A Connection Between Finite dP Automata and Multi-head Finite Automata

Erzsébet Csuhaj-Varjú and György Vaszil

Computer and Automation Research Institute
Hungarian Academy of Sciences

Kende u. 13-17, H-1111 Budapest, Hungary
{csuhaj,vaszil@sztaki.hu}

Outline of the talk

- P automata
- dP automata – distributed system of P automata
- Multi-head finite automata
- dP automata versus multi-head automata
- Conclusions, open problems

P automata

Automata-like
purely communicating P systems

[Csuhaj-Varjú, Vaszil, 2002]

Related model: analysing P system

[Freund, Oswald, 2002]

Motivation

Formulating and **studying** concepts of **accepting P systems** which combine characteristics of **conventional automata** (acceptors) and **P systems which are in interaction with their environments**.

Expected Results

- Obtaining a **better understanding** of the **theoretical principles, the driving forces behind natural processes**, and
- **enhancing and expanding the mathematical theory** of formal languages and **automata.**

P automaton

- P sytems **communicating** with their own **environments** through multisets of **objects** (input/output multisets of objects);
- inputs (input sequences) are distinguished as **accepted/not accepted inputs (input sequences)**

P automaton – the basic variant

1. **Symport/antiport** rules (with promoters and inhibitors) are used (rules for communicating objects)
2. Successful computations are defined by **final states** (accepting states)
3. The elements of the **input alphabet** are obtained by a mapping from the **multisets of objects** entering the system.

A P automaton

$$\Pi = (O, \mu, P_1, \ldots, P_k, c_0, \mathcal{F}),\ k \geq 1,$$

with

- **object** alphabet O,
- **membrane structure** μ,
- **sets of antiport rules** (possibly with promoters) P_i, $1 \leq i \leq k$,
- **initial configuration** $c_0 = (\mu, w_1, \ldots, w_k)$ where $w_i \in O^{(*)}$, $1 \leq i \leq k$, is the initial contents of the ith region,
- and set of **accepting configurations** $\mathcal{F}$ of the form $(\mu, v_1, \ldots, v_k)$, $v_i \in O^{(*)}$, $1 \leq i \leq k$.

The communication rules

- $(x, in)|_y$, $(x, out)|_y$ - symport
- $(x, in)|_{\overline{y}}$, $(x, out)|_{\overline{y}}$ - symport
- $(x, out; y, in)|_z$ - antiport
- $(x, out; y, in)|_{\overline{z}}$ - antiport

Symport/antiport rules with **promoters** and **inhibitors**

The rules are applied in **maximally parallel** or **sequential** manner.

Functioning of a P automaton

The **configurations** of the P automaton are **changed** by applying the rules in the **non-deterministic maximally parallel manner**.

This way, there is a **sequence of multisets** which **enter the system from the environment** during the steps of its computations.

If the computation **ends in an accepting configuration** from $\mathcal{F}$, then this multiset sequence is called an **accepted multiset sequence**.

Functioning of a P automaton - continued

If **P automaton** Π **accepts by final states**, then $\mathcal{F}$ is given as $E_1 \times \ldots \times E_k$, $E_i \subseteq O^{(*)}$, such that E_i is either a finite set of finite multisets, or $E_i = O^{(*)}$, $1 \leq i \leq k$.

If Π **accepts by halting**, then $\mathcal{F}$ consists of all halting configurations of Π, that is, of all configurations in which no rule can be applied in any of the regions.

Language of a P automaton

The **language accepted by P automaton** Π with respect to a **mapping**

$$f : O^{(*)} \rightarrow 2^{\Sigma^*}$$

is the f-image of **the set of multiset sequences accepted by** Π.

Remark on the language of a P automaton

The **choice of** f **essentially influences** the power of the **P automaton**.

In [Freund, Oswald, 2002] the authors consider a P automata variant with $f : O^{(*)} \rightarrow 2^{O^*}$ defined in such a way that a multiset over O is mapped by f to the set of strings which **consists of all permutations of the elements of the multiset**.

In the following, we will denote this **mapping by** f_{perm}.

Classical versus P automata

- P automata *resemble classical automata* in the sense that in *each computational step they receive input from their environment*, and *this input has influence on their operation*, i.e. the input changes their state and their functioning.

- One of significant characteristics which makes them *different* from classical automata is the property that the *workspace* they can *use for computation* is *provided by the objects of the already consumed input multisets*.

Classical versus P automata

- The *object of the computation and the machine which executes the computation cannot be separated in the same way* as in classical automata. This is a feature that can usually also be observed when we look at *natural processes as "computation"*.

