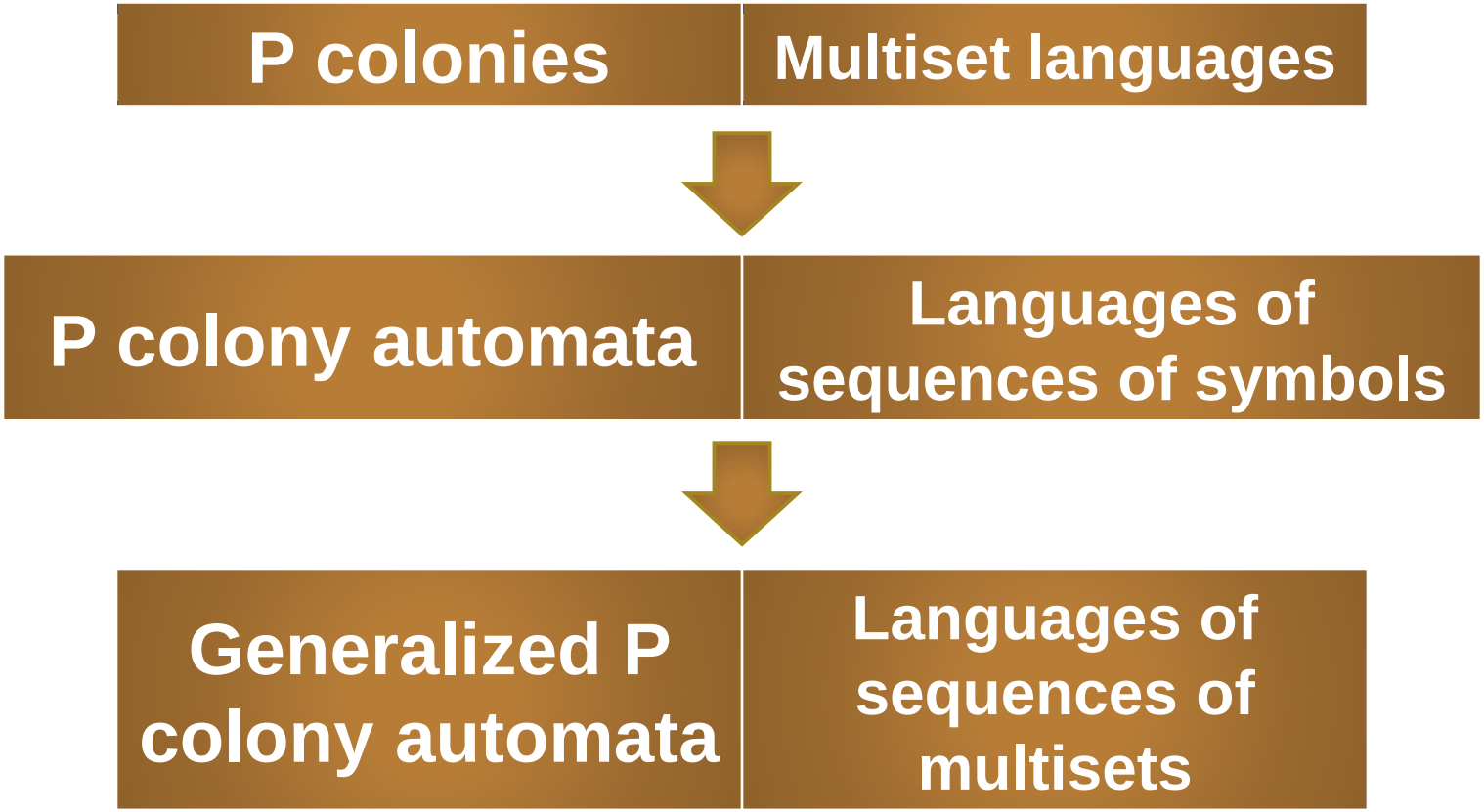
Variants of P Colony Automata

Kristóf Kántor, **György Vaszil**
University of Debrecen

Overview

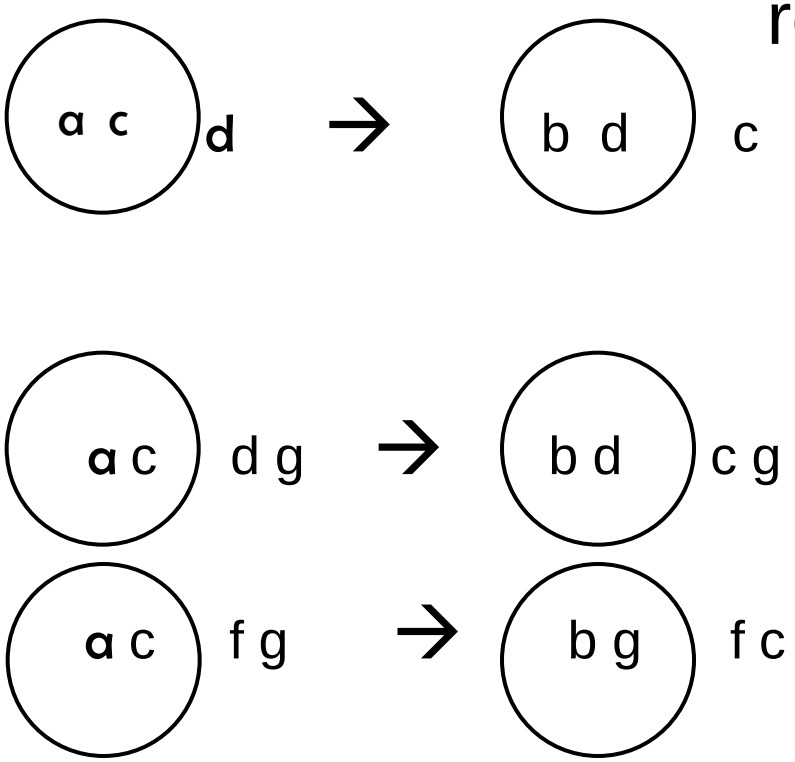

P colonies
Multiset languages
P colony automata
