Sublinear-space P systems with active membranes

Antonio E. Porreca Alberto Leporati Giancarlo Mauri
Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca

13th International Conference on Membrane Computing
28–31 August 2012, Budapest, Hungary

The story so far...

Definition

Space complexity for P system is measured as

$$\#\text{membranes} + \#\text{objects}$$

Theorem

P systems working in polynomial (resp., exponential) space have the same computing power as Turing machines working in polynomial (resp., exponential) space $\square$

Conjecture

The same holds for larger space bounds (work in progress)

But what about smaller space bounds?

We have two problems:

1. Separate input space from working space
2. We need weaker uniformity conditions

P systems with active membranes and input alphabet

Definition

- $\Pi = (\Gamma, \Delta, \Lambda, \mu, w_1, \ldots, w_d, R)$
- Γ is a working alphabet, Δ a disjoint input alphabet
- Input objects cannot be created during the computation:
  - $[a \to w]_h^\alpha$ at most one object $b \in \Delta$ in w (in that case, $a = b$)
  - $a\ [\ ]_h^\alpha \to [b]_h^\beta$ if $b \in \Delta$ then $a = b$
  - $[a]_h^\alpha \to [\ ]_h^\beta\ b$ if $b \in \Delta$ then $a = b$
  - $[a]_h^\alpha \to b$ if $b \in \Delta$ then $a = b$
  - $[a]_h^\alpha \to [b]_h^\beta\ [c]_h^\gamma$ if $b \in \Delta$ (resp., $c \in \Delta$) then $a = b$ and $c \notin \Delta$ (resp., $a = c$ and $b \notin \Delta$)

Space complexity with input alphabet

Definition

- The size $|\mathcal{C}|$ of a configuration $\mathcal{C}$ is the sum of the number of membranes and non-input objects in $\mathcal{C}$
- The space required by a computation $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ is
$$|\vec{\mathcal{C}}| = \max\{|\mathcal{C}_0|, \ldots, |\mathcal{C}_k|\}$$
- The space required by a P system is
$$|\Pi| = \sup\{|\vec{\mathcal{C}}| : \vec{\mathcal{C}} \text{ is a computation of } \Pi\}$$
- A family $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ works in space $f(n)$ iff $|\Pi_x| \le f(|x|)$ for each $x \in \Sigma^\star$

Generalised uniformity conditions

Definition (see Murphy, Woods)

Let E and F be classes of functions. A family of P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ is said to be *(E, F)-uniform* if and only if

- There exists a function $f \in F$ such that $f(1^n) = \Pi_n$, i.e., mapping the unary representation of each natural number to an encoding of the P system processing all inputs of length n
- There exists a function $e \in E$ mapping each string $x \in \Sigma^\star$ to a multiset $e(x) = w_x$ (represented as a string) over the input alphabet of Π_n, where $n = |x|$
- For each $x \in \Sigma^\star$ we have $\Pi_x = \Pi_n(w_x)$, i.e., Π_x is Π_n with the multiset encoding x placed inside the input membrane

DLOGTIME Turing machines I

Definition

- A *deterministic log-time (***DLOGTIME***) Turing machine* is a Turing machine having a read-only input tape of length n, a constant number of read-write work tapes of length $O(\log n)$, and a read-write address tape, also of length $O(\log n)$
- The input tape is not accessed by using a sequential tape head (as the other tapes are); instead, during each step the machine has access to the i-th symbol on the input tape, where i is the number written in binary on the address tape (if $i \geq |n|$ the machine reads an appropriate end-of-input symbol, such as a blank symbol)
- The machine is required to operate in time $O(\log n)$

DLOGTIME Turing machines II

Proposition (Mix Barrington, Immerman, Straubing, JCSS 90)

DLOGTIME *Turing machines are able to*

- *compute the length of their input*
- *compute sums, differences and logarithms of numbers of* $O(\log n)$ *bits*
- *decode simple pairing functions on strings of length* $O(\log n)$
- *extract portions of the input of size* $O(\log n)$ $\square$

DLOGTIME Turing machines are used as a uniformity condition for circuits (e.g., $\mathbf{AC}^0$ circuits)

DLOGTIME-uniform families of P systems I

Definition

A family of P systems is **DLOGTIME**-uniform if the following predicates (describing a mapping $1^n \mapsto \Pi_n$) are **DLOGTIME**-computable

- ALPHABET($1^n, m$) holds for a single integer m such that $\Gamma \cup \Delta \subseteq \{0,1\}^m$, i.e., each symbol of the alphabets of Π_n (whose index is provided in unary notation) can be represented as a binary number of m bits
- LABELS($1^n, m$) is true for a single integer m such that $\Lambda \subseteq \{0,1\}^m$
- INSIDE($1^n, h_1, h_2$) holds iff the membrane labelled by h_1 is immediately contained in h_2 in the initial configuration of Π_n
- INPUT($1^n, h$) holds iff the input membrane of Π_n is h
- MULTISET($1^n, h, i, a$) holds iff the i-th symbol of the string representing the initial multiset contained in membrane h is a