The power of P automata

For linear space computable mapping f:

$$\mathcal{L}_{seq}(PA) = \mathcal{L}(restricted - 1LOG) \subset \mathcal{L}(1LOG))$$

$$\mathcal{L}_{par}(PA) = \mathcal{L}(restricted - 1LIN) = \mathcal{L}(1LIN) = \mathcal{L}(CS)$$

[Csuhaj-Varjú, Ibarra, Vaszil, 2004]

Computational Completeness

- Any **recursively enumerable language** can be obtained as the **language of a one-way P automaton** with 7 membranes **modulo a mapping.**

[Csuhaj-Varjú, Vaszil 2002]

- For a slightly modified definition: **One-way P automata** are able to recognize any **recursively enumerable language with 2 membranes, promoters of size 2, and with moved multisets of size 2.**

[Freund, Martín-Vide, Obtulowicz, Păun 2003]

Summary on P automata

Csuhaj-Varjú, E., Oswald, M., Vaszil, G.: P automata, In: *The Oxford Handbook of Membrane Computing*, (G. Păun, G. Rozenberg, A. Salomaa, Eds.), Chapter 6, Oxford University Press, 2010.

Distributed (d)P automaton

A **system** of a **finite number of P automata**

- which have **their own separate inputs** and
- which also **may communicate with each other** by **means of special antiport-like rules**.

[Păun, Pérez-Jiménez, 2010]

Aims

Study

- **communication complexity**
- **distribution**
- synchronization
- **relation to classical models of computing**

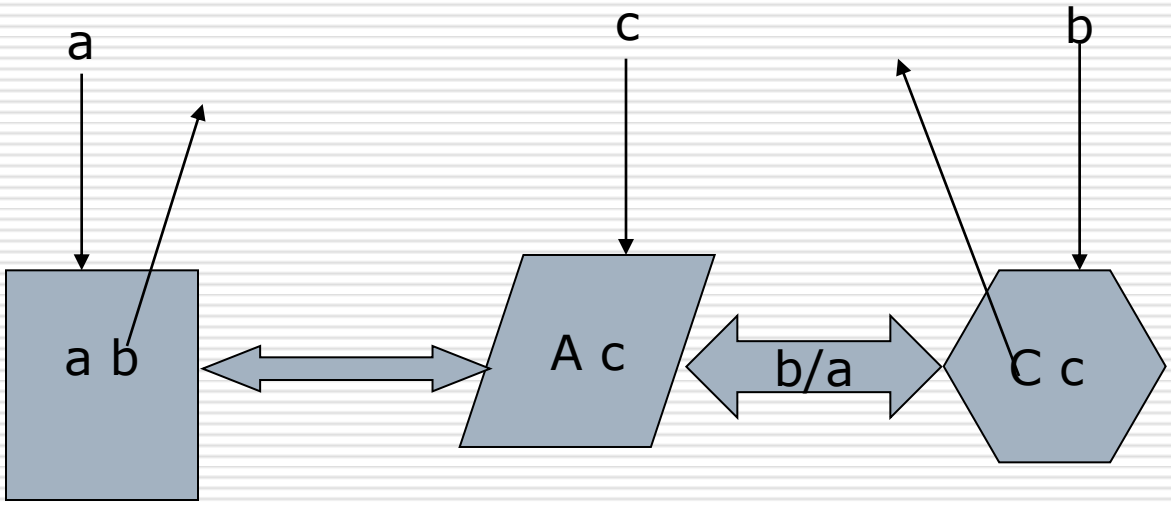
Distributed P (dP) automaton

$$\Delta = (O, \Pi_1, \ldots, \Pi_n, R, \mathcal{F}),\ n \geq 1,$$

- O is an alphabet, the **alphabet of objects**;
- $\Pi_i = (O, \mu_i, P_{i,1}, \ldots, P_{i,k_i}, c_{i,0}, \mathcal{F}_i)$ is a **P automaton** of degree $k_i \geq 1$, $1 \leq i \leq n$, called the ith **component of the system**;
- R is a **finite set of rules** of the form $(s_i, u/v, s_j)$, $1 \leq i, j \leq n$, $i \neq j$, $uv \in O^{(+)}$, called the set of **inter-component communication** (shortly, communication) rules of Δ; s_k, $1 \leq k \leq n$ denotes the skin membrane of Π_k;
- $\mathcal{F} \subseteq \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$, is called the **set of accepting configurations** of Δ.

An **inter-component communication rule** $(s_i, u/v, s_j)$, $1 \leq i, j \leq n$, $i \neq j$, realizes a **direct communication between components** Π_i and Π_j: a multiset u in the skin region of Π_i is exchanged with a multiset v in the skin region of Π_j.

A dP automaton



P automata communicating with each other and their environment

Configuration of a dP automaton

A **configuration** of Δ is

$$((\mu_1, u_{1,1}, \ldots, u_{1,k_1}), \ldots, (\mu_n, u_{n,1}, \ldots, u_{n,k_n})),$$

where $u_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq k_i$, is a multiset over O.

The **initial configuration** of Δ is the n-tuple

$$((\mu_1, w_{1,1}, \ldots, w_{1,k_1}), \ldots, (\mu_n, w_{n,1}, \ldots, w_{n,k_n})) = (c_{1,0}, \ldots, c_{n,0})$$

where $c_{i,0}$, $1 \leq i \leq n$, is the **initial configuration of component** Π_i.