Languages of sequences of symbols
Generalized P colony automata
Languages of sequences of multisets

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

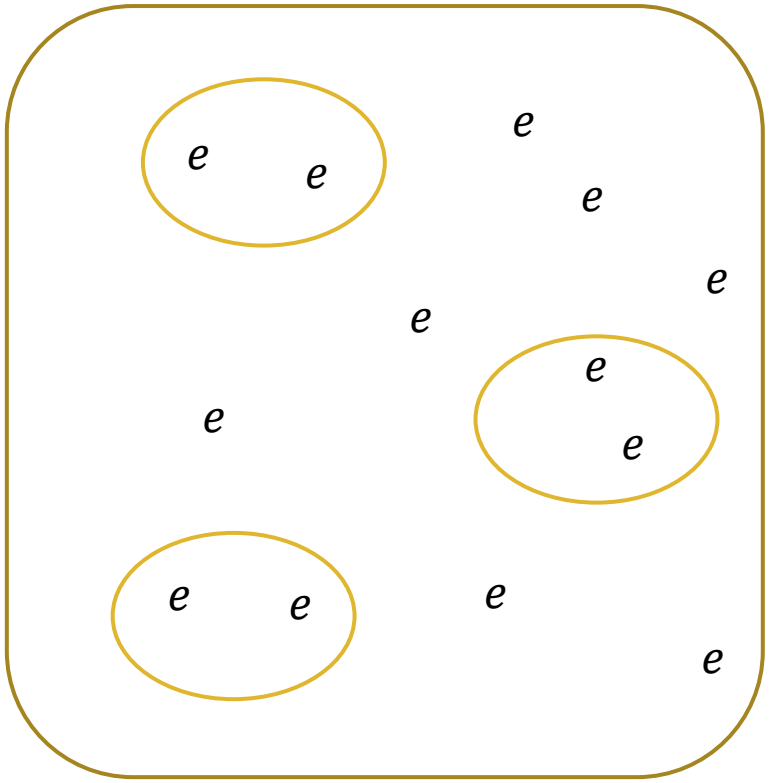
$$(a \to b, c \leftrightarrow d / c \leftrightarrow g)$$