DLOGTIME-uniform families of P systems II

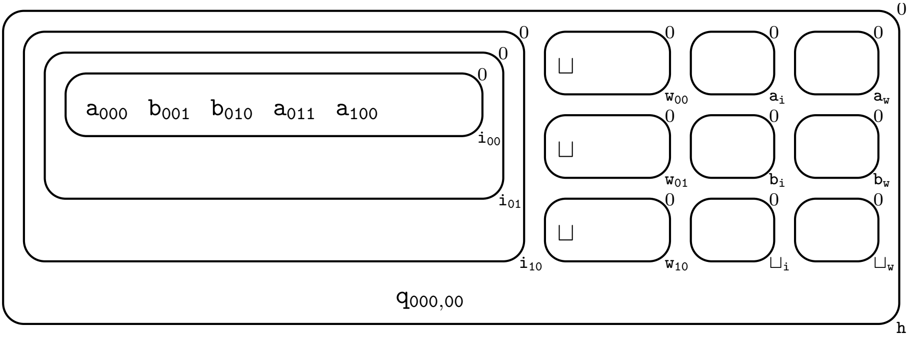
- $\#\text{EVOLUTION}(1^n, h, \alpha, a, m)$ holds iff Π_n has m object evolution rules of the form $[a \to w]_h^\alpha$
- $\text{EVOLUTION}(1^n, h, \alpha, a, i, j, b)$ is true when the i-th rule of the form $[a \to w]_h^\alpha$ (under any chosen, fixed total order of the rules) has $w_j = b$
- $\text{SEND-IN}(1^n, h, \alpha, a, \beta, b)$ iff $a\, [\,]_h^\alpha \to [b]_h^\beta \in R$
- $\text{SEND-OUT}(1^n, h, \alpha, a, \beta, b)$ iff $[a]_h^\alpha \to [\,]_h^\beta\, b \in R$
- $\text{DISSOLVE}(1^n, h, \alpha, a, b)$ iff $[a]_h^\alpha \to b \in R$
- $\text{ELEM-DIVIDE}(1^n, h, \alpha, a, \beta, b, \gamma, c)$ iff $[a]_h^\alpha \to [b]_h^\beta\, [c]_h^\gamma \in R$

The encoding of the input must also be **DLOGTIME**-computable

- $\text{ENCODING}(x, i, a)$ holds when the i-th object of the input multiset encoding x is a

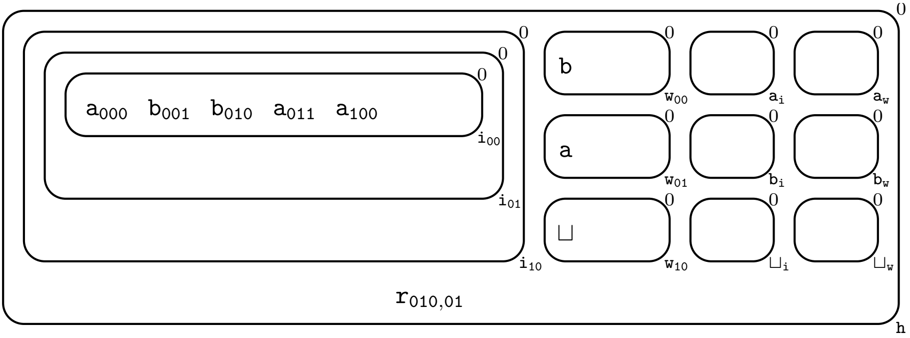
Simulating logspace Turing machines

Initial configuration



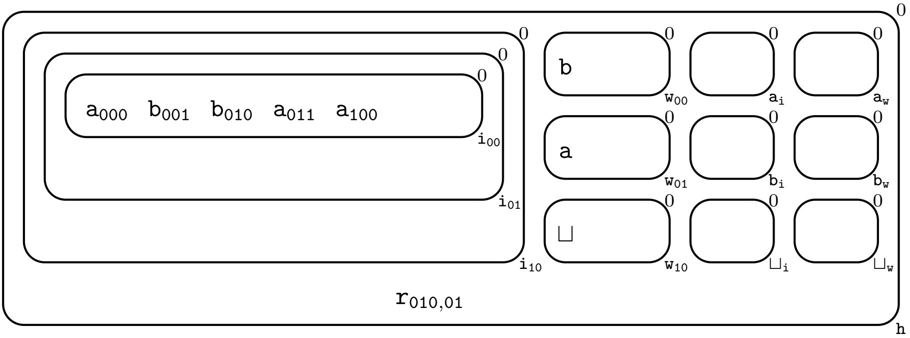
Simulating logspace Turing machines

Later configuration



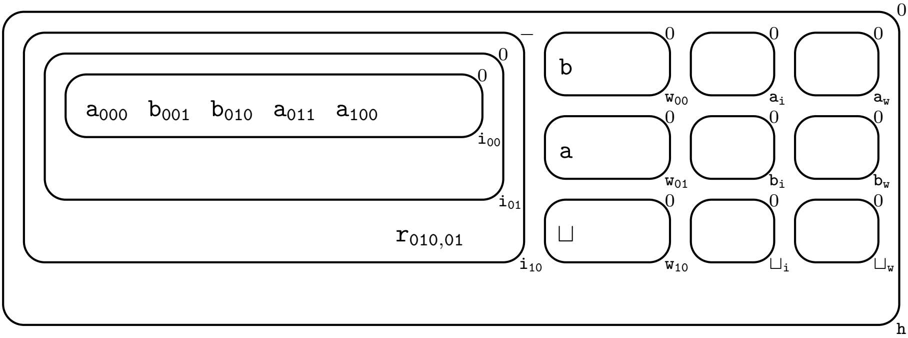
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



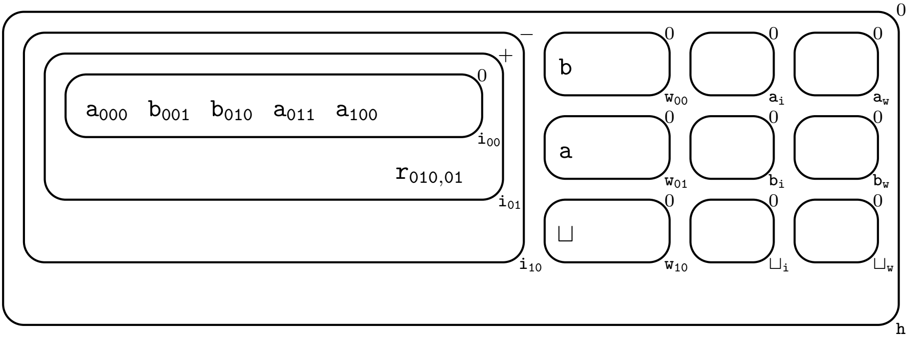
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



