MTA SZTAKI

Variants of Distributed P Automata and the Efficient Parallelizability of Languages

György Vaszil

MTA SZTAKI – Hungarian Academy of Sciences

Budapest, Hungary

25.08.2011

Outline

- Types of parallelism in P systems
  - parallel rule application
  - parallel processing in the different regions

- Distributed P systems
  - P automata, distributed P automata

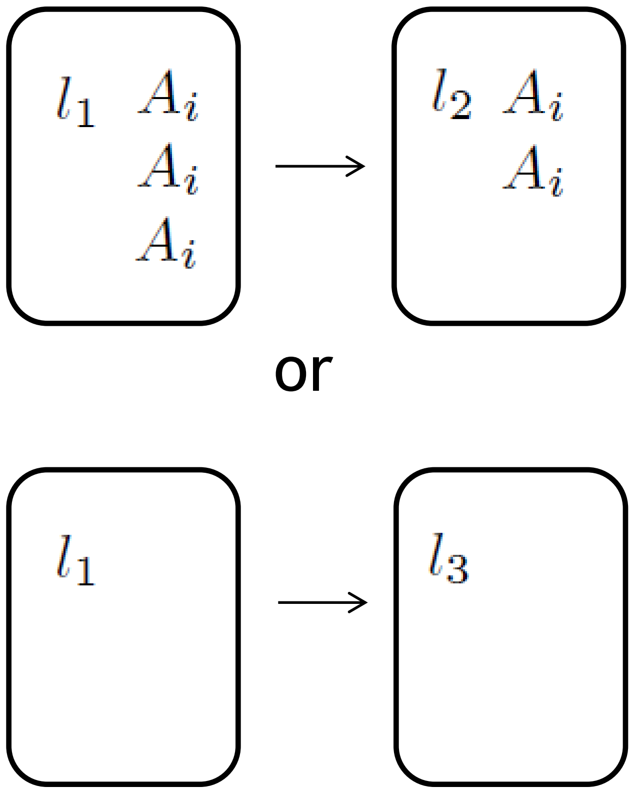
- Parallelizability of languages
  - properties influencing the efficiency of parallelization

Parallelism in P systems

The “philosophy” behind distributed P systems

Examples of parallelism in P systems

The use **maximal parallel** way of rule application for “**zero-check**”. A register machine-like instruction: (l_1, R_i-, l_2, l_3)



$(l_1, out; l_1' l_1'', in)$

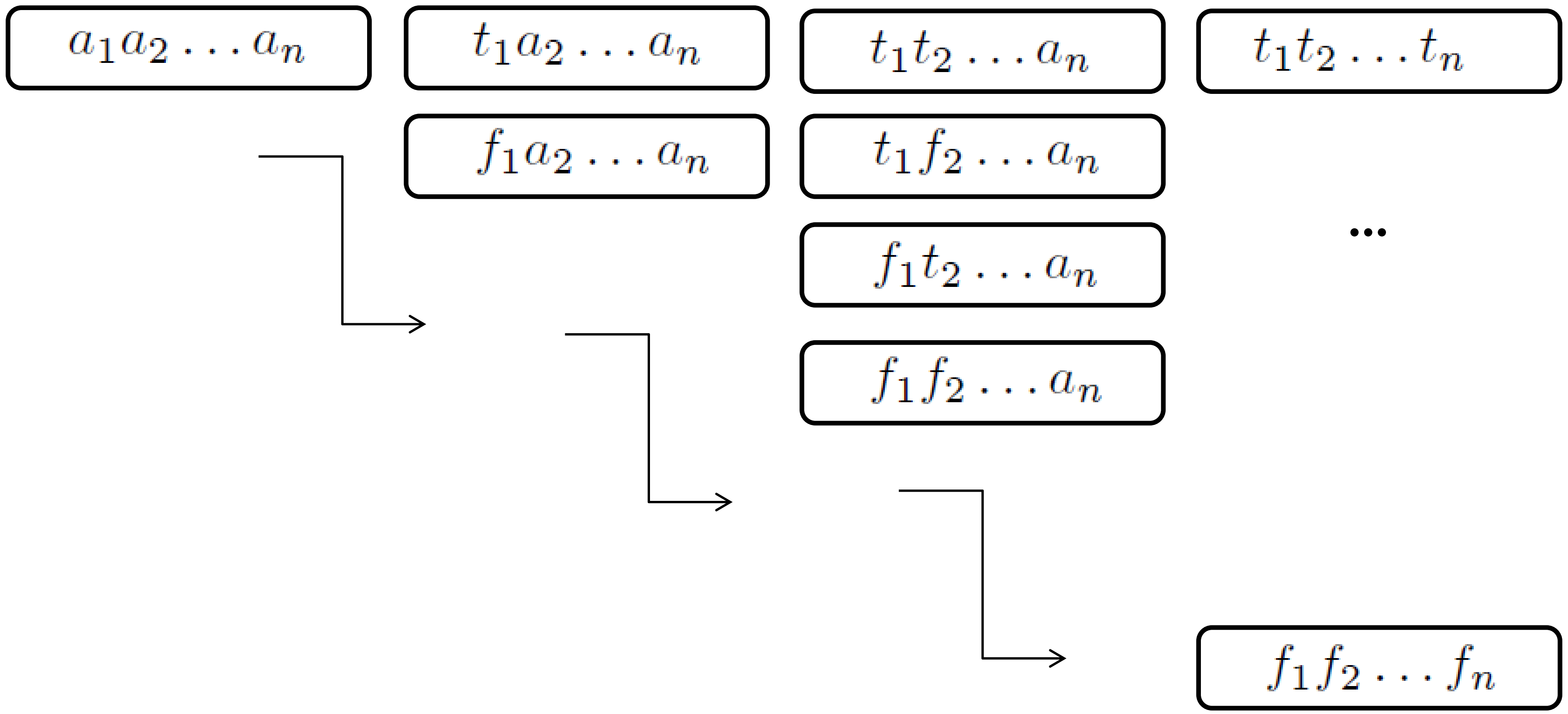
$(l_1' A, out; l_1''', in)$ $(l_1'', out; l_1^{iv}, in)$

$(l_1''' l_1^{iv}, out; l_2, in)$

$(l_1' l_1^{iv}, out; l_3, in)$

Examples of parallelism in P systems

Creating **exponential** workspace in **linear** time: $[a_i] \to [t_i][f_i]$



Examples of parallelism in P systems

We have seen:

- **maximal parallel** rule application
- the creation of **exponential** workspace in **linear** time

The **system** is still viewed as **one processing unit** and the whole input is given to it.

Distributed P systems

The idea behind **distributed** P systems is **different:**

- The system is composed of **several** processing **units or components**
- The components process **different parts of the input** in **parallel**
- The components **communicate** with each other

Possible questions concerning distributed P systems

- Is it possible to **split the input** into pieces?
- Is the distributed computation **more efficient** than the non-distributed one?

Distributed P automata

- The input is a **string**:

splitting the input ←→ cutting the string into pieces

- The **efficiency** of the distributed computation:

the number of **computational** steps / time

+

the amount of **communication**

Bibliographical remarks - distributed P automata

Gh. Păun and M. J. Pérez-Jiménez. Solving problems in a distributed way in membrane computing: dP systems. *International Journal of Computing, Communication and Control*, V(2):238–250, 2010.