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**

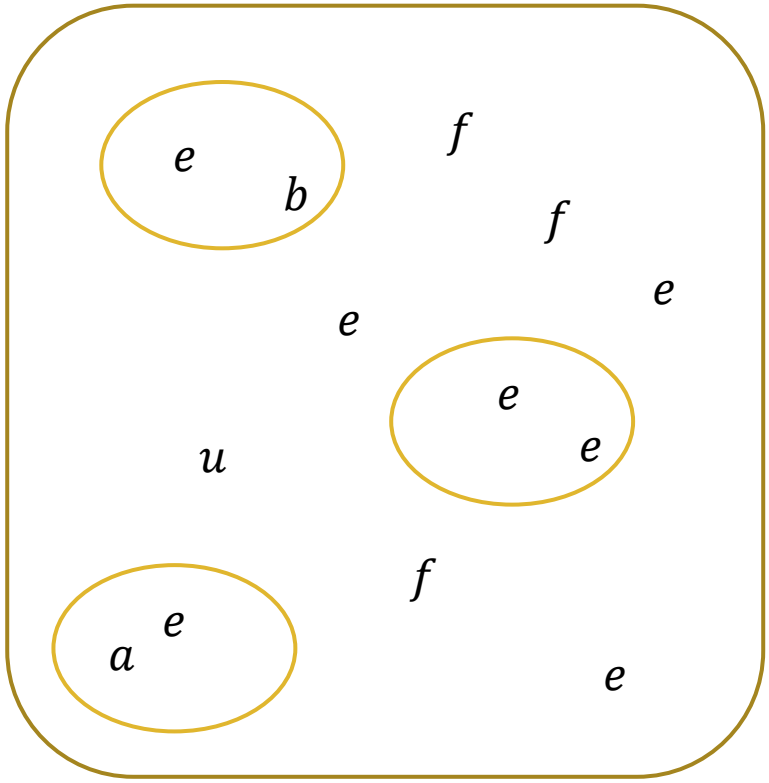
The computation

initial configuration



$\Rightarrow \ldots \Rightarrow$

a possible result



Computational power

- P colonies with **two object cells** and **checking** rules generate **any** computable set of numbers with
  - at most **4 programs** in one cell, the number of **cells unbounded**
  - **one cell**, the number of **programs unbounded**
- P colonies with **two object cells** and **no checking** rules need **8 components**
- P colonies with **3 object cells** need
  - at most **3 programs** in one cell with **checking** rules
  - **7 programs** with **no checking** rules

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

Simplifying the cells even more

P colonies with **one object cells**, programs of the form $(a \rightarrow b)$, $(a \leftrightarrow b)$ or $(a \leftrightarrow b / a \leftrightarrow c)$.

- **One object** P colonies with **checking** rules generate **any** set of numbers with **4 cells.**

  [Cienciala, Ciencialova, Kelemenova 2007]

- With **no checking** rules **one object** P colonies generate **any** set of numbers with **6 cells.**

[Ciencielová, Csuhaj Varjú, Kelemenová, Vaszil 2009]

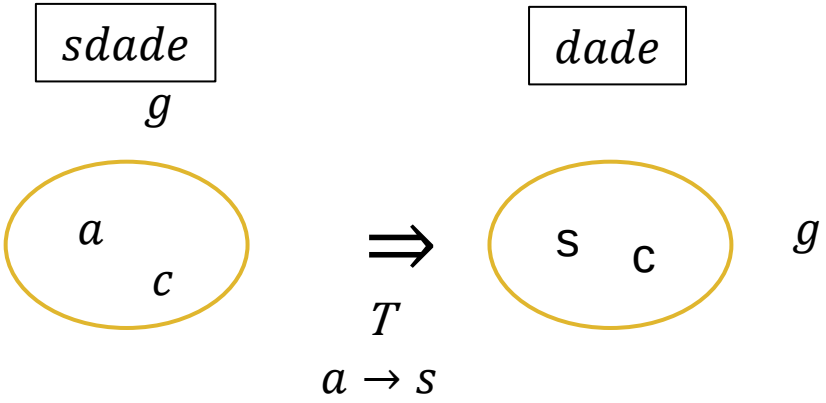
P colony automata

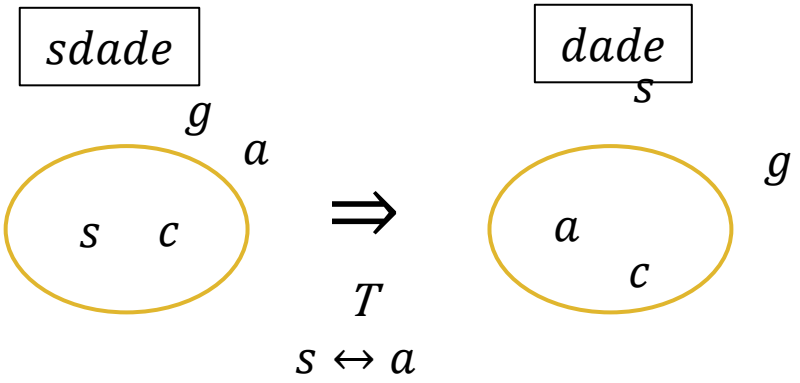
- Response to the **changes in the environment**
- **Automata-like** behavior - an **input string** is given
- **Tape rules** and **non-tape rules**: the application of programs with tape rules **reads a symbol** of the input

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

P colony automata

The effect of tape rules:





Different computational modes...