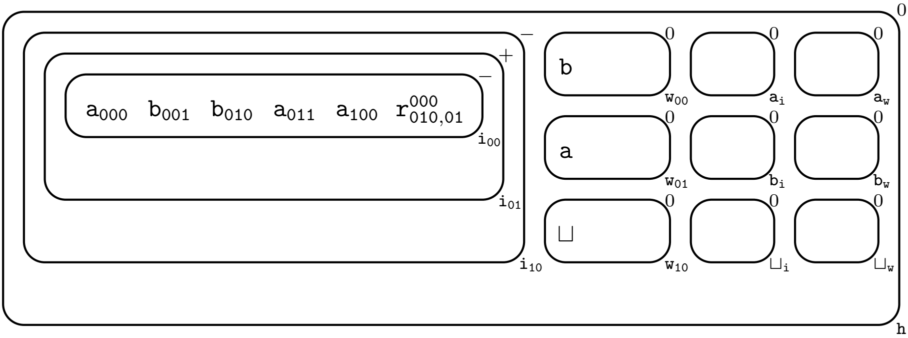
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



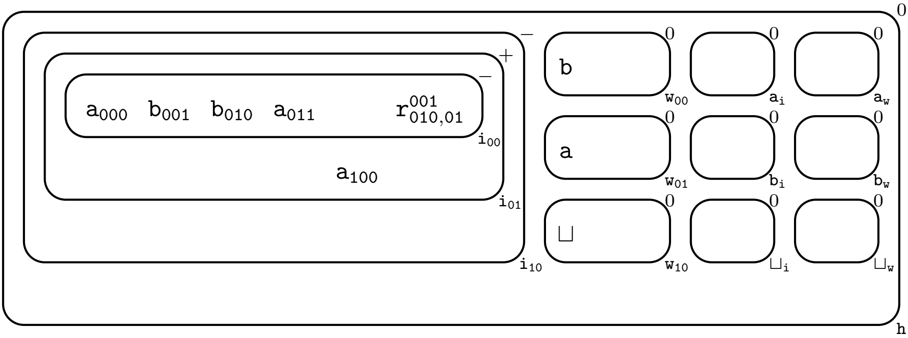
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



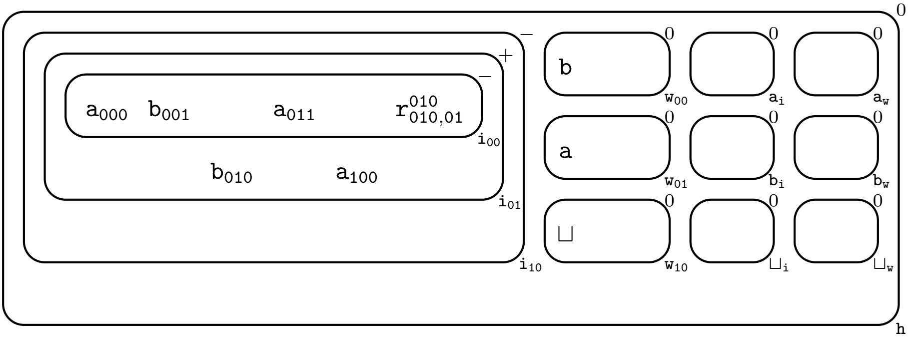
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r},\mathtt{b},\mathtt{a}) = (\mathtt{s},\mathtt{b},+1,-1)$



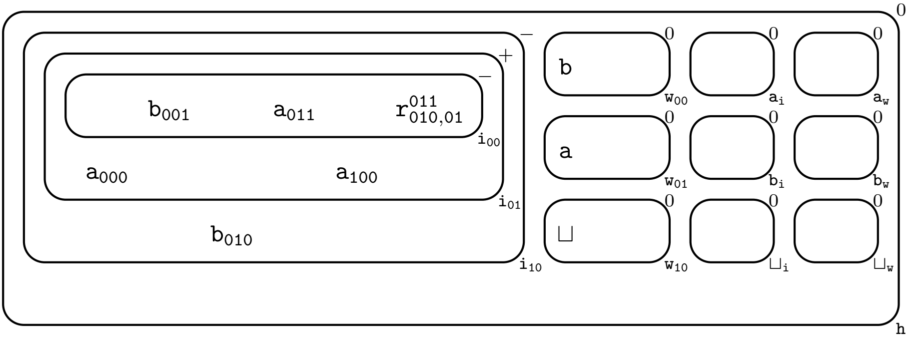
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



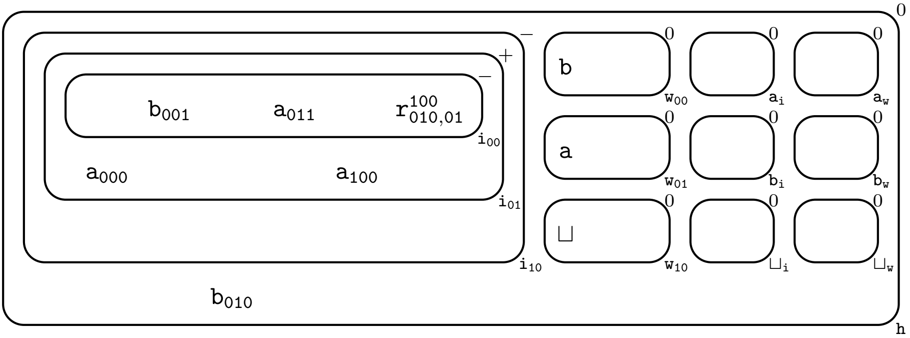
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



