Verifying APCol systems

Ludek Cienciala[1] Lucie Ciencialová[1] Erzsébet Csuhaj-Varjú [2]
György Vaszil[3]

CMC 19, September 7, 2018, Dresden Germany

[1]Institute of Computer Science and Research Institute of the IT4Innovations Centre of Excellence, Silesian University in Opava, Czech Republic
lucie.ciencialova@fpf.slu.cz

[2]Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
csuhaj@inf.elte.hu

[3]Faculty of Informatics, University of Debrecen, Hungary
vaszil.gyorgy@inf.unideb.hu

Outline

Introduction

APCol systems

APCol systems (Automaton-like P colonies)

were introduced in[1] as an extension of P colonies[2] - a very simple variant of membrane systems inspired by colonies of formal grammars.

[1] L. Cienciala, L. Ciencialová, and E. Csuhaj-Varjú. "Towards on P colonies processing strings". In: *Proc. BWMC 2014, Sevilla, 2014*. Sevilla, Spain: Fénix Editora, 2014, pp. 102–118.

[2] J. Kelemen, A. Kelemenová, and Gh. Păun. "Preview of P colonies: A biochemically inspired computing model". In: *Workshop and Tutorial Proceedings. Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)*. Boston, Mass, 2004, pp. 82–86.

Introduction

An APCol system consists of

- a finite number of components called agents - finite collections of objects embedded in a membrane
- a shared environment, that is represented by a string.

Agents

- equipped with programs which are composed from rules that allow them to interact with their environment.
- Capacity - the number of objects inside each agent is constant and it is usually a very small number: 1, **2** or 3.

Introduction

Environment

- The environmental string is processed by the agents
- It is used as a communication channel for the agents as well. Through the string, the agents are able to affect the behaviour of another agent.

Introduction

The activity of the agents is based on rules[3].

Rules

[3] J. Kelemen, A. Kelemenová, and Gh. Păun. "Preview of P colonies: A biochemically inspired computing model". In: *Workshop and Tutorial Proceedings. Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)*. Boston, Mass, 2004, pp. 82–86.

Introduction

The activity of the agents is based on rules.

Rules

- Rewriting rule $a \rightarrow b$ - rewrite (evolve) one object a to object b. Both objects are placed inside the agent.

Rewriting rule $a \rightarrow b$

$wdab \ldots$

Introduction

The activity of the agents is based on rules.

Rules

- Rewriting rule $a \rightarrow b$ - rewrite (evolve) one object a to object b. Both objects are placed inside the agent.

Rewriting rule $a \rightarrow b$

$wdab \ldots$

Introduction

The activity of the agents is based on rules.

Rules

- Rewriting rule $a \rightarrow b$ - rewrite (evolve) one object a to object b. Both objects are placed inside the agent.
- Communication rule $c \leftrightarrow d$ - exchange object c placed inside the agent with object d in the string.

Communication rule $c \leftrightarrow d$

$wdab \ldots$

Introduction

The activity of the agents is based on rules.

Rules

- Rewriting rule $a \rightarrow b$ - rewrite (evolve) one object a to object b. Both objects are placed inside the agent.
- Communication rule $c \leftrightarrow d$ - exchange object c placed inside the agent with object d in the string.

Communication rule $c \leftrightarrow d$

$wcab \ldots$

Introduction

Programs

The rules are combined into programs in such a way that all objects inside the agent are affected by execution of the rules.

The number of rules in the program is the same as the number of objects inside the agent.

Definition

APCol systems

Definition (APCol system[3])

An APCol system is a construct

$$\Pi = (O, e, A_1, \ldots, A_n), \text{ where}$$

- O is an alphabet; its elements are called the objects,
- $e \in O$, called the basic object,
- $A_i,\ 1 \leq i \leq n$, are agents.

[3] L. Cienciala, L. Ciencialová, and E. Csuhaj-Varjú. "Towards on P colonies processing strings". In: *Proc. BWMC 2014, Sevilla, 2014*. Sevilla, Spain: Fénix Editora, 2014, pp. 102–118.