…with different uses of the tape rules:

- *t-transition*, denoted by $\Rightarrow_t$, if $u' = u$ and P_c is maximal set of programs with respect to the property that every $p \in P_c$ is a tape program with $read(p) = a$;
- *tmin-transition*, denoted by $\Rightarrow_{tmin}$, if $u' = u$ and P_c is maximal set of programs with at least one $p \in P_c$, such that p is a tape program with $read(p) = a$;
- *tmax-transition*, denoted as $\Rightarrow_{tmax}$, if $u' = u$ and $P_c = P_T \cup P_N$ where P_T is a maximal set of applicable tape programs with $read(p) = a$ for all $p \in P_T$, the set P_N is a set of nontape programs, and $P_c = P_T \cup P_N$ is maximal;
- *n-transition*, denoted by $\Rightarrow_n$, if $u' = au$ and P_c is maximal set of nontape programs.

Power of the different 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]

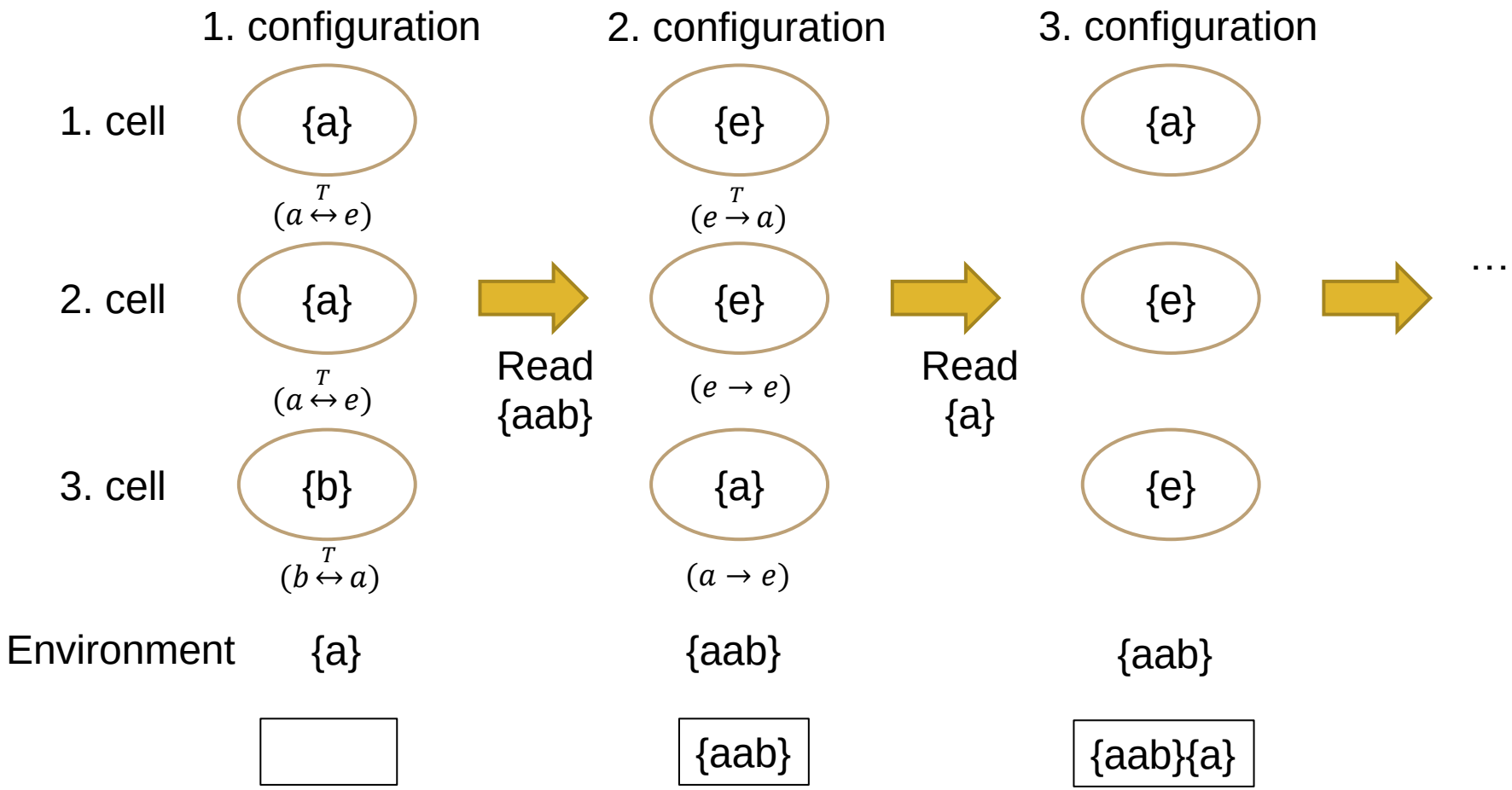
Common in all modes…

- …that the **tape rules** must read the **same symbol**, even when **more than one** tape rules are applied in **one** computational **step.**