Functioning

- The dP automaton functions by changing its configurations.
- The components work **synchronously**, governed by a global clock, **using the rules from their own rule sets** and **the set of communication rules** R **in the non-deterministic maximally parallel manner**.
- Each **component** Π_i, $1 \leq i \leq n$, **takes an input** (may be the empty multiset) from the environment, **works on it by using the rules** in sets $P_{i,1}, \ldots, P_{i,k_i}$ and **possibly communicates** with the other components.
- The **communication** is done by **means of rules** in R.

Configuration change

A **configuration** C **changes to configuration** C' by importing the n-tuple of multisets $(u_1, \ldots, u_n)$ from the environment, denoted by

$$(u_1, \ldots, u_n, C) \Longrightarrow C',$$

if C' **can be obtained from** C **by applying the rule sets of** Δ (including R) such that the skin membrane of component Π_i takes multiset u_i from the environment, i.e., u_i enters Π_i from the environment, $1 \leq i \leq n$.

Computation

A **computation** in Δ is a sequence of configurations following each other, starting from the initial configuration.

It is **successful** if it enters one of the accepting configurations of $\mathcal{F} \subseteq \mathcal{F}_1 \times \ldots \times \mathcal{F}_n$.

If the components accept by **final states**, then $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$, or if Δ **accepts by halting**, then $\mathcal{F} \subseteq \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$, it contains the direct product of those halting configurations of the components which are also halting configurations of Δ.

Acceptance

A dP automaton Δ **accepts** the n-tuple

$$(\alpha_1, \ldots, \alpha_n),$$

where α_i, $1 \leq i \leq n$, is a sequence of multisets over O, if the component Π_i, **starting from its initial configuration**, using the symport/antiport rules as well as the inter-component communication rules in the non-deterministic maximally parallel way, **takes from the environment the multiset sequence** α_i, $1 \leq i \leq n$, and Δ eventually **enters an accepting configuration**.

Language of a dP automaton

The **(concatenated) language** of Δ over an alphabet Σ with respect to the mapping $f = (f_1, \ldots, f_n)$ for $f_i : O^{(*)} \to 2^{\Sigma^*}$, $1 \leq i \leq n$, is defined as

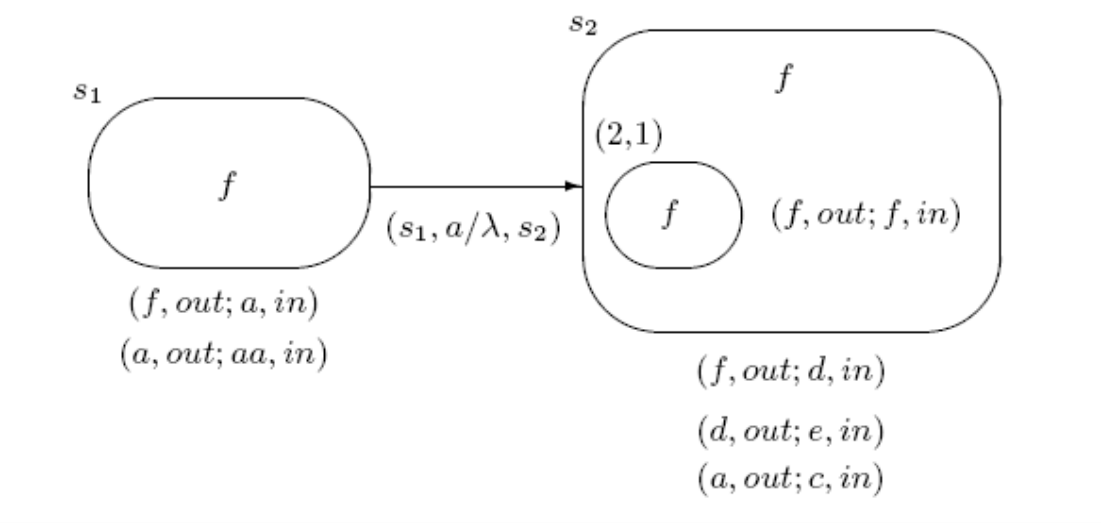
$$L_{concat}(\Delta, f, \Sigma) = \{w_1 \ldots w_n \in \Sigma^* \mid w_i = f_i(v_{i,1}) \ldots f_i(v_{i,s_i}) \text{ and } \alpha_i = v_{i,1} \ldots v_{i,s_i},\ 1 \leq i \leq n, \text{ for an } n\text{-tuple of accepted multiset sequences } (\alpha_1, \ldots, \alpha_n)\}.$$

The words accepted by the components are concatenated to obtain the words of the language accepted by the dP automata.

[Păun, Pérez-Jiménez, 2010]

In this seminal article mapping f_{perm} was used.

