P Colony Automata

An overview of recent and not-so-recent results, and some open problems

György Vaszil

University of Debrecen

- **Multisets: collection** of objects/symbols, **multiplicities**
- **Complex behavior:** computational completeness, universality
- **Simple building blocks:** simple symbol processing **agents** in a **shared environment** (multiset) which they modify

Emergent behavior

The **“whole”** is **more** than the **sum of** its **“parts”.**

Outline

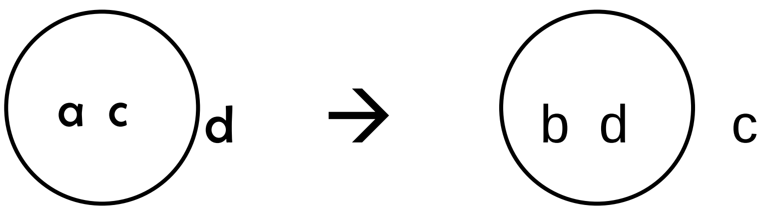
- P colonies
  - structure, functioning, computational power, **multiset languages**
- P colony automata
  - languages of **strings of symbols**
- Generalized P colony automata
  - languages of **strings/sequences of multisets**

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**

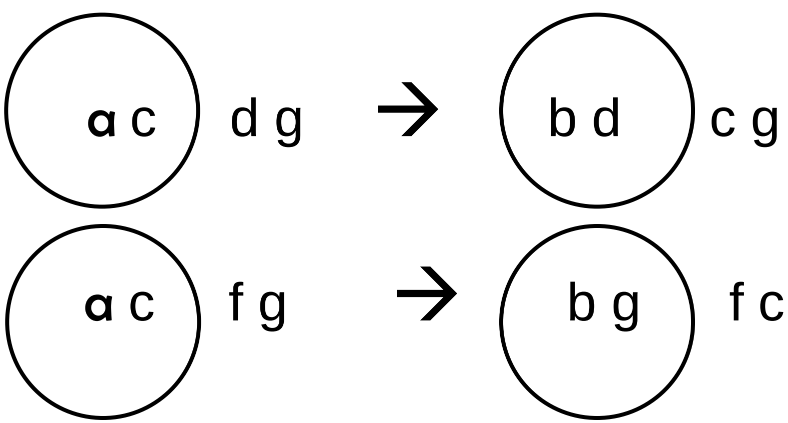
[Kelemen, Kelemenova, Paun 2004]