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**

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**

[Kántor, Vaszil 2014]

Generalized P colony automata

$$\Pi = (V, e, w_E\ (w_1, P_1), \ldots, (w_n, P_n), F)$$
$$with\ capacity\ k\ (k, n \geq 1)$$

- V: alphabet, its elements = objects
- $e \in V$: environmental object
- $w_E \in (V - \{e\})^*$: initial environmental contents different from e
- $(w_i, P_i), 1 \leq i \leq n$: the i-th cell
  - $|w_i| = k$: cell contents
  - $|P_i| = k$: set of programs
- F: set of accepting configurations

Accepted input sequence – accepted language

- The **set of input sequences accepted**: The set of the sequences of read multisets
  - $\{u_1 u_2 \dots u_s \mid u_i \in (V - \{e\})^*, 1 \leq i \leq s,$
    $and\ there\ exists\ a\ configuration\ sequence\ c_0, \dots, c_s,$
    $with\ c_0 = (w_E, w_1, \dots, w_n), c_s \in F\ , and\ c_i$

$$\Rightarrow c_{i+1}\ with \bigcup_{p \in P_{c_i}} read(p) = u_{i+1}\ for\ all\ 0 \leq i \leq s-1\}$$

- The **language accepted** with respect to a mapping $(f: (V - \{e\})^* \rightarrow 2^{\Sigma^*})$:

$$\mathcal{L}(\Pi, f)$$
$$= \{f(u_1) \cdot f(u_2) \cdot \dots \cdot f(u_s) \in \Sigma^* \mid u_1 \dots u_s \text{ is an accepted input sequence}\}$$

Special mapping - f_{perm}

- $f_{perm}: (V - \{e\})^* \rightarrow 2^{(V-\{e\})^*}$, where $f(x) = \{y \in (V - \{e\})^* \mid y = perm(x)\}$

- For example:
  - $f_{perm}\ (ab) = \{ab, ba\}$
  - $f_{perm}\ (a) = \{a\}$
  - $f_{perm}\ (cca) = \{acc, cac, cca\}$
  - $f_{perm}\ (bb) = \{bb\}$

Example

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

$P =$

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

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

Possible computation:

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

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

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

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

Classes of Languages

- $\mathcal{L}_{\mathcal{F}}(genPCol, com - tape(k))$ is the class of languages accepted by GenPCol automata with capacity k and with mappings from the class $\mathcal{F}$ where all the communication rules are tape rules

- $\mathcal{L}_{\mathcal{F}}(genPCol, all - tape(k))$ – all programs must have at least one tape rule

- $\mathcal{L}_{\mathcal{F}}(genPCol, * (k))$ – programs do not have restrictions