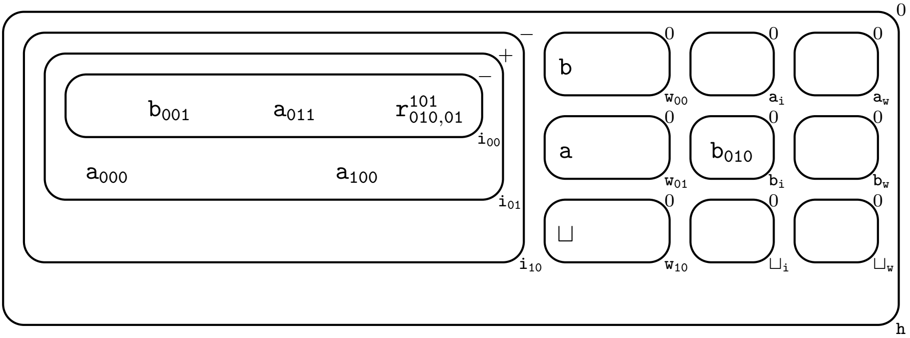
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



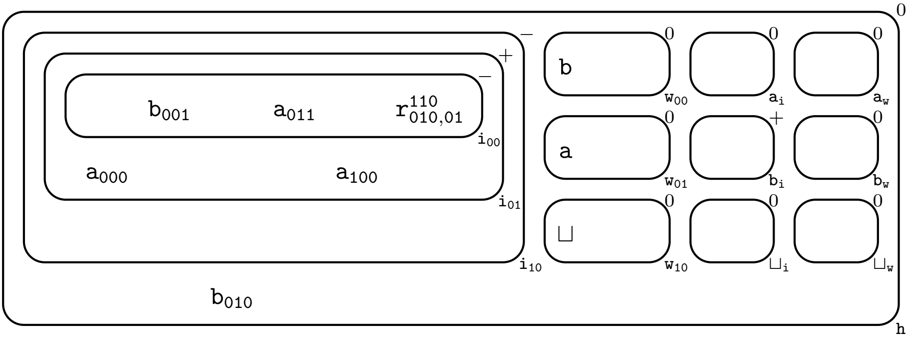
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



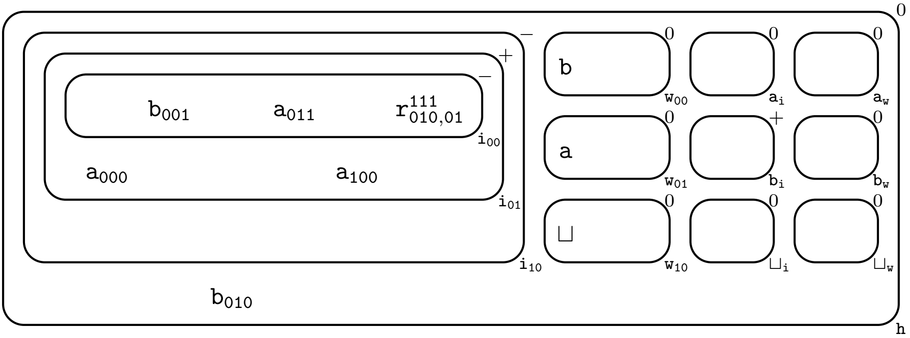
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



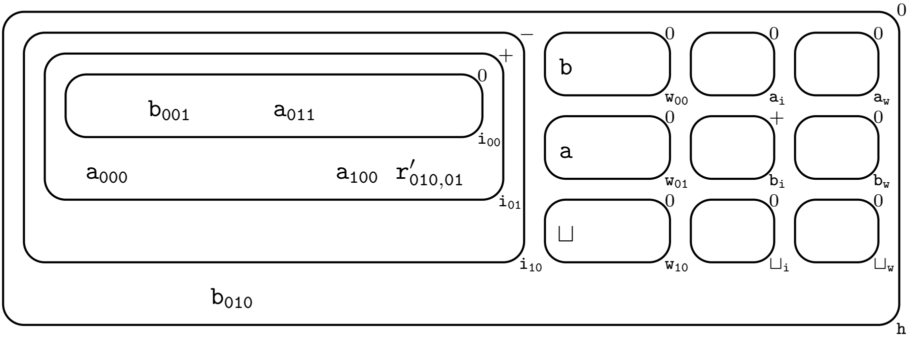
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



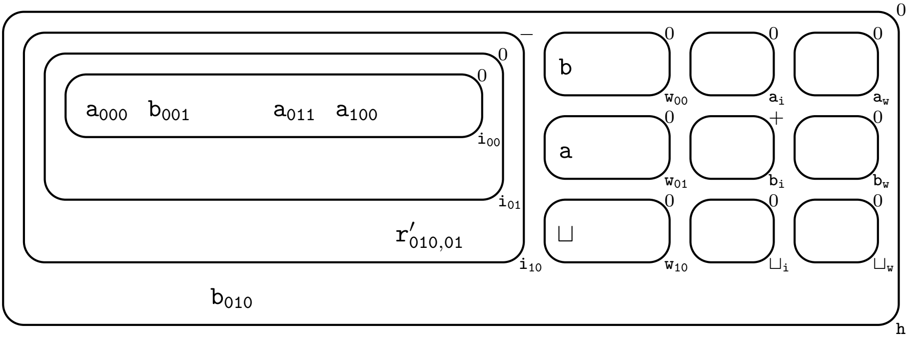
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