R. Freund, M. Kogler, Gh. Păun, and M. J. Pérez-Jiménez. On the power of P and dP automata. *Annals of Bucharest University Mathematical-Informatics Series*, LVIII:5–22, 2010.

Gy. Vaszil. On the parallelizability of languages accepted by P automata. In J. Kelemen and A. Kelemenová, editors, *Computation, Cooperation, and Life. Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday*, volume 6610 of *Lecture Notes in Computer Science*, pages 170–178. Springer, Berlin Heidelberg, 2011.

Bibliographical remarks - other useful sources

E. Csuhaj-Varjú, M. Oswald, and Gy. Vaszil. P automata. In Gh. Păun, G. Rozenberg, and A. Salomaa, editors, *The Oxford Handbook of Membrane Computing*, chapter 6, pages 144–167. Oxford University Press, 2010.

Gh. Păun, M.J. Pérez-Jiménez: P and dP automata: A survey. *Rainbow of Computer Science* (C.S. Calude, G. Rozenberg, A. Salomaa, eds.), LNCS, Springer, Berlin, 2010 (in press).

Distributed P systems

(Non-distributed) P automata

P automata

- An **antiport P system** in an **environment** from where the **input** is read
- Given an **initial configuration** and a set of final **(accepting) configurations**
- A **sequence of multisets** is read from the environment during the computation
- The multiset sequence is **accepted** if the computation ends in an **accepting configuration**

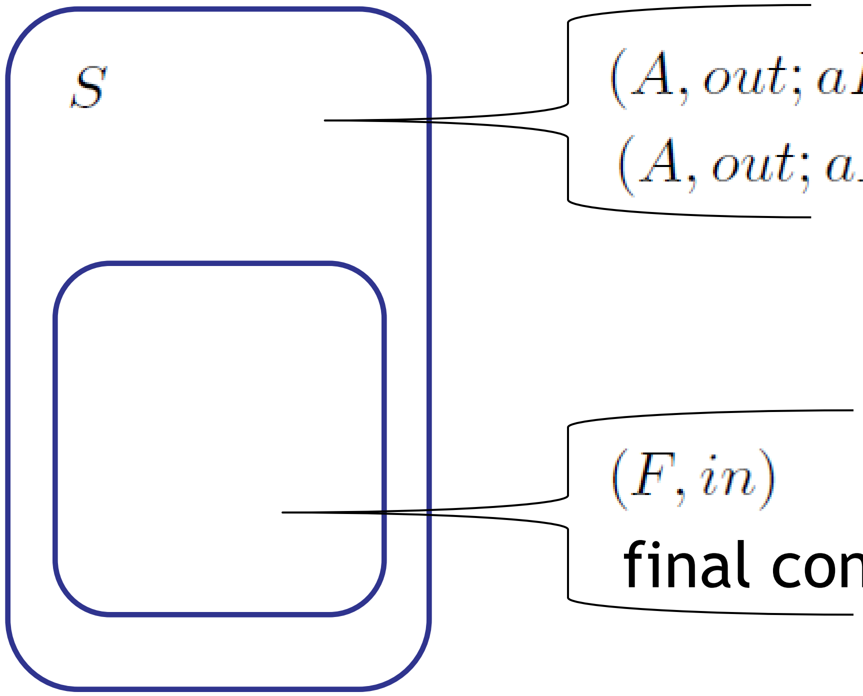
[Csuhaj-Varjú, Vaszil 2002]

P automata - An example

Given a **regular grammar** with rule types: $A \rightarrow aB, A \rightarrow a \in P$

initial configuration:

rules:

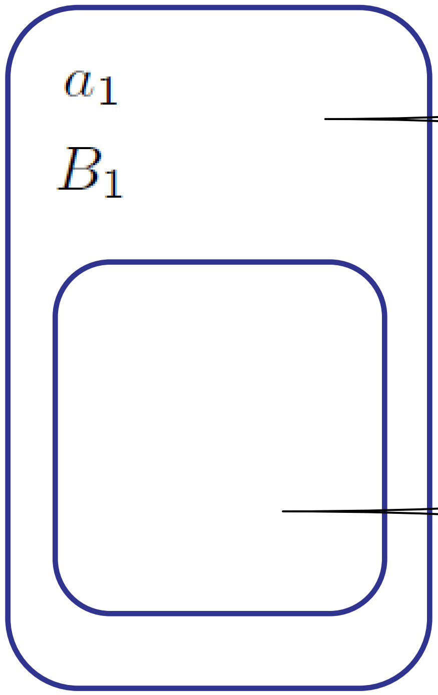


final configuration: F is in the region

P automata - An example

Given a **regular grammar** with rules: $A \to aB, A \to a \in P$

configuration: rules:



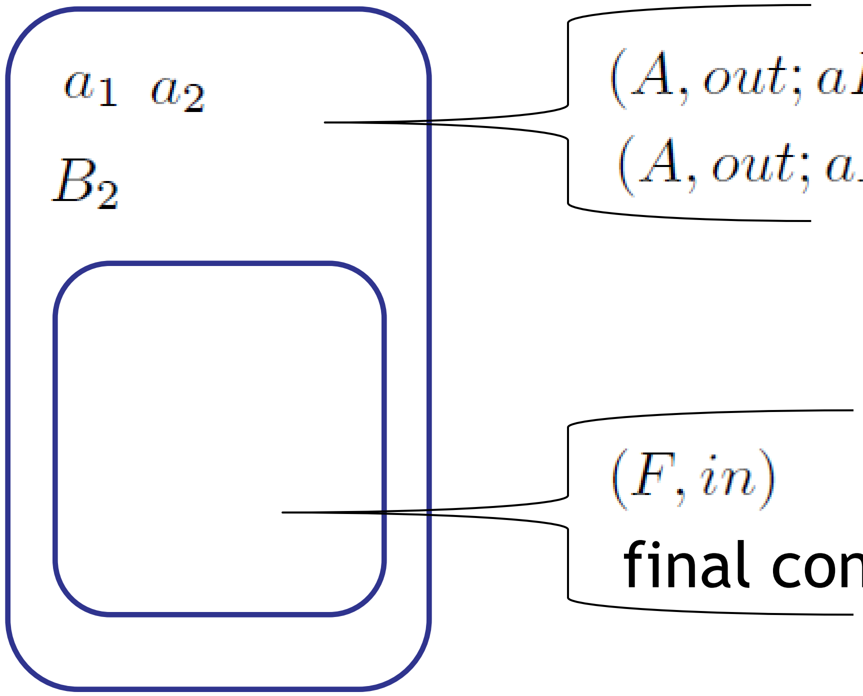
$$\left\{ \begin{array}{l} (A, out; aB, in) \\ (A, out; aF, in) \end{array} \right.$$

$$\left\{ \begin{array}{l} (F, in) \\ \text{final configuration: } F \text{ is in the region} \end{array} \right.$$

P automata - An example

Given a **regular grammar** with rules: $A \to aB, A \to a \in P$

configuration: rules:

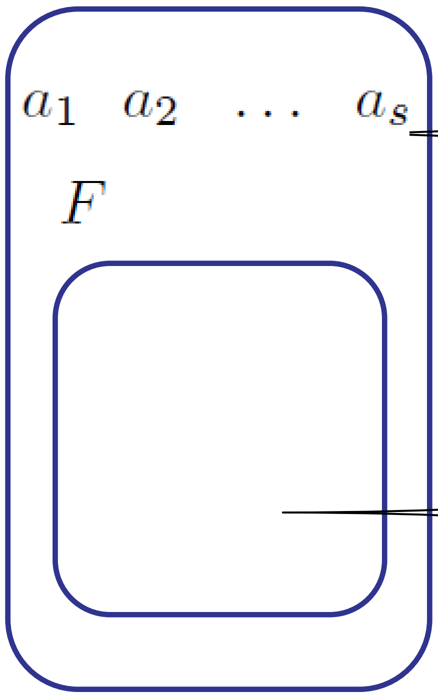


final configuration: F is in the region

P automata - An example

Given a **regular grammar** with rules: $A \to aB, A \to a \in P$

configuration: rules:



$(A, out; aB, in)$
$(A, out; aF, in)$

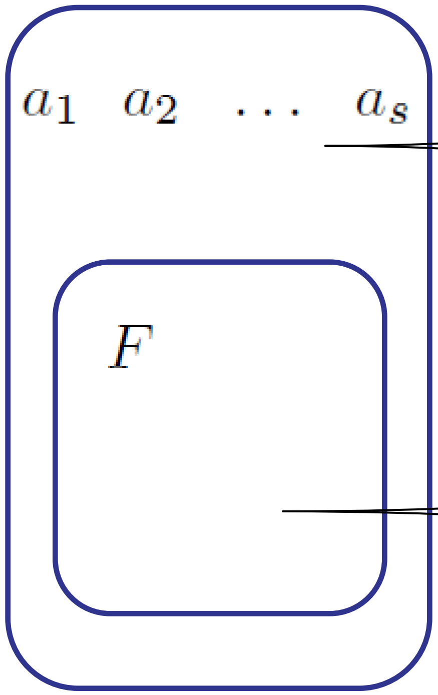
(F, in)
final configuration: F is in the region

P automata - An example

Given a **regular grammar** with rules: $A \to aB, A \to a \in P$

final configuration:

rules:



$(A, out; aB, in)$

$(A, out; aF, in)$

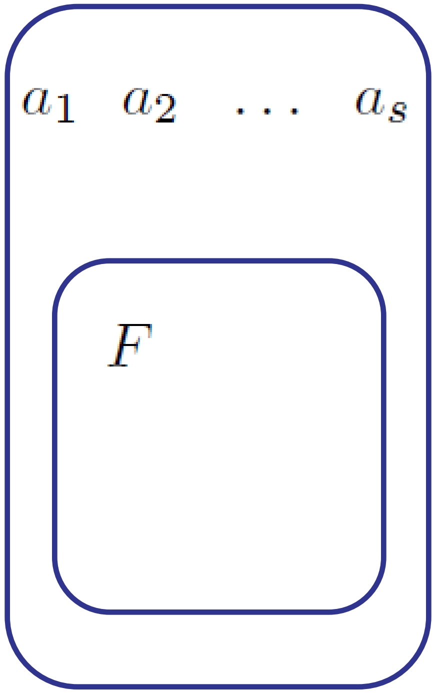
(F, in)

final configuration: F is in the region

P automata - An example

Given a **regular grammar** with rules: $A \to aB, A \to a \in P$

final configuration:



The set of accepted **multiset sequences:**

$$\boxed{\{a_1B_1, a_2B_2, \ldots, a_sF \mid a_1a_2\ldots a_s \in L\}}$$

P automata - An other example

A **finite automaton** $M = (\Sigma_1, Q, \delta, q_0, F)$, $\Sigma_1 = \{a_1, \ldots, a_k\}$.
A simulating P automaton with 2 membranes:

$$\begin{aligned} w_1 &= a\#, \\ P_1 &= \{(a^i, in; a, out)|_t, (a^{i-1}, out)|_{t'} \mid t = [q_j, a_i, q_k],\ i > 1\} \cup \\ &\quad \{(a, in; a, out)|_t \mid t = [q_j, a_1, q_k]\}, \end{aligned}$$

$$\begin{aligned} w_2 &= \{\{t, t' \mid t \in TR\}\}, \\ P_2 &= \{(\#, in; t_0, out) \mid t_0 = [q_0, a_i, q]\} \cup \\ &\quad \{(t, in; t', out), (t', in; s, out) \mid t \in TR,\ s \in next(t')\}, \\ F_2 &= \{\ \{\{t, t' \mid t \in TR\}\} - \{\{s'\}\} \mid \text{for} \\ &\qquad \text{all } s' \in TR' \text{ such that } s' = [q, a_i, q_f]',\ q_f \in F\}. \end{aligned}$$