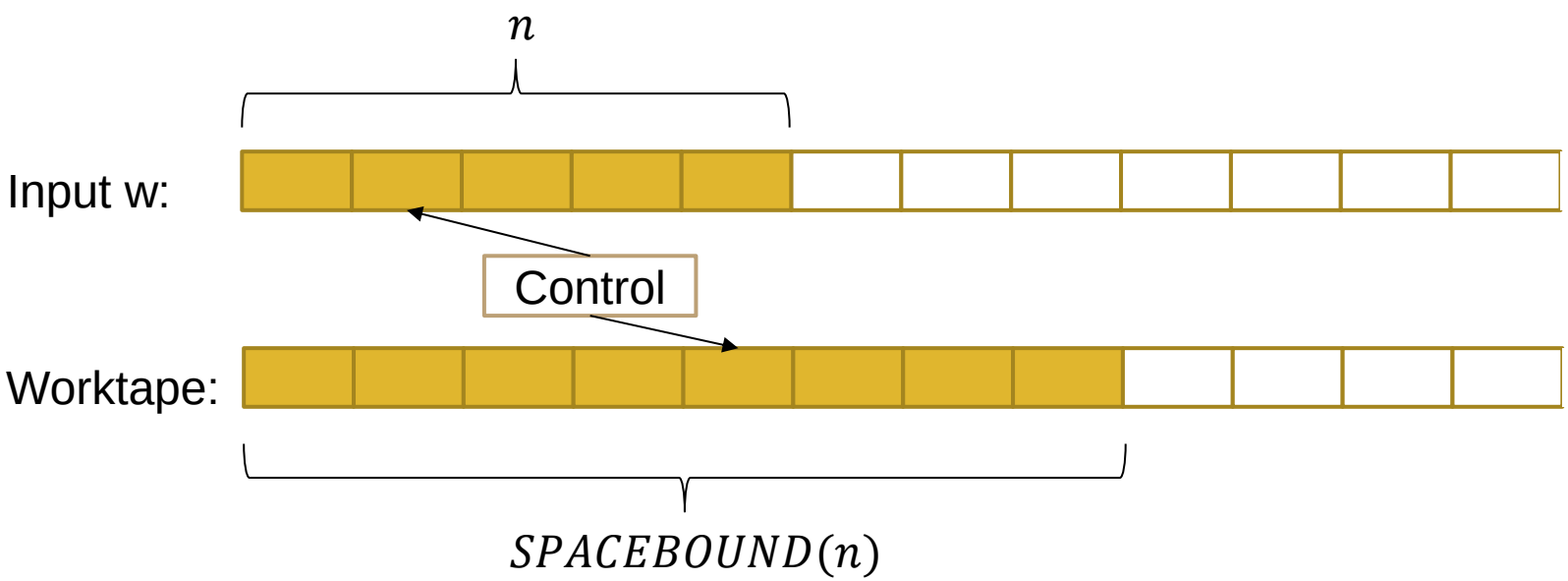
Results

- $\mathcal{L}_{perm}(genPCol,*\,(1)) = \mathcal{L}(RE)$
- $\mathcal{L}_{perm}\big(genPCol, X(1)\big)\backslash\mathcal{L}(REG) \neq \emptyset\ for\ X \in \{all - tape, com - tape\}$
- $\mathcal{L}_{perm}\big(genPCol, com - tape(k)\big) \subseteq r - 1LOGSPACE\ for\ any\ k \geq 1$
- $\mathcal{L}_{perm}\big(genPCol, com - tape(2)\big)\backslash\mathcal{L}\big(PA, f_{perm}\big) \neq \emptyset$

K. Kántor, Gy. Vaszil: On the Classes of Languages Characterized by Generalized P Colony Automata. Theoretical Computer Science, accepted.

Space bounded computation

- Turing Machine, one way input tape, worktapes
- Number of nonempty cells on worktapes: bounded by $S(n)$
- n: the length of the input