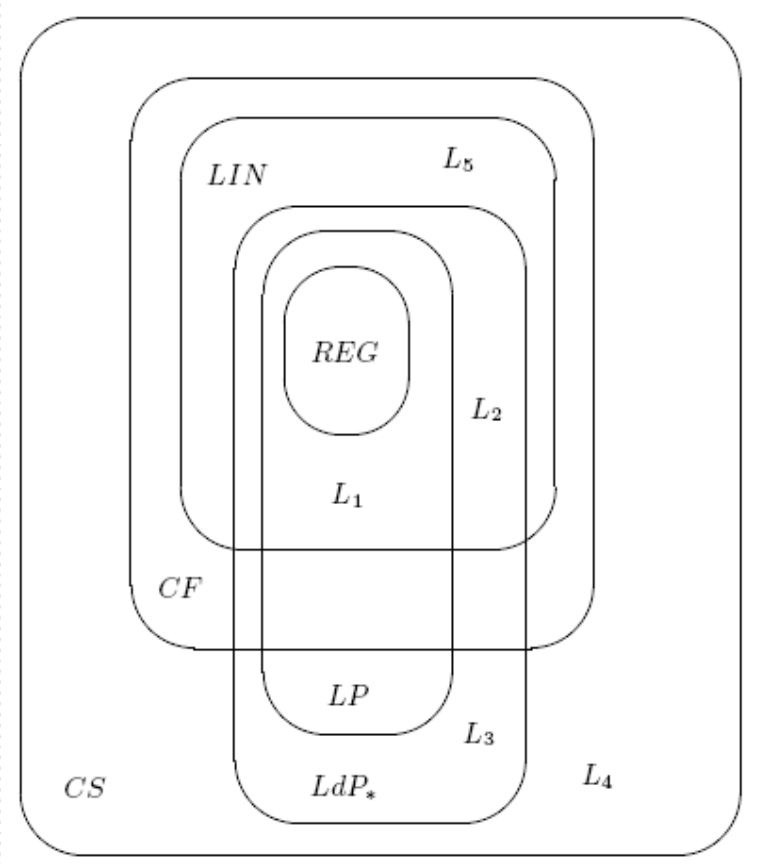
Example



$$L(\Delta) \cap a^*dc^*e = \{a^{2n}dc^{2n}e \mid n \geq 0\}$$

[Păun,Pérez-Jiménez, 2011]

Results on the concatenated languages of dP automata



[Păun, Pérez-Jiménez, 2011]

A variant of dP automata

A dP automaton Δ is called **finite**, if the **number of configurations reachable from its initial configuration is finite**.

[Păun, Pérez-Jiménez, 2010]

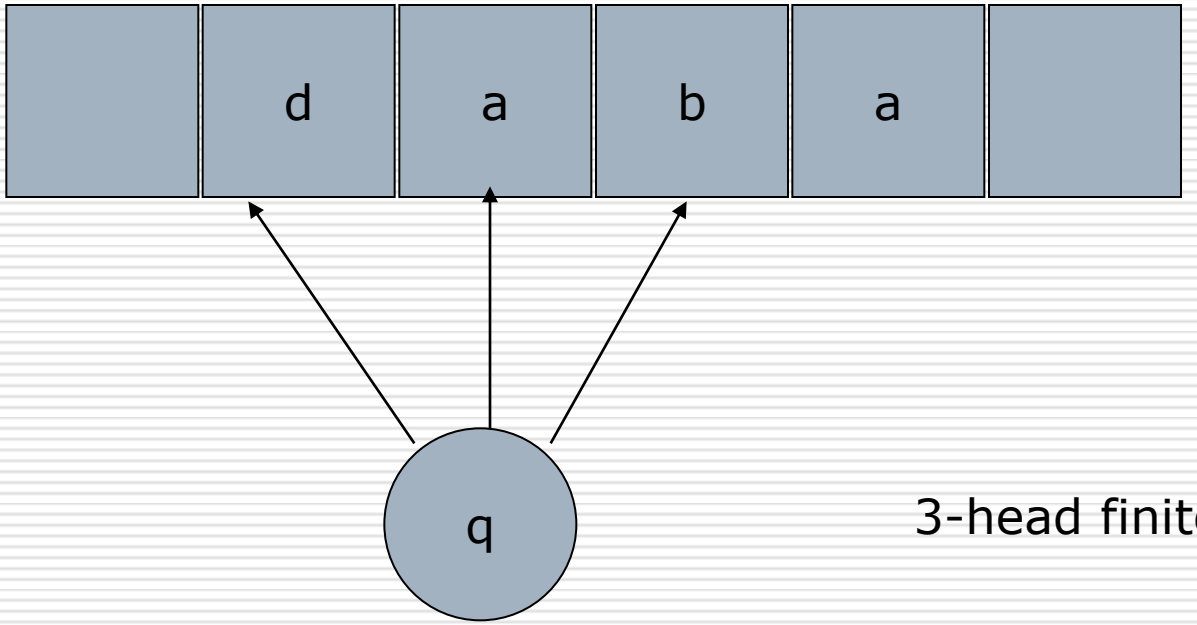
dP automata versus classical automata

Finite dP automaton resembles to multi-tape (multi-head) finite automaton

Multi-head finite automaton

- **Natural extension** of a **finite automaton**
- It may have **more than one heads reading the same input word**, the heads may scan the input symbol and move when the state of the automaton changes.
- **Acceptance is defined as in the one-head case**: an input string is accepted if starting from the beginning of the word with all heads (that never leave the input word), the automaton enters an accepting state.
- Analogously to the one-head case, **deterministic** and **non-deterministic, one-way** and **two-way variants** are considered. (If the heads are allowed to move in both directions, the automaton is called two-way, if only from left to right, then one-way.)