P colonies



rewriting + communication

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



rewriting + checking communication

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

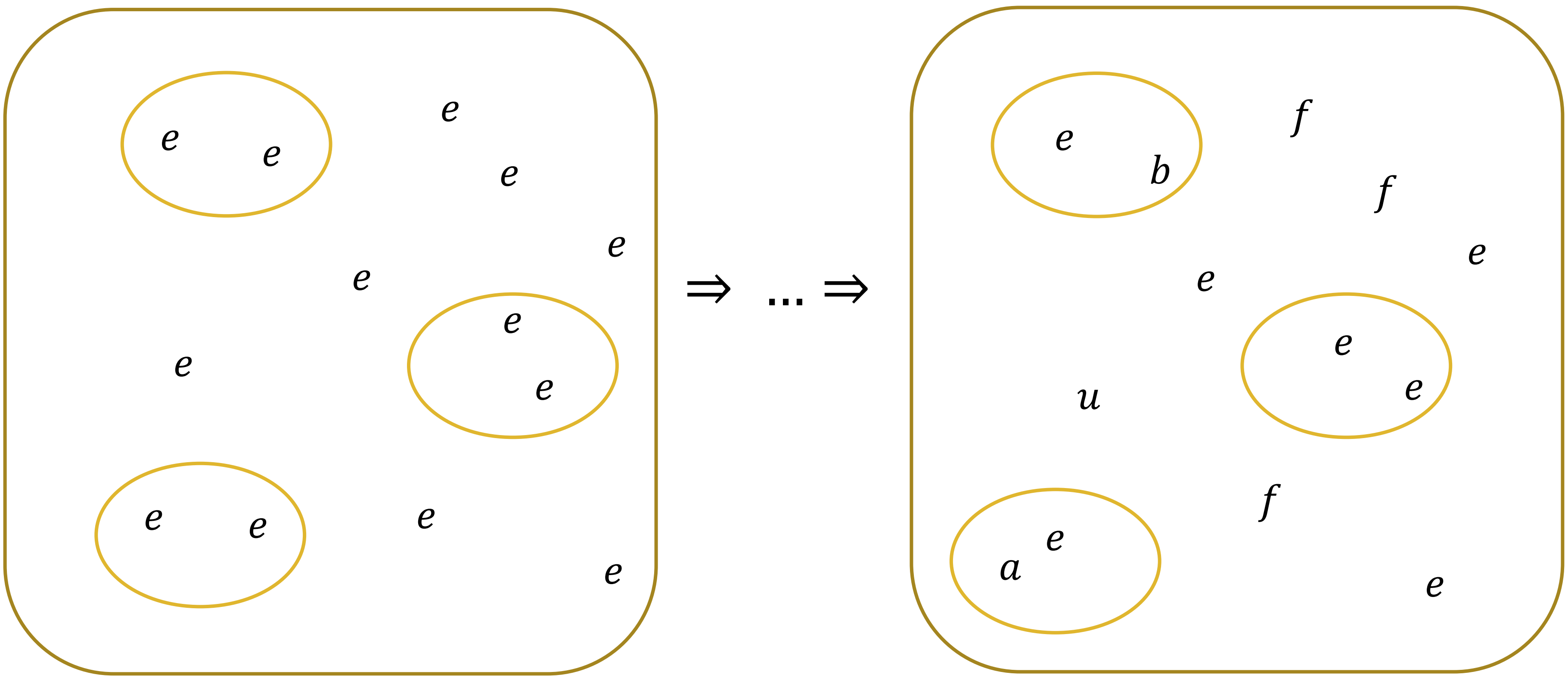
The computation

- Start in an **initial configuration**
- Apply the 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

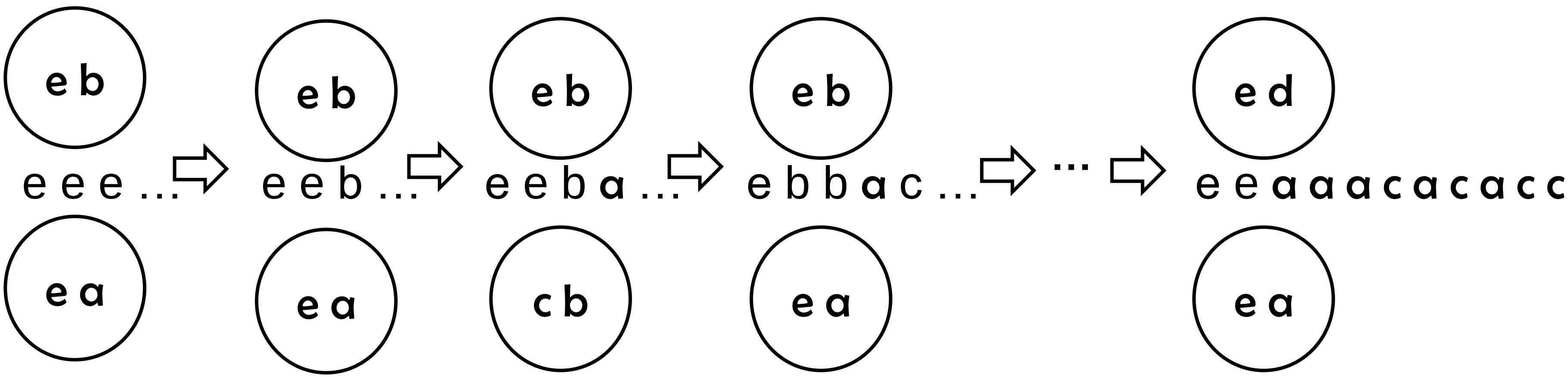
initial configuration

a possible result



The computation

$(e \to b, b \leftrightarrow e)$
$(e \to d, b \leftrightarrow e)$



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

We obtain $a^n c^n, n \geq 1$ in the environment.