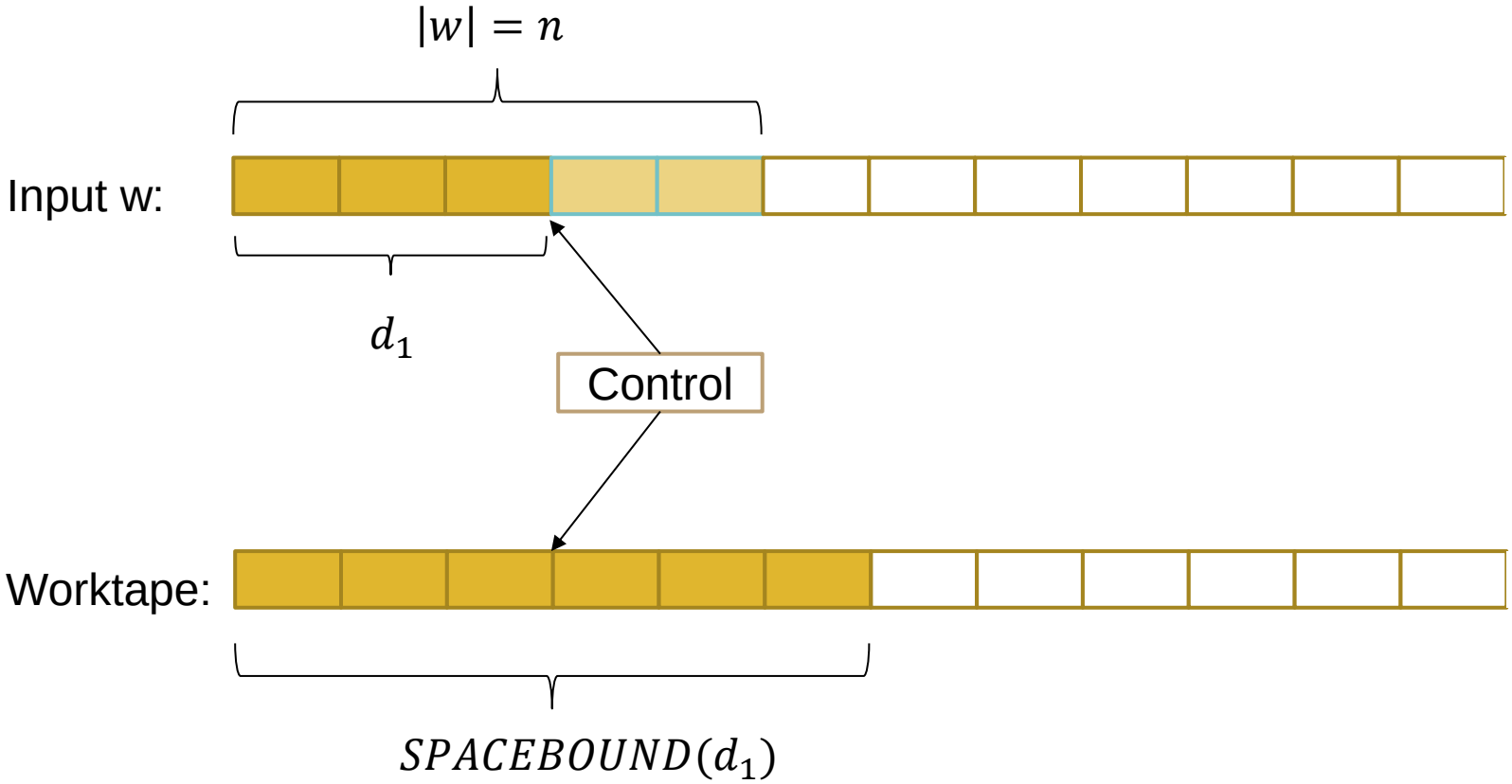


”Restricted” space bounded computation

- Turing Machine, one way input tape, worktapes
- Number of nonempty cells on worktapes: bounded by $S(d)$
- d: Current distance of the reading head from the left end of the input tape

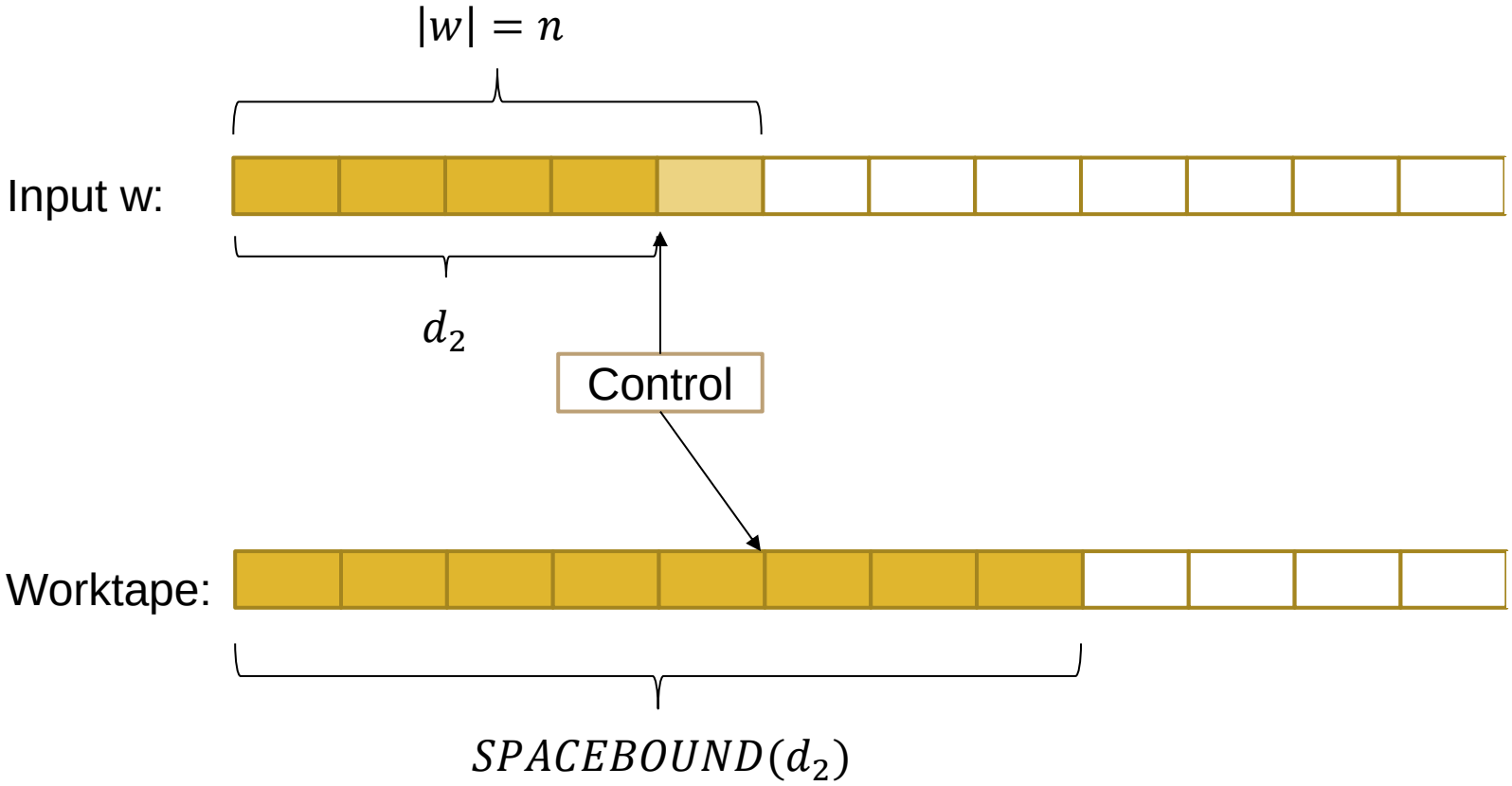
”Restricted” space bounded computation

- After reading d_1 input cells:



”Restricted” space bounded computation

- After reading d_2 input cells:



(About P Automata)

- $r-1LOGSPACE \subset 1LOGSPACE$
- $\mathcal{L}_{seq}\,(PA, f) = r-1LOGSPACE\ for\ any\ nonereasing\ f$
- $\mathcal{L}\left(PA, f_{perm}\right) \subset r-1LOGSPACE$

E. Csuhaj-Varjú, Gy. Vaszil, P automata with restricted power International Journal of Foundations of Computer Science 25(4) 391-408, 2014

E. Csuhaj-Varjú, O.H. Ibarra, Gy. Vaszil, On the computational complexity of P automata, in: C. Ferretti, G. Mauri, C. Zandron, eds., *10th International Workshop on DNA Computing, LNCS 3384* (Springer Berlin Heidelberg, 2005) 76-89.

Results

- $\mathcal{L}_{perm}(genPCol,*\,(1)) = \mathcal{L}(RE)$
- $\mathcal{L}_{perm}\big(genPCol, X(1)\big)\backslash\mathcal{L}(REG) \neq \emptyset\ for\ X \in \{all - tape, com - tape\}$
- $\mathcal{L}_{perm}\big(genPCol, com - tape(k)\big) \subseteq r - 1LOGSPACE\ for\ any\ k \geq 1$
- $\mathcal{L}_{perm}\big(genPCol, com - tape(2)\big)\backslash\mathcal{L}\big(PA, f_{perm}\big) \neq \emptyset$

K. Kántor, Gy. Vaszil: On the Classes of Languages Characterized by Generalized P Colony Automata. Theoretical Computer Science, accepted.

What about other types of input functions?

- Instead of the f_{perm} permutation mapping, consider mappings defined by a finite transducer, such that:
  - Nonerasing
  - $f(v) = \{w\}$, all string representations of $v \in V^*$ as input should result in the same $w \in \Sigma^*$

Notation: $f \in TRANS$

Example

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

$P =$

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

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

Possible computation:

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

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

$$A(\Pi) = \{(a)^n (b)^n | n \geq 1\}$$

$$L(\Pi, f_1) = \{(cd)^n (ef)^n | n \geq 1\}, f_1 \in TRANS, with$$
$$f_1 : \{a, b\}^* \to \{c, d, e, f\}^*, f_1(a) = \{cd\}, f_1(b) = \{ef\}$$

Initial results

- $\mathcal{L}_{TRANS}\big(genPCol,*\,(1)\big) = \mathcal{L}(RE)$
- $\mathcal{L}_{TRANS}\big(genPCol, all-tape(k)\big) = \mathcal{L}(RE)\ for\ k \geq 2$
- $REG \subseteq \mathcal{L}_{TRANS}\big(genPCol, X(1)\big),$

$$for\ X \in \{all-tape, com-tape\}$$

- $\mathcal{L}_{TRANS}\big(genPCol, com-tape(2)\big) \subseteq r-1LOGSPACE$

Thank you for your attention!

P colonies

- **Simple** building blocks (**cells**) in a **shared environment** working with multisets to achieve **complex behavior**
- A **maximal parallel set** of programs is applied

P colony automata

- **Simple** building blocks (**cells**) in a **shared environment** working with multisets to achieve **complex behavior**
- A **maximal parallel set** of programs is chosen
- Tape rules which might “read” **one** symbol **in one computational step**

Generalized P colony automata

- **Simple** building blocks (**cells**) in a **shared environment** working with multisets to achieve **complex behavior**
- A **maximal parallel set** of programs is chosen
- Tape rules which might “read” **several** different symbols (a multiset) **in one computational step**

What is a program?

- They contain rules: $(a, b \in V)$
  - Rewriting tape rules

$$a \overset{T}{\rightarrow} b$$

  - Communication tape rules

$$a \overset{T}{\leftrightarrow} b$$

  - Rewriting nontape rules

$$a \rightarrow b$$

  - Communication nontape rules

$$a \leftrightarrow b$$