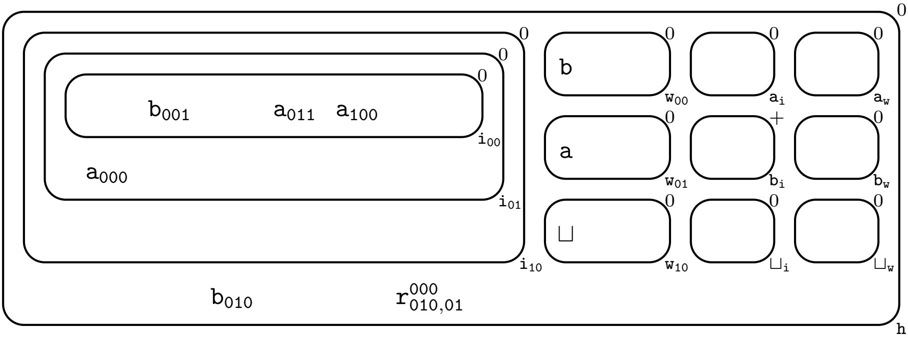
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



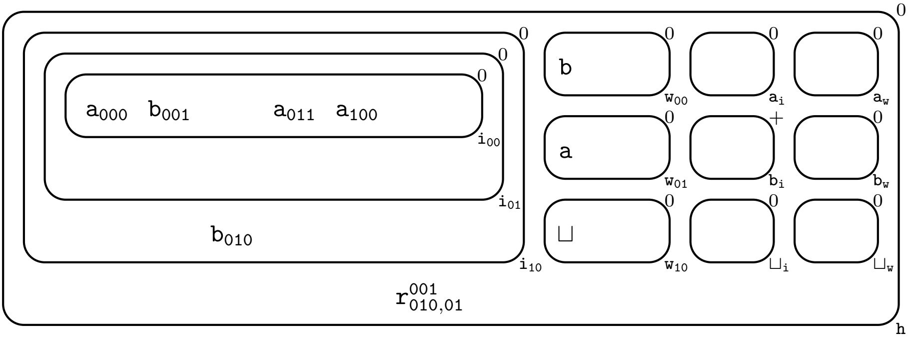
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



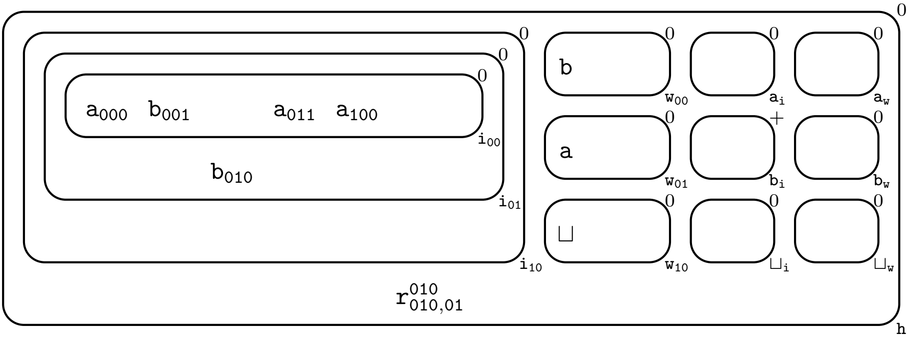
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



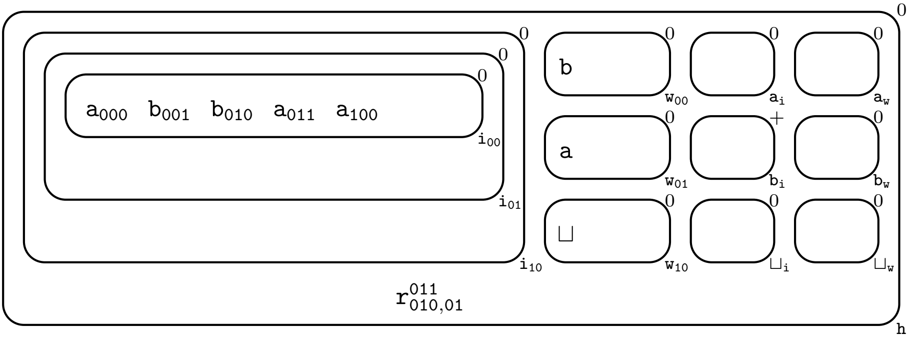
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$