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-Varju, Kelemen, Kelemenova, Paun, Vaszil 2006a]

Simplifying the cells even more

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

[Ciencielova, Csuhaj Varju, Kelemenova, Vaszil 2009]

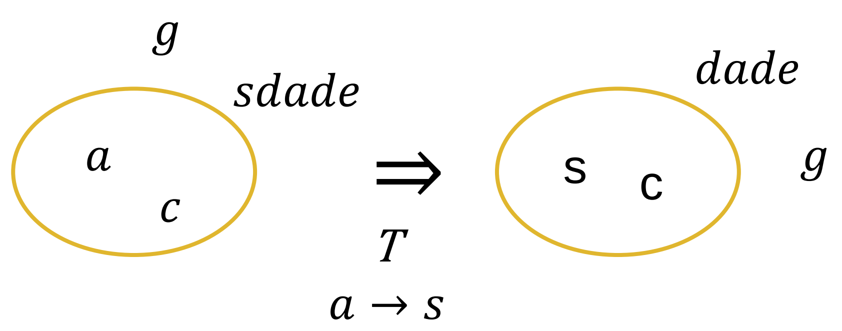
P colony automata

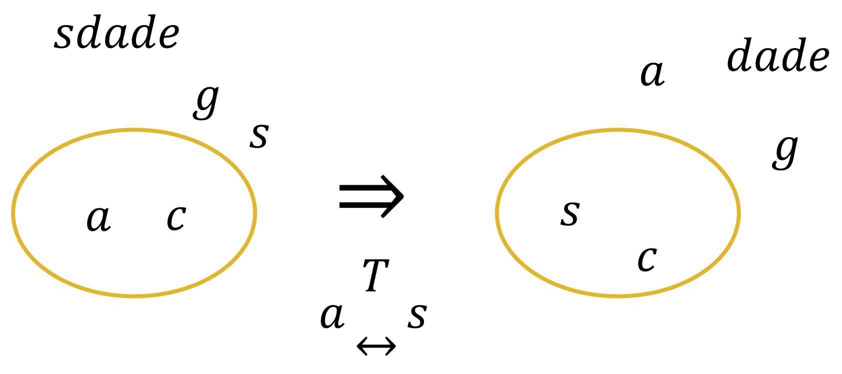
- Response to the **changes in the environment**
- **Automata-like** behavior - an **input string** is given
- **Tape rules** and **non-tape rules**: the application of programs with tape rules **reads a symbol** of the input

[Ciencialova, Cienciala, Csuhaj-Varju, Kelemenova, Vaszil 2010]

P colony automata

The effect of tape rules:





Power of the different modes

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

[Ciencialova, Cienciala, Csuhaj-Varju, Kelemenova, Vaszil 2010]

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

[Cienciala, Ciencialova 2011a]

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

[Cienciala, Ciencialova 2011b]

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.

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 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**, a permutation of these have to be the prefix of the input
- **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]