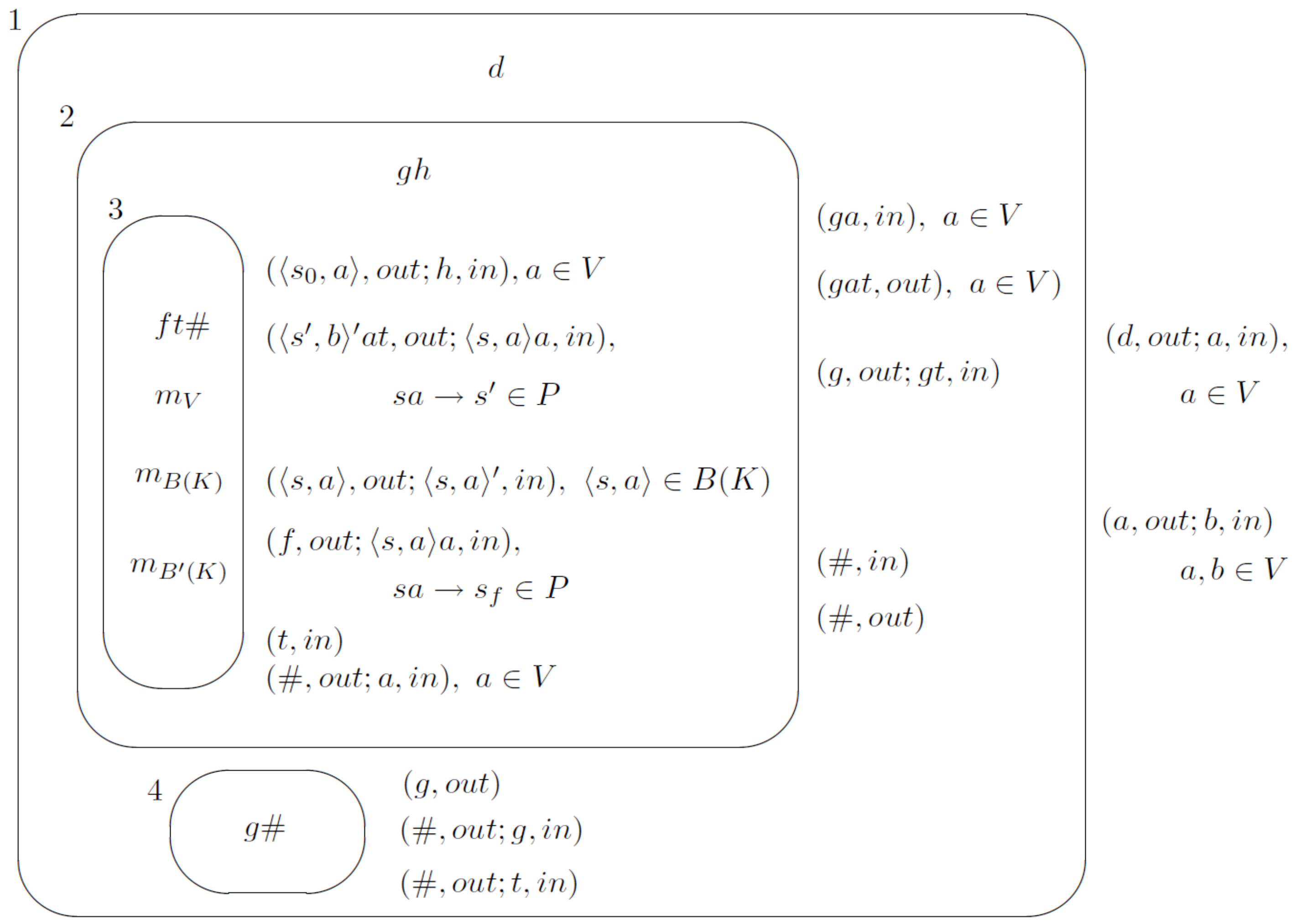
MTA SZTAKI

P automata - An other example

The accepted **multiset sequences:**

$$\boxed{\{\underbrace{a\ldots a}_{i_1},\underbrace{a\ldots a}_{i_2},\ldots,\underbrace{a\ldots a}_{i_s} \mid a_{i_1}a_{i_2}\ldots a_{i_s}\in L\}}$$

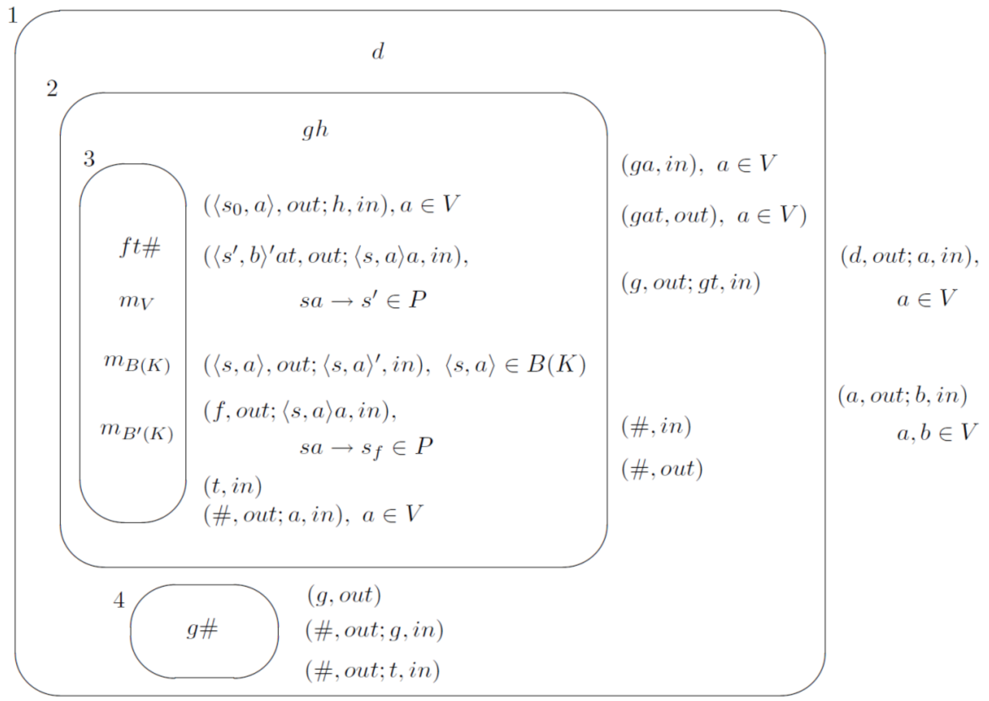
P automata - A third example



[Freund, Kogler, Paun, Pérez-Jiménez 2010]

MTA SZTAKI

P automata - A third example



The set of accepted **multiset sequences:**

$$\boxed{\{a_1, a_2, \ldots, a_s \mid a_1 a_2 \ldots a_s \in L\}}$$

P automata

- An **antiport P system** in an **environment** from where the **input** is read
- Given an **initial configuration** and a set of final **(accepting) configurations**
- A **sequence of multisets** is read from the environment during the computation
- The multiset sequence is **accepted** if the computation ends in an **accepting configuration**

- The accepted **multiset sequence** is **interpreted as a string**

P automata - a more formal definition

A P automaton is

$$\Pi = (V, \mu, P_1, \ldots, P_n, c_0, \mathcal{F})$$

with

- object alphabet
- membrane structure
- rules corresponding to the regions
- initial configuration $c_0 = (w_1, \ldots, w_n)$, $w_i \in V^*$
- set of accepting configurations $E_1 \times \ldots \times E_n,\ E_i \subseteq V^*$ with E_i being finite, or $E_i = V^*$.

The input mapping

Maps the **sequences of multisets** over the object alphabet **to strings** over the input alphabet:

$$\boxed{f : V^* \to 2^{T^*}}$$

The **language accepted** by a P automaton Π :

$$L(\Pi, f) = \{f(v_1) \dots f(v_s) \mid v_1, \dots, v_s \text{ is an accepted multiset sequence of } \Pi\}$$

The input mapping

The **first** example: $\{a_1B_1, a_2B_2, \ldots, a_sF \mid a_1a_2\ldots a_s \in L\}$

- the mapping: $V = N \cup T,\ f(aA) = \{a\}$ where $A \in N,\ a \in T$

The **second** example: $\{a^{i_1}, a^{i_2}, \ldots, a^{i_s} \mid a_{i_1}a_{i_2}\ldots a_{i_s} \in L\}$

- the mapping: $V = \{a\},\ f(a^i) = \{a_i\},\ a_i \in T = \{a_1, \ldots, a_t\}$

The **third** example: $\{a_1, a_2, \ldots, a_s \mid a_1a_2\ldots a_s \in L\}$

- the mapping: $V = T = \{a_1, \ldots, a_t\}$,

$f(a_{i_1}a_{i_2}\ldots a_{i_k}) = \{a_{j_1}a_{j_2}\ldots a_{j_k} \mid j_1j_2\ldots j_k \text{ is a}$

$\text{permutation of } i_1i_2\ldots i_k\}$

The choice of the input mapping and the power of P automata

If **erasing** is allowed, ***RE* languages** are easily **obtained** with simple systems having just **one membrane** (extended P automata, analyzing P systems).

[Freund, Oswald 2002]

The choice of the input mapping and the power of P automata - input mappings with erasing

Counter machines with a read-only **input tape** can be simulated.

Register machine-like instructions for two counters + instructions for reading the input tape:

$$(l_1, x, R_i-, l_2, l_3), x \in T \cup \{\varepsilon\}$$

$$\boxed{\begin{array}{ll} l_1 & A_i \\ & A_i \\ & A_i \end{array}} \longrightarrow \boxed{\begin{array}{ll} l_2 & A_i \\ x & A_i \\ & \end{array}}$$

the input mapping is **erasing**:

$$f(xl_2) = \{x\}$$

The choice of the input mapping and the power of P automata

The input mapping should be **nonerasing**:

- in order to explore new possibilities

The input mapping should be **simple** from the point of view of **computational complexity**:

- the power of the system should not come from the power of a complex input mapping

Two types of input mappings

- $f = f_{perm}$ if and only if $V = T$, and
  $f(v) = \{a_1 a_2 \ldots a_s \mid |v| = s, \text{ and } a_1 a_2 \ldots a_s \text{ is a permutation of the elements of } v\}$

  (Example 3)

- $f \in \text{TRANS}$ if and only if, we have $f(v) = \{w\}$ for some $w \in T^*$ which is obtained by applying a finite transducer to the string representation of the multiset v.