Multi-head finite automaton



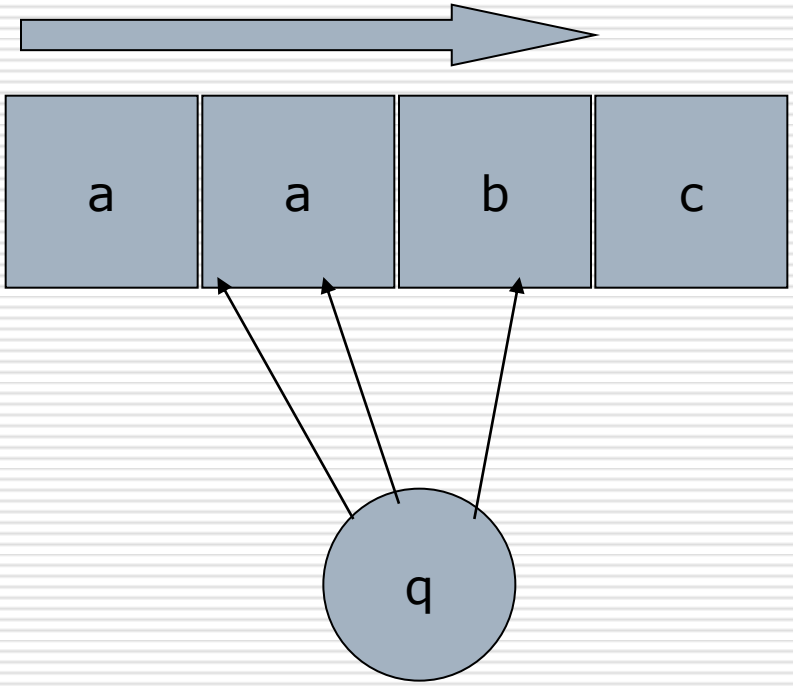
3-head finite automaton

[Rabin, Scott, 1964]

[Rosenberg, 1966]

Non-deterministic one-way k-head finite automaton (a 1NFA(k))

$$M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$$

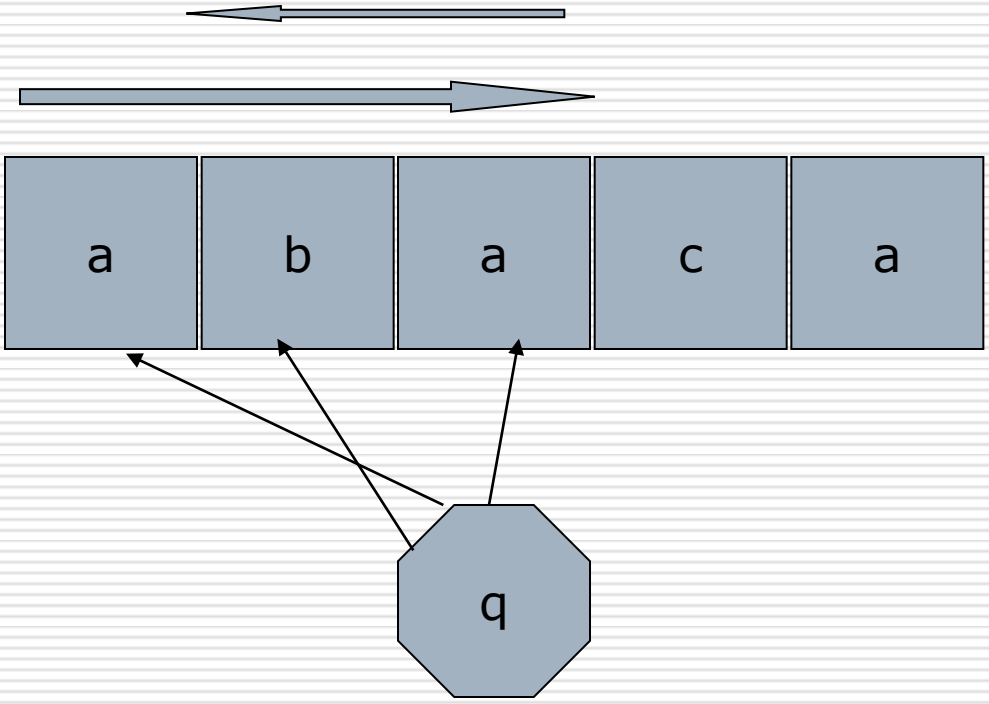


3-head one-way finite automaton

movement in one direction

Non-deterministic two-way k-head finite automaton (a 2NFA(k))

$$M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$$



3-head two-way finite automaton

movement in two directions

Configuration of a 2NFA(k)

A **configuration** of a 2NFA(k) $M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$ is a triplet

$$c = (w, q, p),$$

where $w \in \Sigma^*$ is the **input**, $q \in Q$ is the **current state**, and $p = (p_1, \ldots, p_k) \in \{0, 1, \ldots, |w| + 1\}^k$ gives the **current head positions**.