Computational power

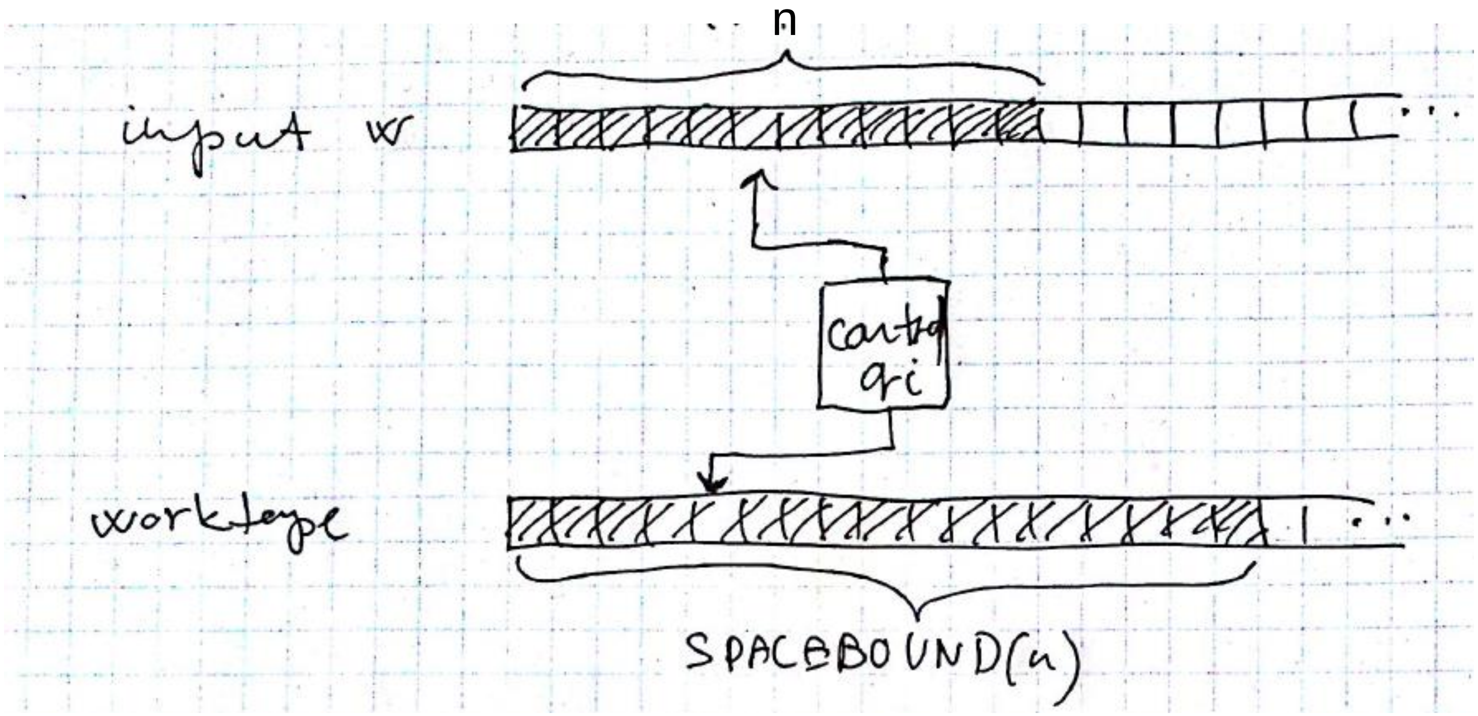
- $\mathcal{L}(GenPCol, com-tape) \cup \mathcal{L}(GenPCol, all-tape) \subseteq \mathcal{L}(GenPCol,*)$
- $\mathcal{L}(GenPCol, com-tape) \cap \mathcal{L}(GenPCol, all-tape) - \mathcal{L}(CF) \neq \emptyset$
- $\mathcal{L}(GenPCol, all-tape) \cup \mathcal{L}(GenPCol, com-tape) \subseteq$ *r-1LOGSPACE*
- $\mathcal{L}(GenPCol,*) = RE$.

Turing machines with restricted space bound

A nondetermininstic Turing machine with a **one-way** input tape is **restricted** $S(n)$ **space bounded** if the number of **nonempty cells** on the worktape(s) is **bounded by** $S(d)$, where d is the **distance of the reading head** from the left-end of the one-way input tape.

