initial configuration ⇒ … ⇒ a possible result



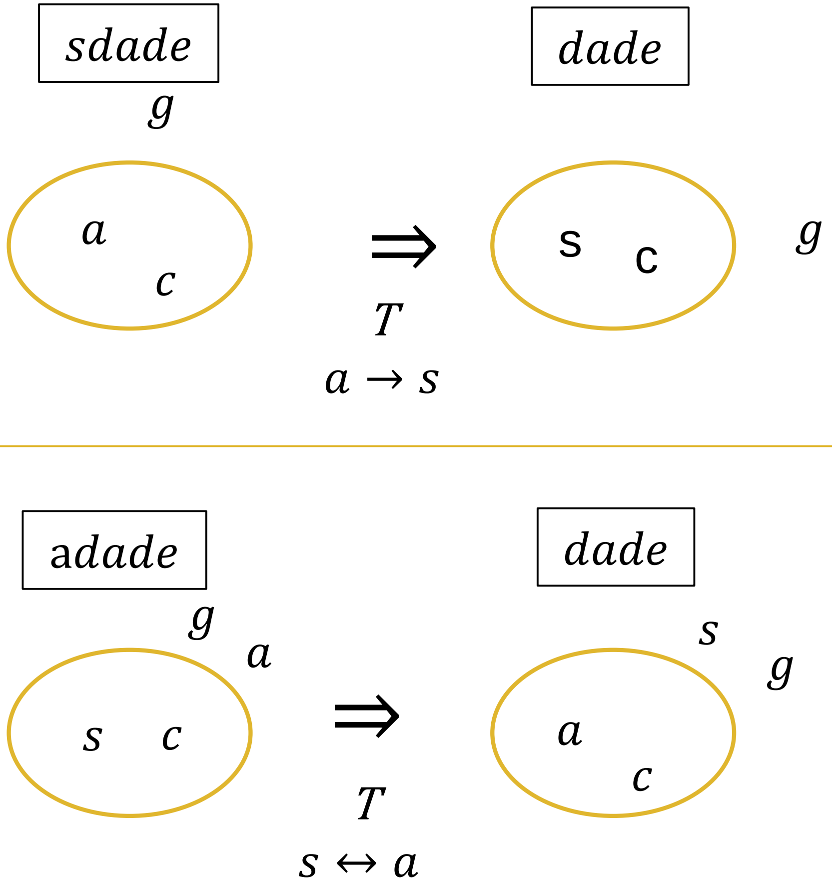
P colony automata

- Response to the **changes in the environment**
- **Automata-like** behavior - an **input string** is given
- **Tape rules** and **non-tape rules**: the application of programs with tape rules **reads a symbol** of the input

[Ciencialová, Cienciala, Csuhaj-Varjú, Kelemenová, Vaszil 2010]

P colony automata

The effect of tape rules:



Different computation modes ...

- **nt, ntmax, ntmin**: any recursively enumerable language can be accepted/characterized

[Ciencialová, Cienciala, Csuhaj-Varjú, Kelemenová, Vaszil 2010]

- **t, one cell**: only CS languages can be generated

[Cienciala, Ciencialová 2011a]

- **initial:** any recursively enumerable language can be characterized

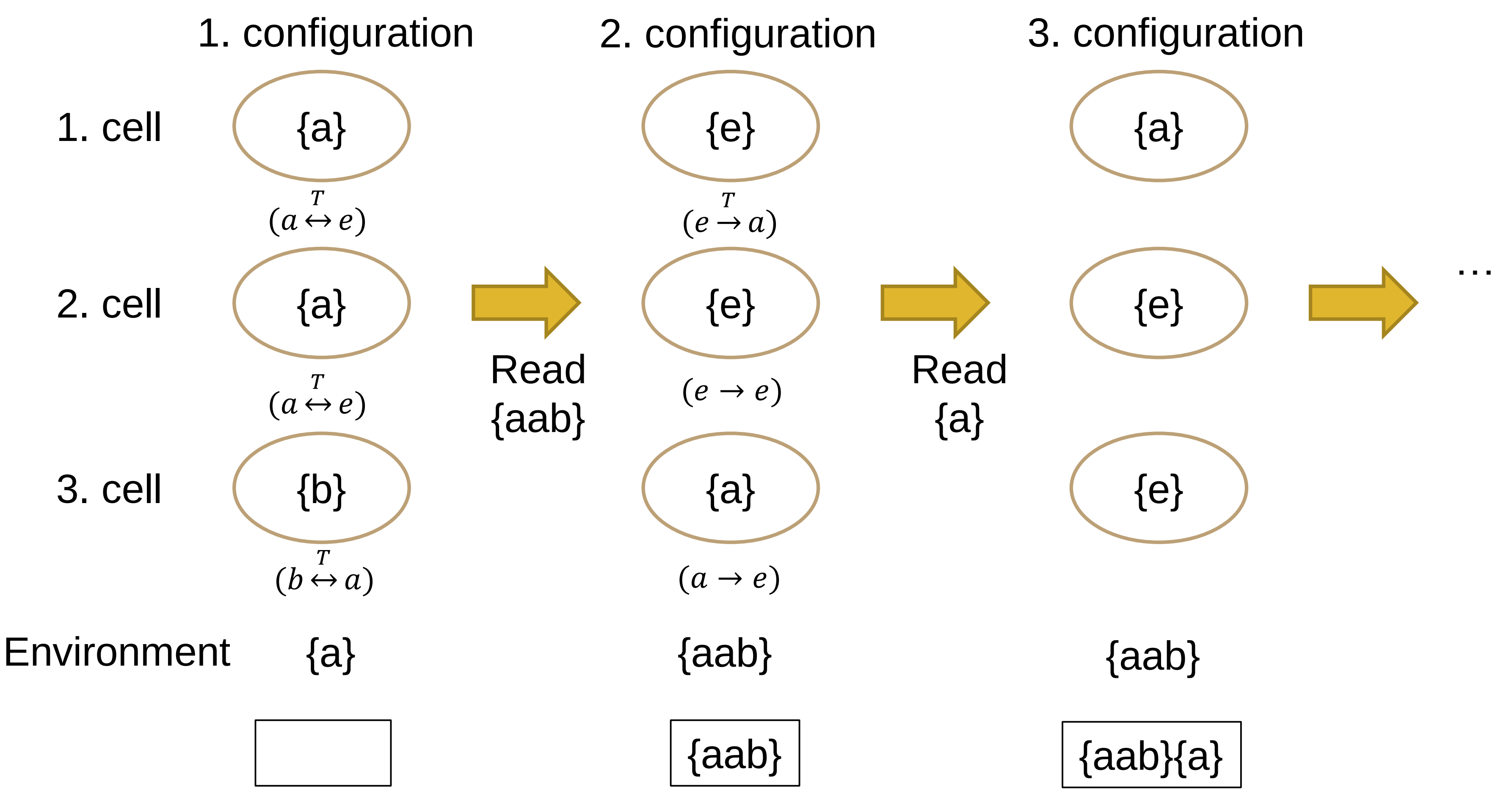
[Cienciala, Ciencialová 2011b]