If a position p_i is 0, then head i is scanning symbol $\triangleright$, if $1 \leq p_i \leq |w|$, then head i scans the p_ith letter of w, and if it is $|w| + 1$, then the head is scanning symbol $\triangleleft$.

The **initial configuration for an input** w is $(w, q_0, (1, \ldots, 1))$, that is, a 2NFA(k) starts computing the word with all of its heads on the first tape cell of the tape.

Computation by a 2NFA(k)

Direct change of configurations

Let $w = a_1 \dots a_n$, be the **input**, $a_0 = \triangleright$, $a_{n+1} = \triangleleft$.

For two **configurations** $c_1 = (w, q, (p_1, \dots, p_k))$ and $c_2 = (w, q', (p'_1, \dots, p'_k))$ we say that c_2 **directly follows** c_1, denoted by

$$c_1 \vdash c_2,$$

if $p'_i = p_i + d_i$, $1 \leq i \leq k$, where $(q', (d_1, \dots, d_k)) \in \delta(q, (a_{p_1} \dots a_{p_k}))$.

Language of a 2NFA(k)

The **language** $L(M)$ accepted by a 2NFA(k) M is the set of words w such that there is a computation which starts with $\triangleright w \triangleleft$ on the input tape and ends when M reaches an accepting state.

$$L(M) = \{w \in \Sigma^* \mid (w, q_0, (1, \ldots, 1)) \vdash^* (w, q_f, (p_1, \ldots, p_k)),\ q_f \in F\}.$$

Results on multi-head finite automata

$$\mathsf{L} = \bigcup_{k\geq 1} \mathscr{L}(2DFA(k)) \text{ and } \mathsf{NL} = \bigcup_{k\geq 1} \mathscr{L}(2NFA(k)).$$

L=DSPACE(log(n)) and NL=NSPACE(log(n))

$$\mathscr{L}(1DFA(k)) \subset \mathscr{L}(1DFA(k+1)) \text{ and } \mathscr{L}(1NFA(k)) \subset \mathscr{L}(1NFA(k+1))$$

k≥1

$$\mathscr{L}(1DFA(k)) \subset \mathscr{L}(2DFA(k)), \quad \mathscr{L}(1DFA(k)) \subset \mathscr{L}(1NFA(k)),$$

k≥2

$$\mathscr{L}(2DFA(k)) \subset \mathscr{L}(2DFA(k+1)) \text{ and } \mathscr{L}(2NFA(k)) \subset \mathscr{L}(2NFA(k+1)).$$

k≥1

Resemblence to finite dP automata

- The **configurations** of the finite dP automaton correspond to the **states** of the finite multi-head automaton,

- the **input** of the multi-head finite automaton corresponds to the (mapping of the) **input sequence of a (any) component P automaton**.

Strong agreement language of a dP automaton

The language contains the **words** which **can be accepted by all components**, **all of them reading** (i.e. having as input) **the same multiset sequence** during a successful computation.

The **strong agreement language** of a dP automaton Δ over an alphabet Σ with respect to a mapping $f = (g, \ldots, g)$ for $g : O^{(*)} \rightarrow 2^{\Sigma^*}$, is defined as

$$L_{s,agree}(\Delta, f, \Sigma) = \{w \in \Sigma^* \mid w = g(v_1) \ldots g(v_s) \text{ and } \alpha = v_1 \ldots v_s, \text{ for an } n\text{-tuple of accepted multiset sequences } (\alpha, \ldots, \alpha) \text{ of } \Delta\}.$$

Weak agreement language of a dP automaton

The language consists of **all strings which can be accepted by all of the components** simultaneously **without requiring that the accepted multiset sequences are also the same**.

The **weak agreement language** of Δ over an alphabet Σ with respect to a mapping $f = (g, \ldots, g)$ for $g : O^{(*)} \rightarrow 2^{\Sigma^*}$, is defined as

$$L_{w,agree}(\Delta, f, \Sigma) = \{w \in \Sigma^* \mid w = g(\alpha_i), 1 \leq i \leq n, \text{ for an } n\text{-tuple of accepted multiset sequences } (\alpha_1, \ldots, \alpha_n) \text{ of } \Delta\}.$$

One-way multi-head finite automata versus finite dP automata

The **weak agreement language** of a finite dP automaton (with respect to the mapping f_{perm}) is equal to the **language of a one-way multi-head finite automaton**.

Theorem

For any **finite dP automaton** $\Delta = (O, \Pi_1, \ldots, \Pi_k, R, \mathcal{F})$, $k \geq 2$, a **non-deterministic one-way** k**-head finite automaton** $M = (Q, O, k, \delta, \triangleright, \triangleleft, q_0, F)$ can be constructed such that $L_{w,agree}(\Delta, f_{perm}, O) = L(M)$.