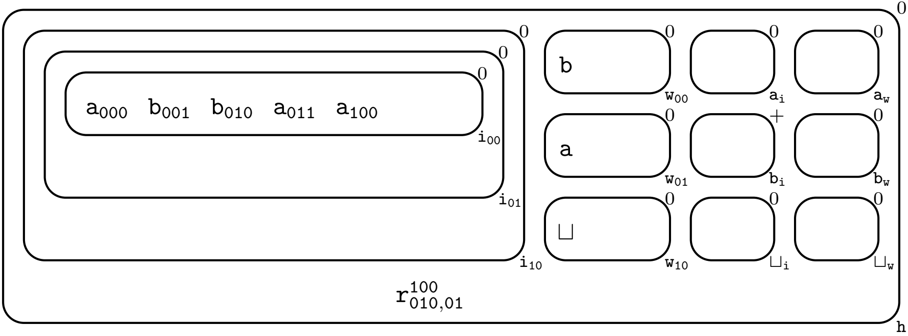


Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$

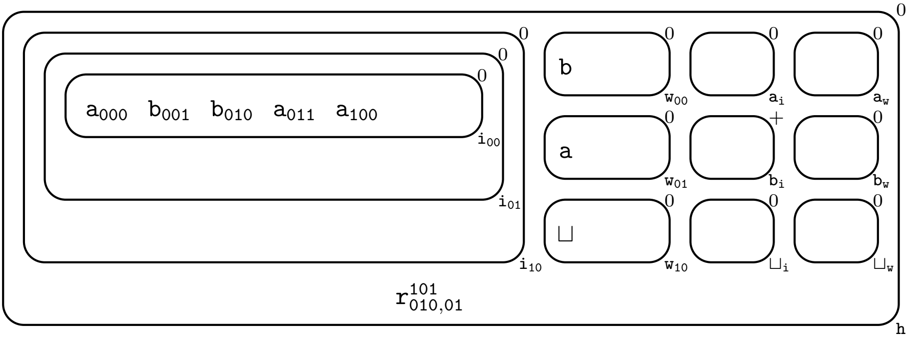


Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



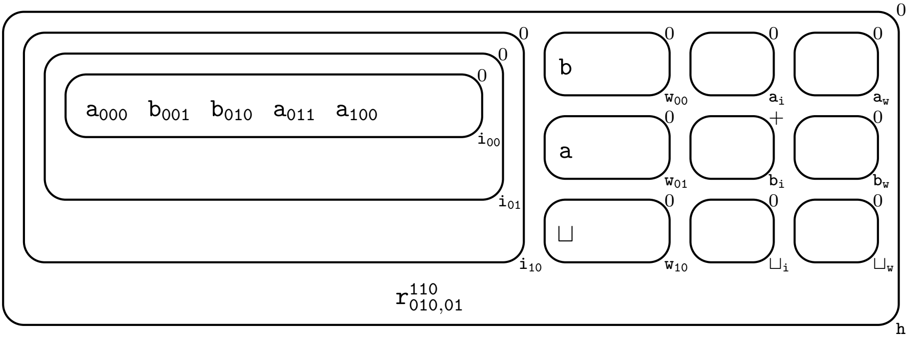
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



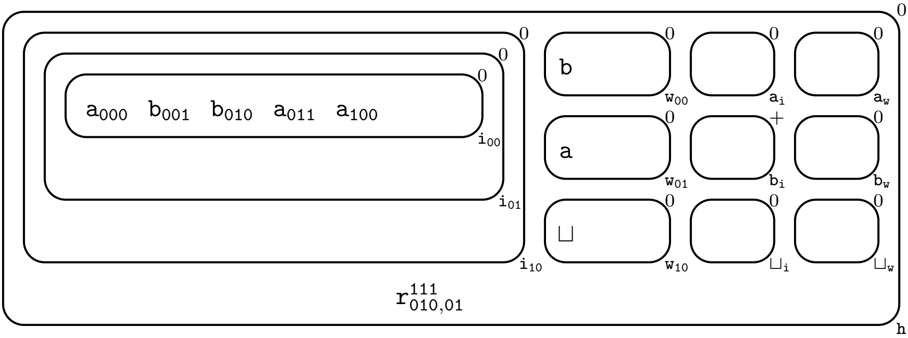
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



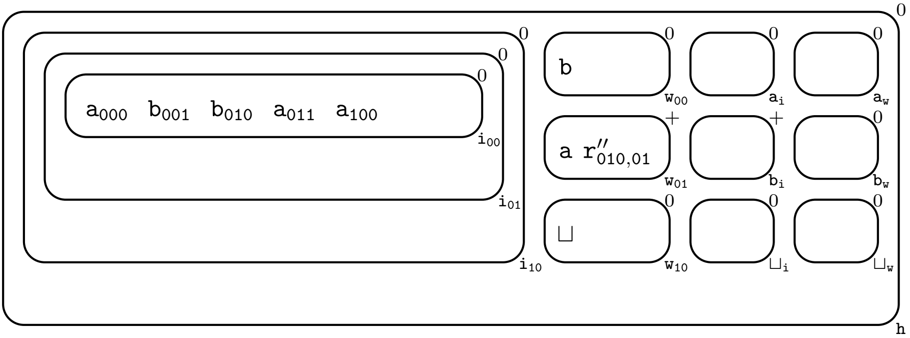
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



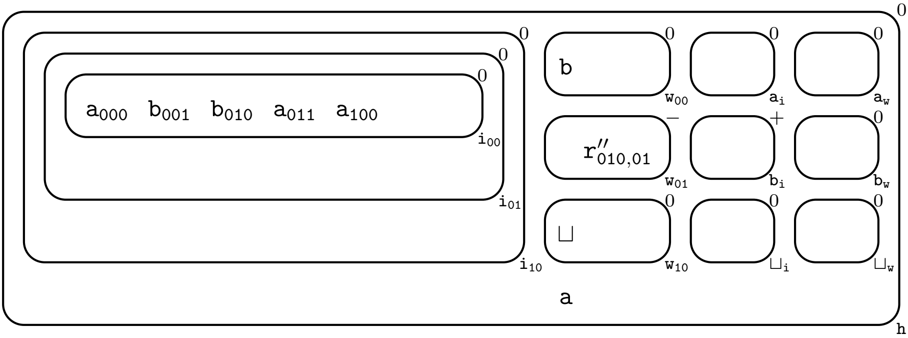
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$