  (Examples 1, 2)

Example 1: $\{a_1B_1, a_2B_2, \ldots, a_sF \mid a_1a_2\ldots a_s \in L\}$

- the mapping: $V = N \cup T,\ f(aA) = \{a\}$ where $A \in N,\ a \in T$

Example 2: $\{a^{i_1}, a^{i_2}, \ldots, a^{i_s} \mid a_{i_1}a_{i_2}\ldots a_{i_s} \in L\}$

- the mapping: $V = \{a\},\ f(a^i) = \{a_i\},\ a_i \in T = \{a_1, \ldots, a_t\}$

Example 3: $\{a_1, a_2, \ldots, a_s \mid a_1a_2\ldots a_s \in L\}$

- the mapping: $V = T = \{a_1, \ldots, a_t\}$,

$$f(a_{i_1}a_{i_2}\ldots a_{i_k}) = \{a_{j_1}a_{j_2}\ldots a_{j_k} \mid j_1j_2\ldots j_k \text{ is a permutation of } i_1i_2\ldots i_k\}$$

Systems with permutation mappings

There are simple **linear languages** which cannot be characterized with systems using f_{perm}.

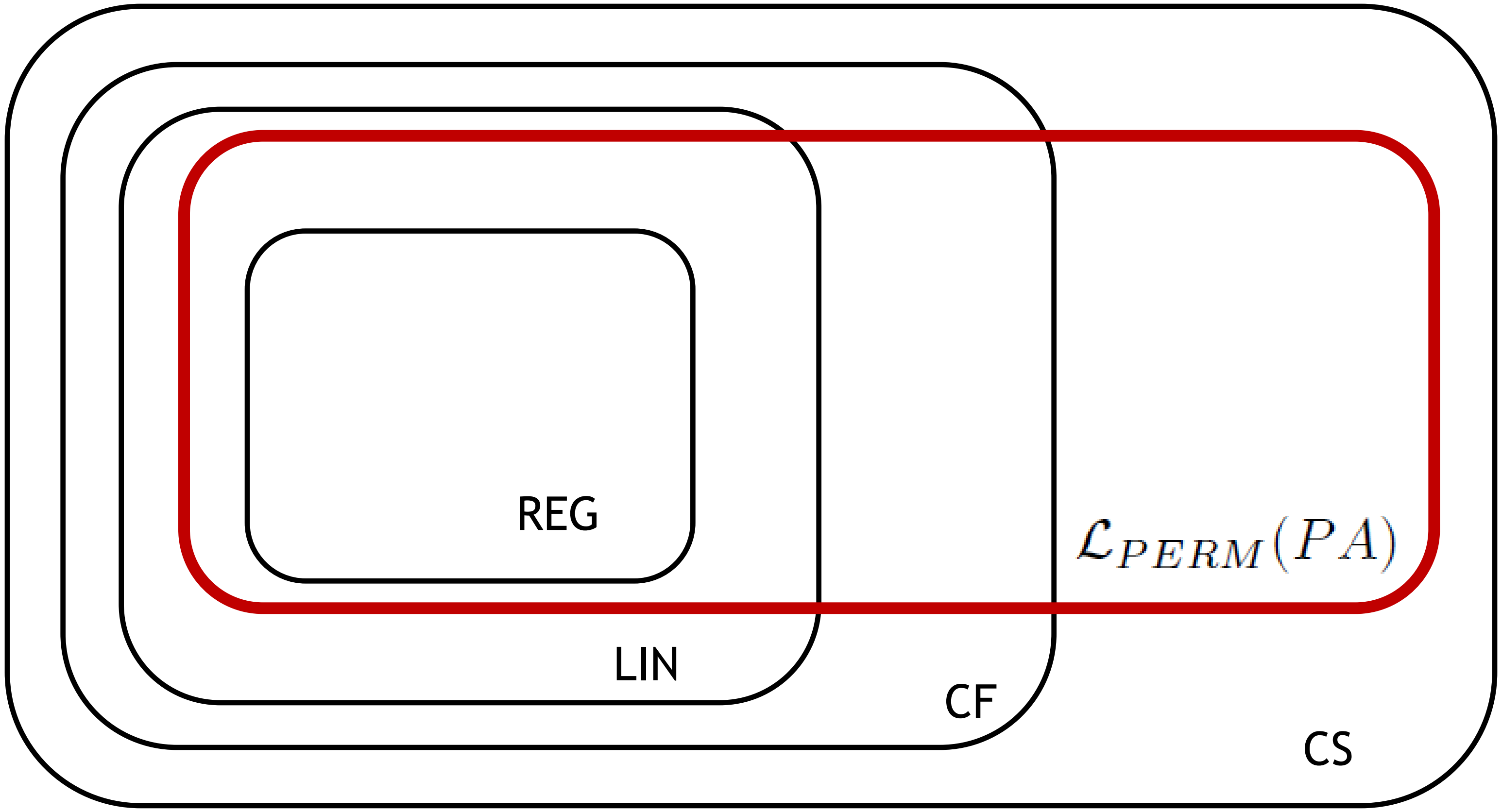
$$\boxed{L = \{(ab)^n(ac)^n \mid n \geq 1\} \notin \mathcal{L}_{PERM}(PA)}$$

On the other hand:

$$\{(aac)^n(bbd)^n \mid n \geq 1\} \in \mathcal{L}_{PERM}(PA)$$

[Paun, Pérez-Jiménez 2010]

Systems with permutation mappings



[Freund, Kogler, Paun, Pérez-Jiménez 2010]

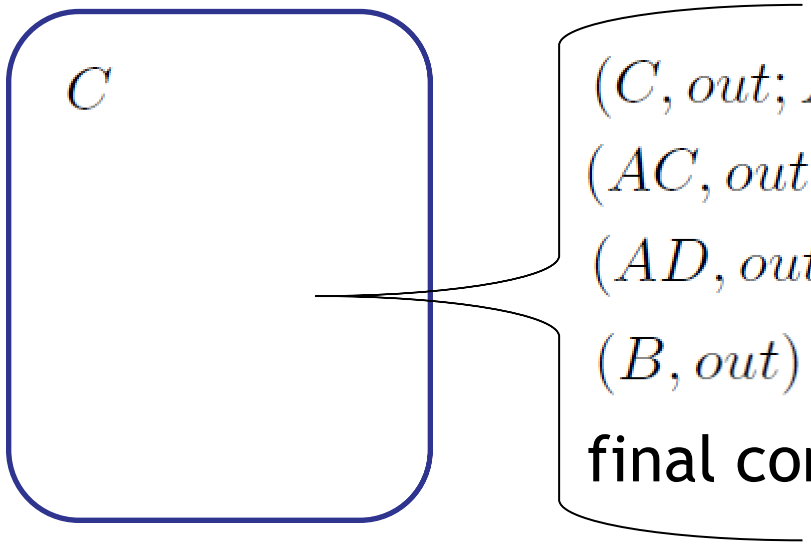
$$L = \{(ab)^n(ac)^n \mid n \geq 1\} \notin \mathcal{L}_{PERM}(PA)$$

$$\Big\updownarrow$$

$$L = \{(aac)^n(bbd)^n \mid n \geq 1\} \in \mathcal{L}_{PERM}(PA)$$

Systems with transduction mappings

initial configuration:



rules:

$(C, out; AC, in)$

$(AC, out; BD, in)$

$(AD, out; BD, in)$

(B, out)

final configuration: A single D in the region

The accepted multiset sequences: $\{(AC)^n(BD)^n \mid n \geq 1\}$

Consider: $f_1(AC) = \{ab\},\ f_1(BD) = \{ac\}$

$f_2(AC) = \{aac\},\ f_2(BD) = \{bbd\}$

Systems with transduction mappings

Any **context-sensitive** language can be characterized.

$$\boxed{\mathcal{L}_{TRANS}(PA) = CS}$$

Moreover:

For any kind of $f : V^* \rightarrow 2^{T^*}$, as long as it is not more complex than **linear space computable** (by Turing machines), $L(\Pi, f) \in \mathrm{CS}$.

[Csuhaj-Varjú, Ibarra, Vaszil 2004]

Distributed P systems

Distributed P automata

Distributed P automata

$$d\Pi = (V, \Pi_1, \ldots, \Pi_k, R)$$

where

$$\Pi_i = (V, \mu_i, P_{i,1}, \ldots, P_{i,m_i}, c_{i,0}, \mathcal{F}_i)$$

are P automata and

- R is a set of **inter-component** communication rules $((i,1), u/v, (j,1))$ with $u, v \in V^*,\ 1 \leq i, j \leq k$.
- $f = (f_1, \ldots, f_k)$ and

$$L(d\Pi, f) \quad = \quad \{w_1 \ldots w_k \in T^* \mid w_i \in f_i(v_{i,1}) \ldots f_i(v_{i,s_i}),\ 1 \leq i \leq k, \text{ where } v_{i,1}, \ldots, v_{i,s_i} \text{ is an accepted multiset sequence of the component } \Pi_i\}.$$

The power of dP automata

There are simple **context-sensitive** languages which **cannot be characterized** with f_{perm} .

$$\boxed{\mathcal{L}_{PERM}(dPA) \subset CS}$$

For example:

$$L = \{(wf(w))^n \mid w \in \{a,b\}^*, n \geq 2\} \notin \mathcal{L}_{PERM}(dPA)$$

where $f(a) = a', f(b) = b'$.

[Paun, Pérez-Jiménez 2010]

The power of dP automata

A **conjecture** from [Freund, Kogler,Paun, Pérez-Jiménez 2010]:

There are simple **linear** languages which cannot be characterized with f_{perm} .

Namely:

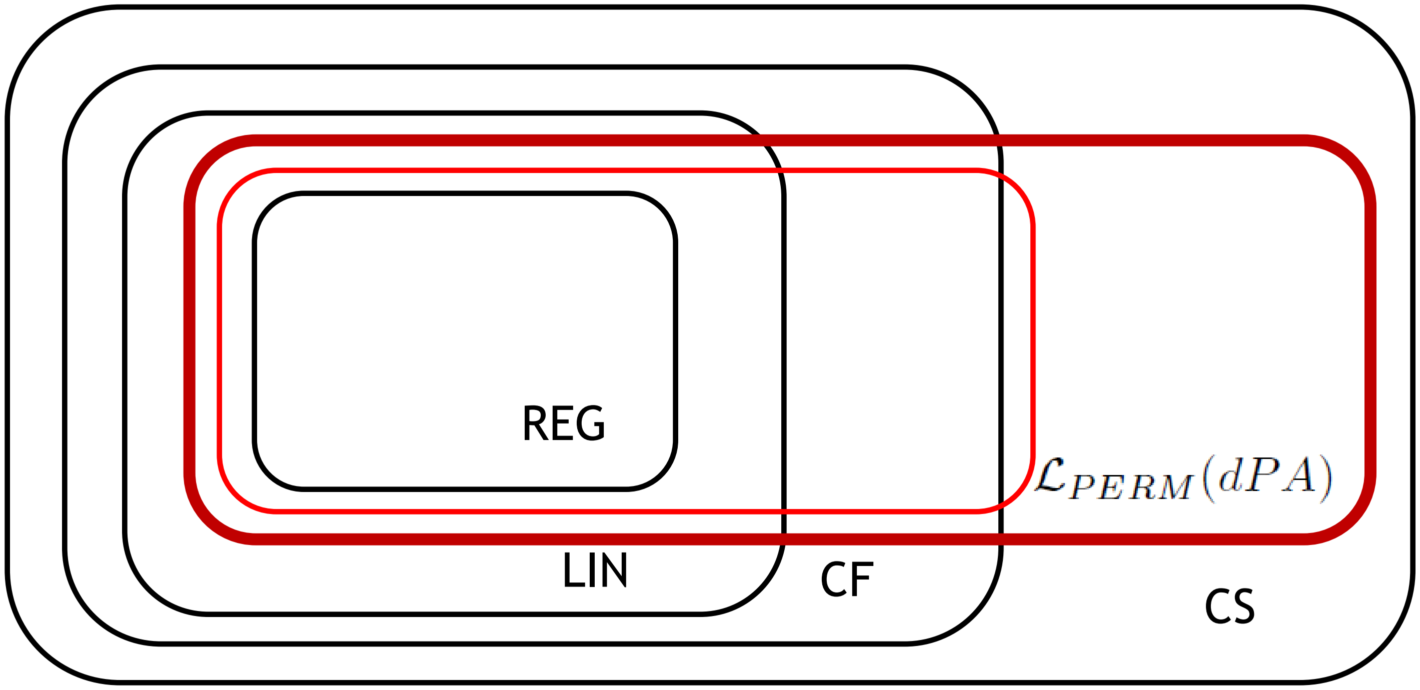
$$L = \{wcw^{-1} \mid w \in \{a,b\}^*, w^{-1} \text{ is the reverse of } w\} \notin \mathcal{L}_{PERM}(dPA)$$

On the other hand:

$$L = \{(ab)^n(ac)^n \mid n \geq 1\} \in \mathcal{L}_{PERM}(dPA)$$

The power of dP automata

With **permutation** mapping:



On the other hand: $\mathcal{L}_{TRANS}(dPA) = CS$

Distributed P automata

The parallelizability of languages

The parallelizability of languages

A **language** is (k, l, m)-efficiently **parallelizable** with respect to a class of **input mappings** F for some $k, m > 1,\ l \geq 1$, if

- L can be accepted with **balanced computations** of a dP automaton $d\Pi$ with k components such that it is $L(d\Pi, f)$ where $f \in F$ and $\mathrm{Com}(d\Pi) \leq l$,

- for **all** (non-distributed) **P automata** Π and input mapping $f' \in F$ such that $L = L(\Pi, f')$, we have

$$\lim_{x \in L, |x| \to \infty} \frac{\mathrm{time}_{\Pi}(x)}{\mathrm{time}_{d\Pi}(x)} \geq m$$