Proof idea

$\Delta = (O, \Pi_1, \ldots, \Pi_k, R, \mathcal{F}),\ k \geq 2$

finite dP automaton

$M = (Q, O, k, \delta, \triangleright, \triangleleft, q_0, F)$

one-way multi-head finite automaton

The states of M, which govern the computation, represent the transitions of Δ.

Any transition in Δ corresponds to a set of transitions sequences in M, where the number of steps performed by M is equal to the maximum of the number of objects imported by the components of Δ in the simulated transition.

One-way multi-head finite automata versus finite dP automata

The **language of any one-way finite multi-head automaton** can be obtained as the **strong or weak agreement language** of a finite dP automaton with respect to the mapping f_{perm}.

Theorem

For any **non-deterministic one-way k-head finite automaton** M, $k \geq 2$, with input alphabet Σ we can construct a dP automaton Δ of degree k, such that

$$L(M) = L_{s,agree}(\Delta, f_{perm}, \Sigma) = L_{w,agree}(\Delta, f_{perm}, \Sigma).$$

Proof idea

$M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$ one-way multi-head finite automaton

$\Delta = (O, \Pi_1, \ldots, \Pi_k, R, \mathcal{F}),$ finite dP automaton

The **idea of the proof** is that the **transitions** in M are simulated by Δ **in a cycle**, the **work of each reading head of** M **is executed by a different component**.

Two-way multi-head finite automata versus finite dP automata

We introduce the notion of a **two-way dP automaton** and we show how **two-way finite dP automata characterize** the **language family accepted by non-deterministic two-way multi-head finite automata.**

We first recall that alphabets of the form $\Sigma \cup \bar{\Sigma}$, where Σ is an alphabet itself and $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ are called double alphabets.

Two-way trail

$$w = w_1 \bar{x}_1 \bar{w}_2 x_2 w_3 \bar{x}_3 \bar{w}_4 x_4 \ldots \bar{x}_{n-2} \bar{w}_{n-1} x_{n-1} w_n$$

1. For all w' with $w = w'w''$, it holds that $|w'|_{\Sigma} \geq |w'|_{\bar{\Sigma}}$

It describes that the head never moves to the left of the left endmarker.

2. for all w'' with $w = w'w''$, it holds that $|w'|_{\Sigma} \geq 2 \cdot |w'|_{\bar{\Sigma}}$;

It guarantees that the head finishes its movement on the last symbol

Two-way trail

$$w = w_1\bar{x}_1\bar{w}_2x_2w_3\bar{x}_3\bar{w}_4x_4\ldots\bar{x}_{n-2}\bar{w}_{n-1}x_{n-1}w_n$$

3. for all $1 \leq i \leq n-2$, the subwords $w_i\bar{x}_i\bar{w}_{i+1}$ are of the form $w'_ix\bar{x}_i\bar{x}\bar{w}'_{i+1}$ where $w_i = w'_ix$, $\bar{w}_{i+1} = \bar{x}\bar{w}'_{i+1}$, and the subwords $\bar{w}_{i+1}x_{i+1}w_{i+2}$ are of the form $\bar{w}'_{i+1}\bar{x}x_{i+1}xw'_{i+2}$ where $\bar{w}_{i+1} = \bar{w}'_{i+1}\bar{x}$, $w_{i+2} = xw'_{i+2}$;

It identifies the subwords which describe the turn of the direction.

4. for all $1 \leq i \leq n-2$, the subwords $w_i\bar{x}_i\bar{w}_{i+1}$ satisfy one of the properties (a) $\bar{w}_{i+1} = \bar{w}_i^R\bar{w}'_{i+1}$, or (b) $w_i = w_{i+1}^Rw'_i$, while the subwords $\bar{w}_{i+1}x_{i+1}w_{i+2}$ satisfy one of the properties (c) $w_{i+2} = w_{i+1}^Rw'_{i+2}$, or (d) $\bar{w}_{i+1} = \bar{w}_{i+2}^R\bar{w}'_{i+1}$, depending on the length of w_i, w_{i+1}, and w_{i+2}.

It requires that after turning, the same symbols are read in the inverse order as the ones that were read before the turn.

Two-way trails and 2NFA(k)

Using the notion of the **two-way trail**, it is easy to see that a **computation in a 2NFA(k)** $M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$ **is accepting** if and only if it is given by a sequence states $q_0, q_1, \ldots, q_s$, $q_s = q_f \in F$, where $q_j \in \delta(q_{j-1}, x_{j-1,1}, \ldots, x_{j-1,k})$, $1 \leq j \leq s-1$ such that $w = x_{1,i} \ldots x_{s,i} \in (\Sigma \cup \bar{\Sigma})^*$ is a **two-way trail** and $x_{j,i} \in (\Sigma \cup \bar{\Sigma} \cup \{\lambda\})$ for $1 \leq j \leq s$, $1 \leq i \leq k$.