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 in one step**

[Kántor, Vaszil 2014]

Computation and rules – small example



Generalized P colony automata

- A **maximal parallel set** of programs is chosen, tape rules and non-tape rules together
- The chosen tape rules might “read” several **different symbols in one step**

- **Three** modes:
  - **all-tape**: all programs contain **at least one** tape rule
  - **com-tape**: all **communication** rules are tape rules
  - **no restriction**

Example

$$\Pi = (\{a,b,c\}, e, \emptyset, (ea, P), F)$$

$P =$

$\langle e \to a, a \overset{T}{\leftrightarrow} e \rangle$	$\langle e \to b, a \overset{T}{\leftrightarrow} e \rangle$	$\langle e \to b, b \overset{T}{\leftrightarrow} a \rangle$
$\langle e \to c, b \overset{T}{\leftrightarrow} a \rangle$	$\langle a \to b, b \overset{T}{\leftrightarrow} a \rangle$	$\langle a \to c, b \overset{T}{\leftrightarrow} a \rangle$

$F =$ $\{(v, ca) \mid a \notin v\}$

Possible computation:

$$(\ , ea) \overset{(a)}{\Longrightarrow} (a, ea) \overset{(a)}{\Longrightarrow} (aa, ea) \overset{(a)}{\Longrightarrow} (aaa, eb) \overset{(b)}{\Longrightarrow} (aab, ba)$$

$$\overset{(b)}{\Longrightarrow} (bba, ab) \overset{(b)}{\Longrightarrow} (bbb, ca)$$

$$A(\Pi) = \{(a)^n (b)^n \mid n \geq 0\}$$

$$L(\Pi, f_{perm}) = \{a^n b^n \mid n \geq \mathbf{0}\}$$

Some results ...

- $\mathcal{L}_{perm}(genPCol, *(1)) = \mathcal{L}(\mathrm{RE})$.
- $\mathcal{L}_{perm}(genPCol, X(1)) \setminus \mathcal{L}(REG) \neq \emptyset$ for $X \in \{all\text{-}tape, com\text{-}tape\}$.
- $\mathcal{L}_{perm}(genPCol, all\text{-}tape(k)) = \mathcal{L}(RE)$ for $k \geq 2$.

[K. Kántor, Gy. Vaszil, *to appear*.]

LL(k) P colony automata

Informal definition:

The next *k* symbols of the not-yet-processed part of the input string determines the cell and the program to be applied in the next computational step.

As we have seen:

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

Similarly,

$$\{(ab)^n (cd)^n \mid n>1\} \cup \{(ab)^n (fg)^n \mid n>1\}$$

is **not** an LL(k) language for any k.

Example with 1 symbol lookahead

$\boldsymbol{P} =$

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

$\boldsymbol{F} =$

$\{(v, ce) \mid b \notin v\} \cup \{(v, fe) \mid b \notin v\}$

Possible computation:

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

$\Longrightarrow (bb, ce) \overset{(c)}{\Longrightarrow} (bc, bd) \overset{(d)}{\Longrightarrow} (bcd, ce) \overset{(c)}{\Longrightarrow} (cdc, bd) \overset{(d)}{\Longrightarrow} (cdcd, ce)$

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

Thus:

L= $\{(ab)^n(cd)^n \mid n>1\} \cup \{(ab)^n(fg)^n \mid n>1\}$

is **not** generated by an **LL(k) grammar** for any k, but

L is an **LL(1) P colony automata** language

Acknowledgment

Research supported in part

- by project no. K 120558, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the K 16 funding scheme, and
- by the construction EFOP-3.6.3-VEKOP-16, a project supported by the European Union,
co-financed by the European Social Fund.