MTA SZTAKI

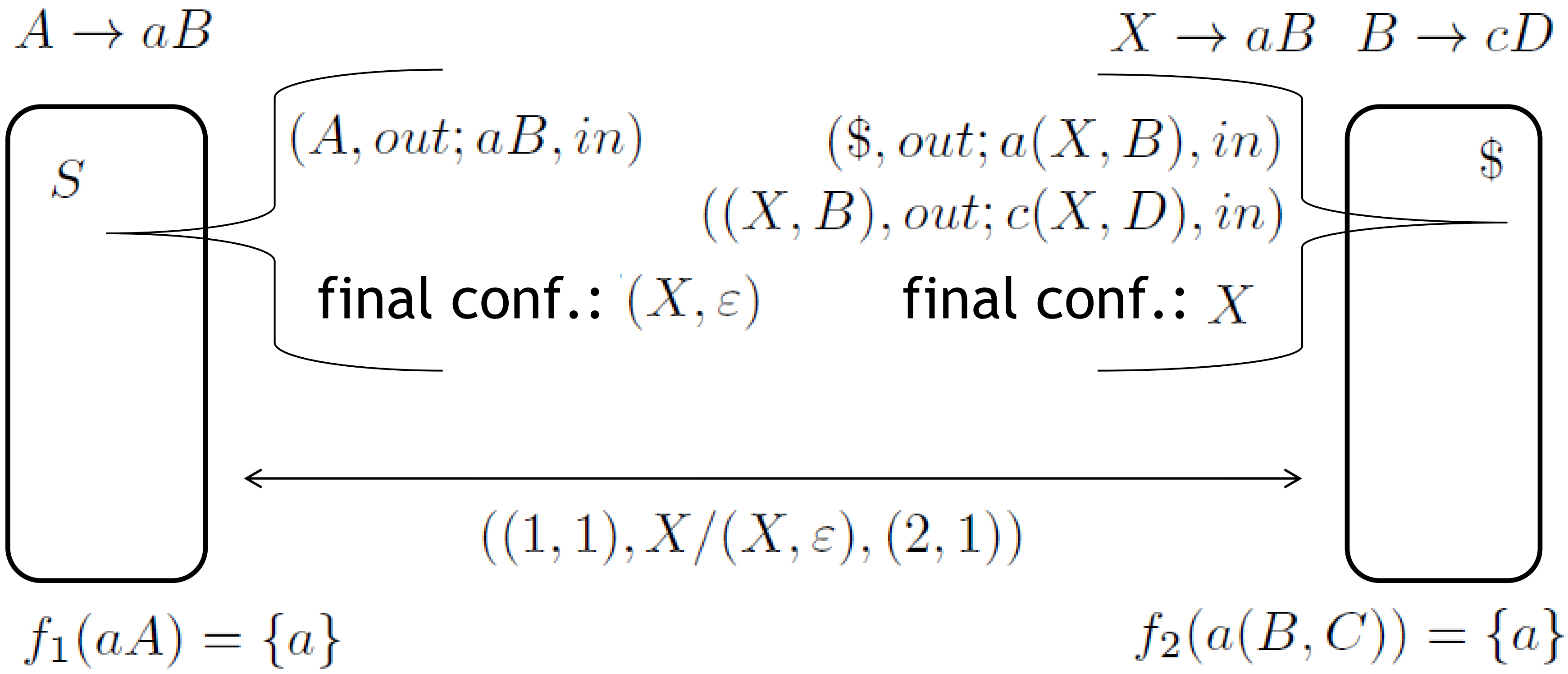
The parallelizability of languages

A **language** is (k, l, m)-efficiently **parallelizable** with respect to a class of **input mappings** F for some $k, m > 1, \ l \geq 1$, if it can be accepted by a dP automaton with k components, such that the dP automaton uses a **finite amount of communication** while being m times **faster** than **any non-distributed P automaton** which accepts L with **any input mapping** from the class F.

The parallelizability of REGular languages

1. All **regular languages** can be accepted by balanced computations of some dP automaton.

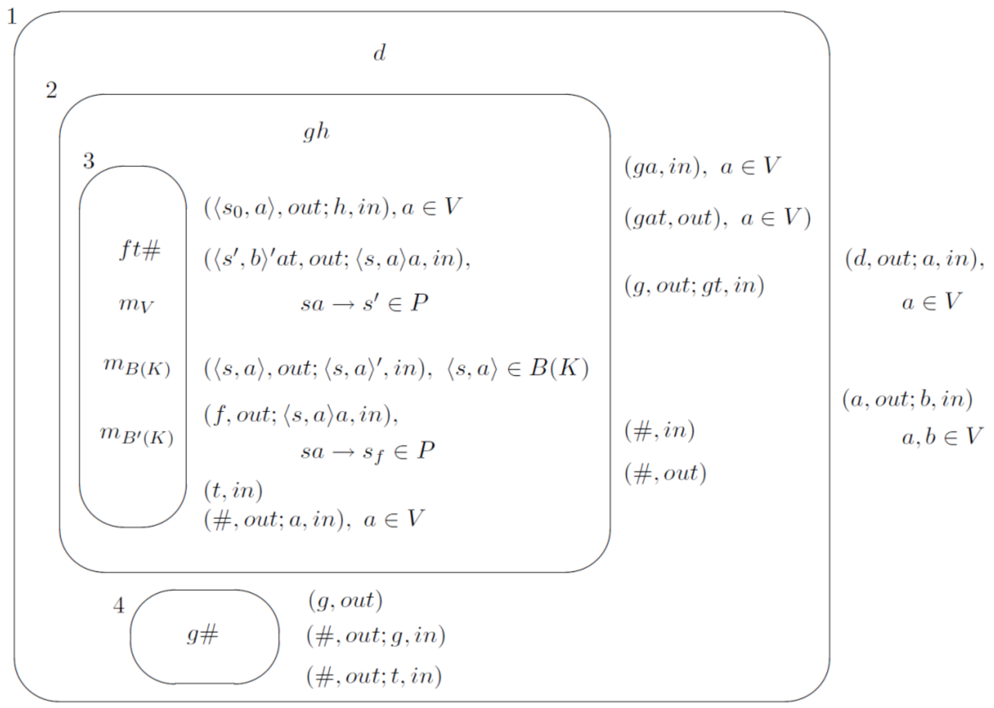
For **transduction** mappings:


$A \to aB$
$X \to aB \quad B \to cD$
S
$(A, out; aB, in)$
$(\$, out; a(X,B), in)$
$((X,B), out; c(X,D), in)$
$\$$
final conf.: (X, ε)
final conf.: X
$((1,1), X/(X,\varepsilon), (2,1))$
$f_1(aA) = \{a\}$
$f_2(a(B,C)) = \{a\}$

The parallelizability of regular languages

1. All **regular languages** can be accepted by balanced computations of some dP automaton.

For **permutation** mappings:



Example 3 can also be modified based on the same idea for a dP system with 2 components and $f_1 = f_2 = f_{PERM}$

The parallelizability of REG

1. All **regular languages** can be accepted by balanced computations of some dP automaton.

   We can use input mappings of **any type**.