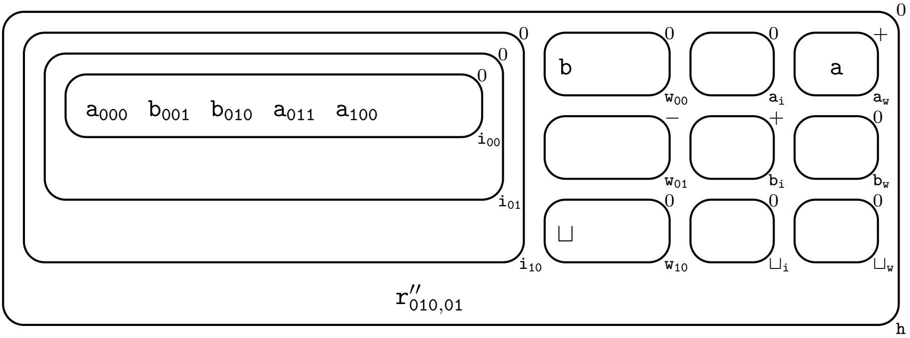


Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$

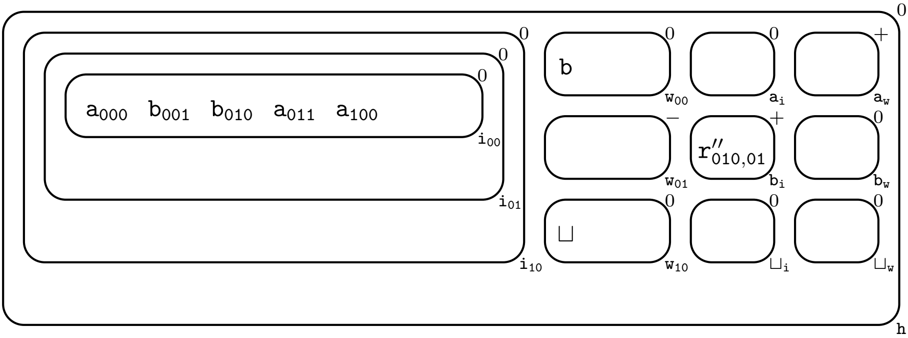


Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



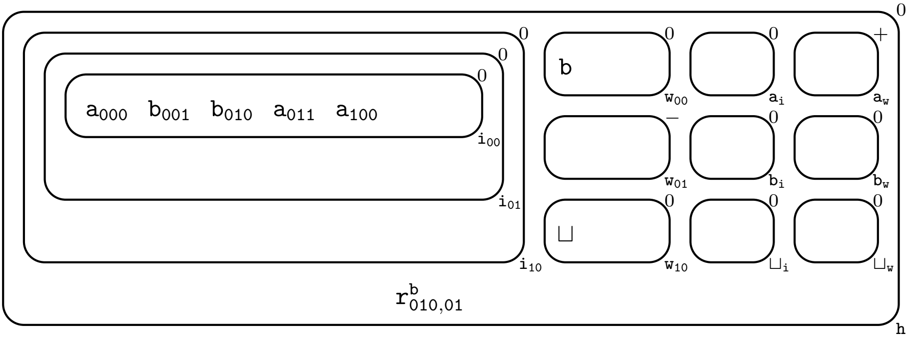
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



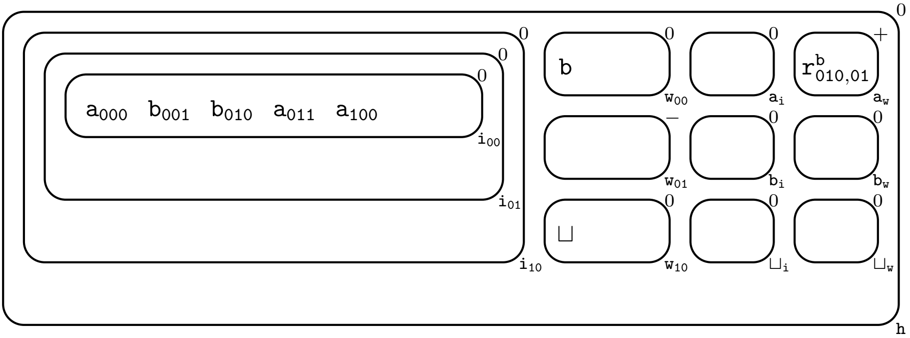
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



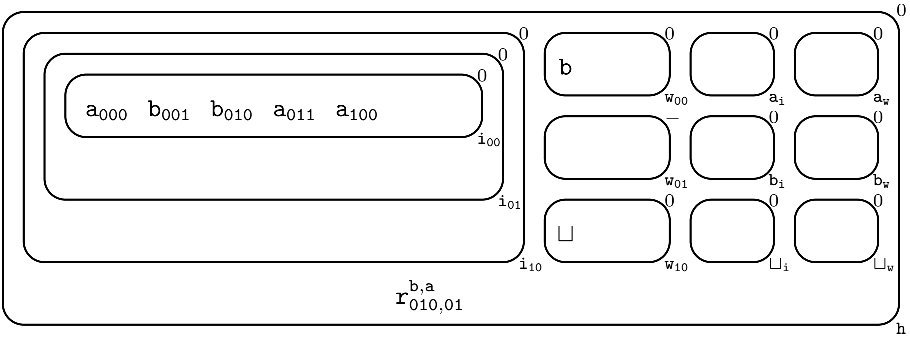
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



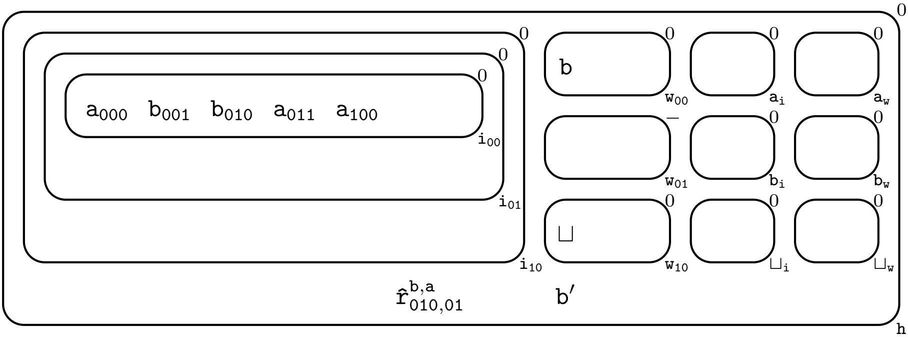
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