The concept of a two-way trail can be extended to the concept of a two-way multiset trail in an obvious manner.

Two-way dP automaton

A dP automaton $\Delta = (O', \Pi_1, \ldots, \Pi_k, R, \mathcal{F})$ where $O' = O \cup \bar{O}$ is a **double alphabet** is called a **two-way dP automaton** if any multiset u_i which enters component Π_i, $1 \leq i \leq k$, in the course of a computation consists of either objects of O, or objects of $\bar{O}$, or it is the empty multiset.

Obviously, if a two-way dP automaton is a finite dP automaton, then we speak of a two-way finite dP automaton.

Reduction mapping for two-way trails

$$h : (\Sigma \cup \bar{\Sigma})^* \rightarrow 2^{(\Sigma \cup \bar{\Sigma})^*}$$

- $w_1 \bar{x}_1 w_2 \in h(w)$ if and only if $w = w_1 x_1 \bar{x}_2 \bar{x}_1 w_2$, for some $x_1, x_2 \in \Sigma$, or
- $w_1 x_1 w_2 \in h(w)$ if and only if $w = w_1 \bar{x}_1 x_2 x_1 w_2$, for some $x_1, x_2 \in \Sigma$, or
- $h(w) = w$ if and only if $w \in \Sigma^*$, or
- let $h(w)$ be undefined otherwise.

Let us define $h^0(w) = h(w)$, $h^i(w) = h(h^{i-1}(w))$ for $i \geq 0$, and let $h^*(w) = h^i(w)$ for some i, such that $h^i(w) = h^{i+1}(w)$.

Languages of two-way dP automata

Strong agreement language

$$L_{s,agree}(\Delta, f, \Sigma) = \{w \in \Sigma^* \mid w = g(v_1) \dots g(v_s), \alpha = v_1 \dots v_s,$$
$$\text{where } v_j \in O^{(*)}, 1 \leq j \leq s, \text{ and } \alpha = h_m^*(\beta_i),$$
$$\text{for a } k\text{-tuple of accepted multiset sequences } (\beta_1, \dots, \beta_k),$$
$$\text{where } \beta_i \text{ is a two-way multiset trail}, 1 \leq i \leq k\}.$$

Languages of two-way dP automata

Weak agreement language

$$L_{w,agree}(\Delta, f, \Sigma) = \{w \in \Sigma^* \mid w = h^*(u_i), 1 \leq i \leq k, u_i = g(v_{1,i}) \dots g(v_{s_i,i}),$$
$$\text{is a two-way trail, } \alpha_i = v_{1,i} \dots v_{s_i,i},\ s_i \geq 1, (\alpha_1, \dots, \alpha_k)$$
$$\text{is a } k\text{-tuple of multiset sequences accepted by } \Delta\}.$$

Results

Theorem

Any language that can be accepted by a non-deterministic two-way k-head finite automaton for $k \geq 2$ is equal to the strong or weak agreement language of a two-way finite dP automaton of degree k with respect to the mapping f_{perm}.

Theorem

Any language which is the **weak agreement language with respect to the mapping f_{perm} of a two-way finite dP automaton** of degree k, $k \geq 2$, **can be accepted by a non-deterministic two-way k-head finite automaton**.

Conclusions, Open problems

1. NSPACE(log n) can be characterized by finite dP automata
2. Decidability question concerning multi-head finite automata and their language classes can be examined in the frame of finite dP automata.

References

1. Birget, J.-C.: Two-way automaton computations, *RAIRO Informatique Théorique et Applications*, **24**, 1990, 44-66.
2. Csuhaj-Varjú, E., Ibarra, O.H., Vaszil, G.: On the computational complexity of P automata. *Natural Computing*, **5(2)**, 2006, 109–126.
3. Csuhaj-Varjú, E., Vaszil, G.: P automata or purely communicating accepting P systems, In: *Membrane Computing, International Workshop, WMC-CdeA, Curtea de Arges, Romania, August 19-23, 2002, Revised Papers* (G. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds.), vol. 2597 of *Lecture Notes in Computer Science*, Springer, Berlin, 2003, 219–233.
4. Csuhaj-Varjú, E., Oswald, M., Vaszil, G.: P automata, In: *The Oxford Handbook of Membrane Computing*, (G. Păun, G. Rozenberg, A. Salomaa, Eds.), Chapter 6, Oxford University Press, 2010.
5. Freund, R., Kogler, M., Păun, G., Pérez-Jiménez, M. J.: On the power of P and dP automata. *Annals of Bucharest University. Mathematics-Informatics Series*, **63**, 2009, 5–22.
6. Hartmanis, J.: On non-determinancy in simple computing devices, *Acta Informatica*, **1**, 1972, 336-344.
7. Holzer, M., Kutrib, M., Malcher, A.: Complexity of multi-head finite automata: Origins and Directions, *Theoretical Computer Science*, **412**, 2011, 83–96.
8. Păun, G.: *Membrane Computing: An Introduction*, Natural Computing Series, Springer-Verlag, Berlin, 2002.