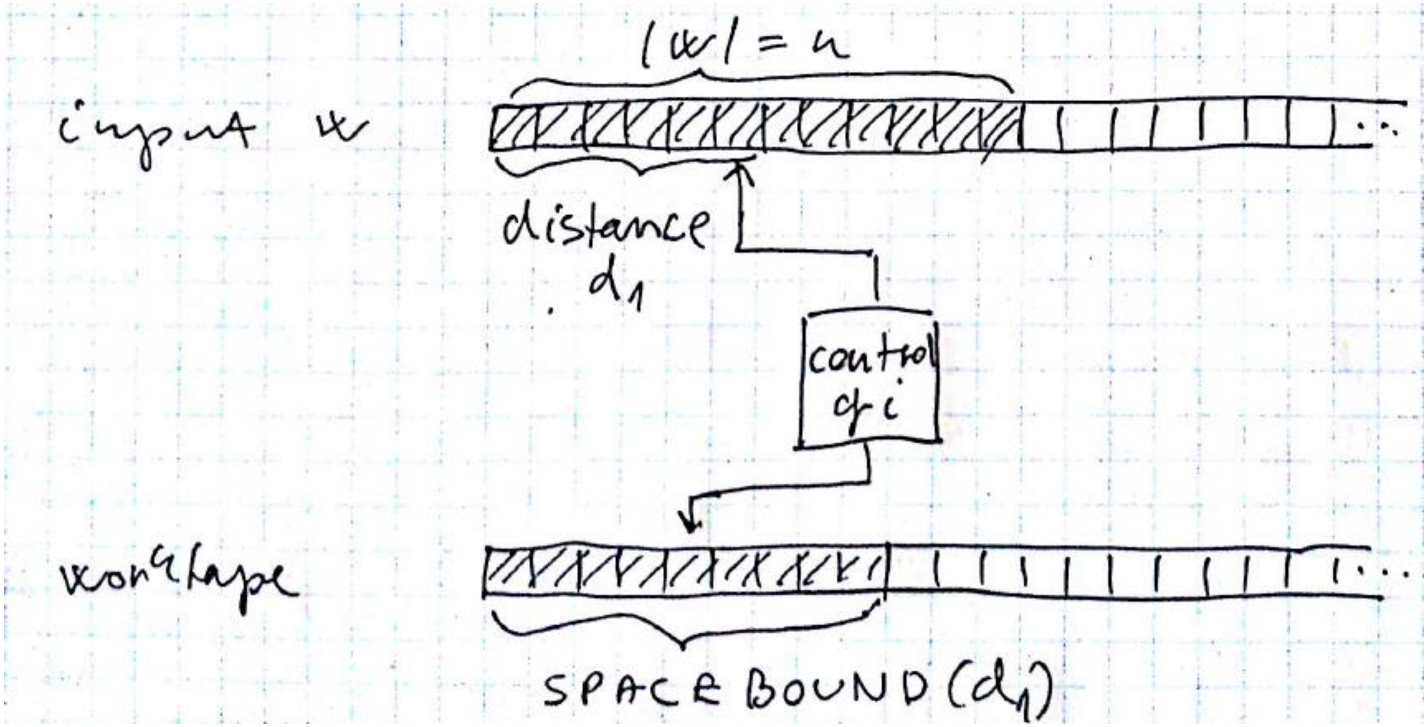
A Turing machine with SPACEBOUND(n)

The length of the available worktape is bounded by the length of the input:



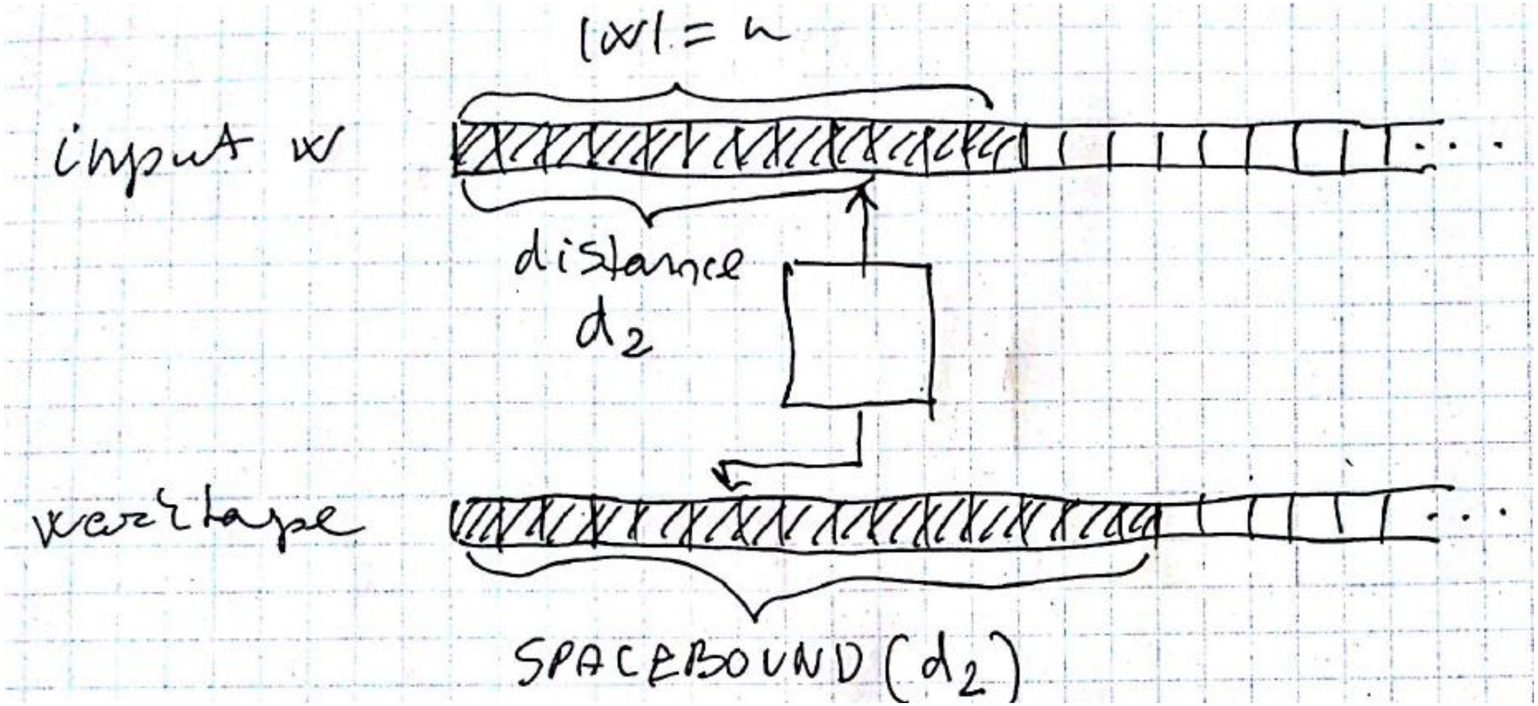
Turing machines with *restricted* space bound

1. After reading d_1 input cells:



Turing machines with *restricted* space bound

2. After reading d_2 input tape cells:



Turing machines with restricted space bound

The **restricted logarithmic space** bound:

- $r1LOGSPACE \subset 1LOGSPACE$

  [Csuhaj-Varju, Ibarra, Vaszil 2004]

- In the **deterministic** case, it is equal to the **strong logarithmic space** bound.

  [Kutrib, Provillard, Vaszil, Wendlandt, 2013]

The **restricted linear space** bound:

- $r1LINSPACE = LINSPACE$

  [Csuhaj-Varju, Ibarra, Vaszil 2004]

Computational power

- $\mathcal{L}(GenPCol, com - tape) \cup \mathcal{L}(GenPCol, all - tape) \subseteq \mathcal{L}(GenPCol,*)$
- $\mathcal{L}(GenPCol, com - tape) \cap \mathcal{L}(GenPCol, all - tape) - \mathcal{L}(CF) \neq \emptyset$
- $\mathcal{L}(GenPCol, all - tape) \cup \mathcal{L}(GenPCol, com - tape) \subseteq$ *r-1LOGSPACE*
- $\mathcal{L}(GenPCol,*) = RE.$

genPCol automata and similar variants of P automata

	all-tape/com-tape	unrestricted
P automata with f_{perm}	$\mathcal{L}(\mathrm{REG}) \subset \cdot \subset$ r-1LOGSPACE	$\mathcal{L}(\mathrm{RE})$
genPCol automata	$\mathcal{L}(\mathrm{REG}) \subset \cdot \subseteq$ r-1LOGSPACE	$\mathcal{L}(\mathrm{RE})$

- Can we obtain more precise results?

Other problems

- The **relationship** of languages characterized by the **all-tape** and **com-tape** modes?
- Are there **other** „interesting” computation **modes**?
- **Map** the input multisets **to strings** in a more **general** way (like in „ordinary” P automata)?

Acknowledgments

- The Hungarian Scientific Research Fund “OTKA” grant no. K75952
- The TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project, co-financed by the European Social Fund