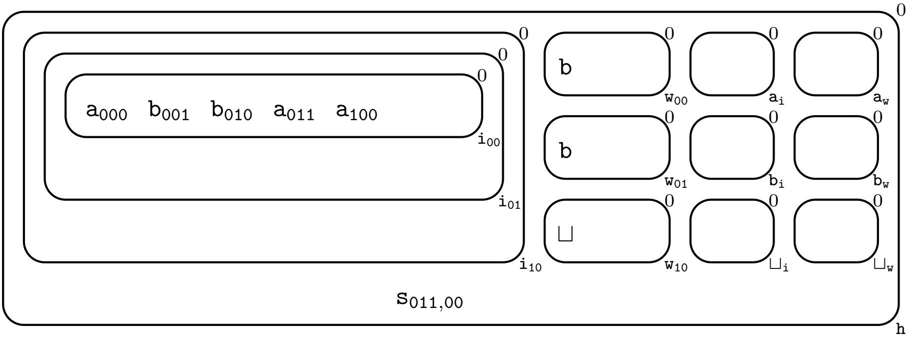
Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$

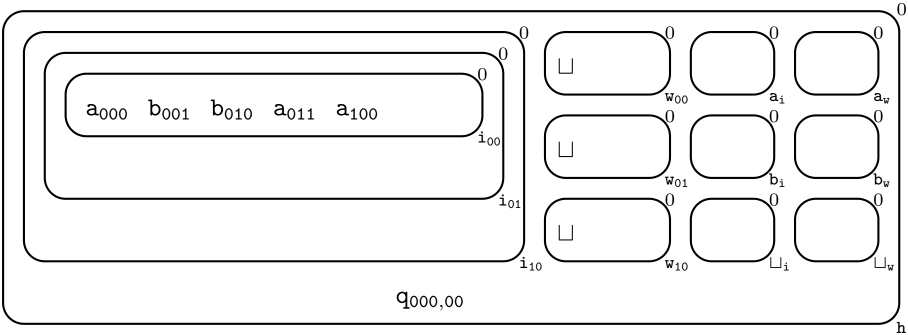


Simulating logspace Turing machines

Simulating $\delta(\mathtt{r}, \mathtt{b}, \mathtt{a}) = (\mathtt{s}, \mathtt{b}, +1, -1)$



Simulating logspace Turing machines



Proposition

The P systems described above work in $O(\log n)$ space $\square$

DLOGTIME-uniformity of the input encoding

Lemma

The ENCODING *predicate is* **DLOGTIME**-*computable*

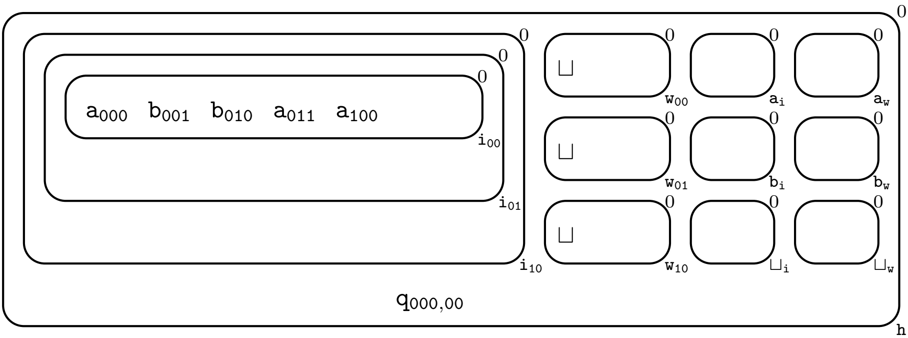
Proof.

We just add subscripts to the input string, denoting their positions

$$\mathrm{ENCODING}(x, i, a_j) \iff j = i \wedge x_i = a$$

The RHS expression can be checked in **DLOGTIME** □

DLOGTIME-uniformity of the membrane structure



$$
\begin{aligned}
\mathrm{INSIDE}(1^n, h_1, h_2) \iff\ & (h_1 = \mathtt{i}_{\ell(n)-1} \wedge h_2 = \mathtt{h}) \vee \\
& (h_1 = \mathtt{i}_j \wedge h_2 = \mathtt{i}_{j+1} \wedge 0 \le j < \ell(n) - 1) \vee \\
& (h_1 = \mathtt{w}_j \wedge h_2 = \mathtt{h} \wedge 0 \le j < s(n)) \vee \\
& (h_1 = a_{\mathtt{i}} \wedge h_2 = \mathtt{h} \wedge a \in \Sigma) \vee \\
& (h_1 = a_{\mathtt{w}} \wedge h_2 = \mathtt{h} \wedge a \in \Sigma)
\end{aligned}
$$

DLOGTIME-uniformity of the rules

Just an example:

$$[q_{i,w}^{t} \to q_{i,w}^{t+1}]_{\mathtt{h}}^{0} \qquad \text{for } 0 \leq t < \frac{n}{2} + \ell(n) + 1,$$
$$q \in Q, 0 \leq i < n, 0 \leq w < s(n)$$

$\mathrm{EVOLUTION}(1^n, \mathtt{h}, 0, q_{i,w}^{t}, 0, j, b)$ can be checked in **DLOGTIME**

Main result

Theorem

Let M be a deterministic Turing machine with an input tape (of length n) and a work tape of length $O(\log n)$. Then, there exists a $(\mathbf{DLT}, \mathbf{DLT})$-uniform family $\mathbf{\Pi}$ of confluent recogniser P systems with active membranes working in logarithmic space such that $L(M) = L(\mathbf{\Pi})$. □

Corollary

$\mathbf{L} \subseteq (\mathbf{DLT}, \mathbf{DLT})\text{-}\mathbf{LMCSPACE}_{\mathcal{AM}}$. □

Conclusions & open problems

- "New" uniformity condition for P systems
- Conjecture: applicable to most previous results
- Logspace P systems with active membranes are at least as powerful as logspace Turing machines
- Is the converse result true?
- If it is not the case, what else is contained? **NL**? **P**?

Köszönöm a figyelmet!

[*Thanks for your attention!*]