2. What about the **efficiency** of parallelization ?

The parallelizability of regular languages

Efficiency with respect to f_{PERM} :

$$\boxed{\text{There are } (k,l,m)\text{-efficiently parallelizable regular languages with respect to } f_{PERM}.}$$

This holds because there are regular languages where the order of **no letters can be exchanged.**

➔**No two letters** can be read **in the same step.**

➔ There is **no P automaton** which needs **less steps** than the **number of letters.**

[Paun, Pérez-Jiménez 2010]

For **all** (non-distributed) **P automata** Π and input mapping $f' \in F$ such that $L = L(\Pi, f')$, we have

$$\lim_{x \in L, |x| \to \infty} \frac{\mathrm{time}_{\Pi}(x)}{\mathrm{time}_{d\Pi}(x)} \geq m$$

The parallelizability of regular languages

Efficiency with respect to input mappings $f \in TRANS$:

For any regular L and $c > 0$, there exists a P automaton Π such that $L = L(\Pi, f)$, $f \in TRANS$, and for any $w \in L$ with $|w| = n$, it holds that $time_{\Pi}(w) \leq c \cdot n$.

Take the **finite automaton** $M = (T, Q, q_0, \delta, F)$ with $L = L(M)$, it needs n steps.

Let $\Pi' = (Q \cup T \cup T', [\ [\ ]_2\ ]_1, P_1', P_2', c_0, \mathcal{F}), T' = \{\underline{ab} \mid a, b \in T\}$,

and $P_1' = \{(q''\underline{ab}, in; q, out) \mid q'' \in \delta(q', b) \text{ for some } q' \in \delta(q, a)\} \cup$

$\{(q'a, in; q, out) \mid q' \in \delta(q, a),\ q' \in F\}$,

$P_2' = \{(a, in) \mid a \in T \cup T'\}$.

with $f'(qa) = a$ and $f'(q\underline{ab}) = \underline{ab}$.

The parallelizability of regular languages

Efficiency with respect to $TRANS$:

There are **no** (k,l,m) -efficiently parallelizable regular languages with respect to $TRANS$.

This holds because with input mappings from $TRANS$, there is no “fastest” (non-distributed) P automata for a regular language.

Speedup with a linear factor is always possible which is a “problem”:

$$\lim_{x\in L,|x|\to\infty}\frac{\mathrm{time}_{\Pi}(x)}{\mathrm{time}_{d\Pi}(x)}\geq m$$

Real-time recognizable languages

A language L is **real-time recognizable** by a P automaton Π, if $L = L(\Pi, f)$ for some f, and Π reads a **nonempty input** multiset in **each step** of any computation accepting the words of L.

The parallelizability of real-time recognizable languages

There are **no** (k,l,m) -efficiently parallelizable real-time recognizable languages with respect to $TRANS$.

This holds because the **linear speedup** of P automata recognizing real-time recognizable languages is also **possible**.

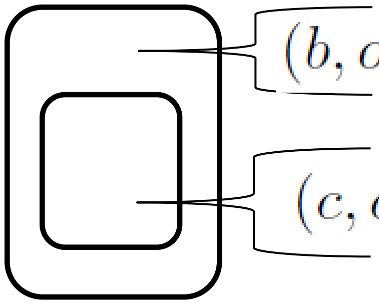
How to speed up real-time P automata

1. Simulate the real-time P automaton with **one membrane**

2. Apply the **same idea** as used in the **regular case** to speed up the one-membrane P automaton

Simulating a real-time P automaton with one membrane

$V = \{a, b, c\}$



$(b, out; a, in)$

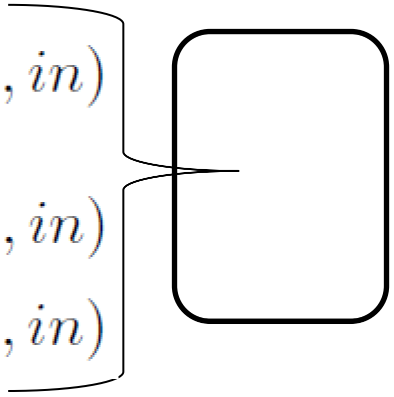
$(c, out; b, in)$

$V' = \{a, b, c, a_1, b_1, c_1, a_2, b_2, c_2\}$

$(b, out; a, in)$ $(b_1, out; a, in)$

$(bc_2, out; b_2c_1, in)$

$(b_1c_2, out; b_2c_1, in)$



The input mappings:

$$f \in TRANS$$
$$f(a) = \alpha$$

$$f' \in TRANS$$
$$f'(ab_2c_1) = f(ab_2c_1 \cap V) = f(a) = \alpha$$

How to speed up real-time P automata

2. Apply the **same idea** as used in the **regular case** to speed up the one-membrane P automaton

There are no (k,l,m)-efficiently parallelizable real-time recognizable languages with respect to TRANS

Let L be a language which is **real-time recognizable** by a P automaton Π_1, such that $L = L(\Pi_1, f)$ for some $f \in TRANS$, and let $c > 0$.

There exists a P automaton Π_2, such that $L = L(\Pi_2, f')$ for some $f' \in TRANS$, and for any $w \in L$, it holds that $time_{\Pi_2}(w) \leq c \cdot time_{\Pi_1}(w)$.

Distributed P automata

Are there other notions of efficient parallelizability?

Are there other reasonable notions of efficient parallelizability?

So far we had (k,l,m)-efficiency – the amount of inter-component **communication** is **finitely bounded.**

Consider $L = \{ww \mid w \in \{a,b,c\}^*,\ |w| = 3t \text{ for some } t \in \mathbb{N}\}$. It is not (k,l,m)-efficiently parallelizable since a **non-constant** amount of **communication** is needed.

What happens if the **communication** steps are **not** more “**expensive**” than the computational steps?

(k,m)-efficient parallelizability

Let $L = \{ww \mid w \in \{a,b,c\}^*,\ |w| = 3t \text{ for some } t \in \mathbb{N}\}$.

There are words in which no **two adjacent symbols** can be **exchanged** such that the result is still in L, a (non-distributed) P automaton with input mapping f_{perm} needs at least $n = 6t$ steps.

Let us take $d\Pi$ with **two components** which communicate after **every third** step:

$$\text{time}_{d\Pi}(w) + \text{Com}_{d\Pi}(w) = \frac{1}{2}n + t = 3t + t = 4t$$

(k,m)-efficient parallelizability

Let $L = \{ww \mid w \in \{a,b,c\}^*,\ |w| = 3t \text{ for some } t \in \mathbb{N}\}$.

There is a $d\Pi$ with two components, such that

$$\frac{\text{time}_{\Pi}(w)}{\text{time}_{d\Pi}(w) + \text{Com}_{d\Pi}(w)} \geq \frac{3}{2}$$

which is **efficient**, although the amount of **communication** is **not bounded** by any constant.

Besides (k,l,m)-efficient parallelizability, there are other efficiency notions

Are there languages which are efficiently parallelizable according to any of them considering input mappings form TRANS?

Does it make any difference, if the splitting of the input is done in a “more complicated” manner?

Conclusions

Conclusions

- **Distributed P systems** aim to employ parallelism in such a way that **different parts** of the **input** are processed **simultaneously** in **different parts/components** of the system
- It is natural to consider **distributed P automata**
- The properties of P automata are influenced by the way the **accepted word** is associated to the **accepted multiset sequence**
- This has **implications** also for the **distributed case**, not only for the **generative power**, but also for the **efficiency** of the parallelization of languages