APCol systems

Definition (Agent)

Agent is a triplet $A_i = (\omega_i, P_i, F_i)$, where

- ω_i is a multiset over O, describing the initial state (content) of the agent, $|\omega_i| = 2$,
- $P_i = \{p_{i,1}, \ldots, p_{i,k_i}\}$ is a finite set of programs associated with the agent, where each program is a pair of rules. Each rule is in one of the following forms:
  - $a \to b$, where $a, b \in O$, called an evolution rule,
  - $c \leftrightarrow d$, where $c, d \in O$, called a communication rule,
- $F_i \subseteq O^*$ is a finite set of final states (contents) of agent A_i,

APCol systems

Context programs

Both rules in a program can be communication rules, an agent can work with two objects in the string in one step of the computation. The agent can act only in one place in a computation step and the change of the string depends both on the order of the rules in the program and on the interacting objects.

- $\langle a \leftrightarrow b; c \leftrightarrow d \rangle$ - $[ac]$ $wbdw' \Rightarrow [bd]$ $wacw'$
- $\langle c \leftrightarrow d; a \leftrightarrow b \rangle$ - $[ac]$ $wdbw' \Rightarrow [bd]$ $wcaw'$
- $\langle a \leftrightarrow b; c \leftrightarrow e \rangle$ - $[ac]$ $wbw' \Rightarrow [be]$ $wacw'$
- $\langle c \leftrightarrow e; a \leftrightarrow b \rangle$ - $[ac]$ $wbw' \Rightarrow [be]$ $wcaw'$

APCol systems

Context programs

Both rules in a program can be communication rules, an agent can work with two objects in the string in one step of the computation. The agent can act only in one place in a computation step and the change of the string depends both on the order of the rules in the program and on the interacting objects.

- $\langle a \leftrightarrow e; c \leftrightarrow e \rangle$ - $[ac] \quad ww' \Rightarrow [ee] \quad wacw'$
- $\langle e \leftrightarrow b; e \leftrightarrow d \rangle$ - $[ee] \quad wbdw' \Rightarrow [bd] \quad ww'$
- $\langle e \leftrightarrow d; e \leftrightarrow b \rangle$ - $[ee] \quad wdbw' \Rightarrow [ee] \quad ww'$
- $\langle e \leftrightarrow e; e \leftrightarrow d \rangle$; $\langle e \leftrightarrow e; c \leftrightarrow d \rangle$, . . . - these programs can be replaced by programs of type $\langle e \rightarrow e; c \leftrightarrow d \rangle$.

APCol systems

Configutation of an APCol system

A configuration of an APCoL system Π is given by $(w; w_1, \ldots, w_n)$, where $|w_i| = 2,\ 1 \leq i \leq n$, w_i represents all the objects placed inside the i-th agent and $w \in (O - \{e\})^*$ is the string to be processed.

Initial configuration

Aan initial configuration of the APCol system is an $(n+1)$-tuple $c = (\omega; \omega_1, \ldots, \omega_n)$ where ω is the initial state of the environment and the other n components are multisets of strings of objects, given in the form of strings, the initial states the of agents.

APCol systems

Computational step

At each step of the computation every agent attempts to find one of its programs to use. If the number of applicable programs is higher than one, the agent non-deterministically chooses one of them. At every step of computation, the maximal possible number of agents have to perform a program.

Computation, halting computation

By applying programs, the automaton-like P colony passes from one configuration to another configuration. A sequence of configurations starting from the initial configuration is called a computation. A configuration is halting if the APCol system has no applicable program.

APCol systems

Accepting mode

In the case of accepting mode, a computation is called accepting if and only if:

- it starts with string to be processed as an initial content of the environment
- the computation is halting
- at least one agent is in final state
- the environmental string is reduced to ε

APCol systems

The results about accepting power of APCol systems[4]:

- The family of languages accepted by jumping finite automata[5] is properly included in the family of languages accepted by APCol systems with one agent
- any recursively enumerable language can be obtained as a projection of a language accepted by an APCol system with two agents.

[4] L. Cienciala, L. Ciencialová, and E. Csuhaj-Varjú. "Towards on P colonies processing strings". In: *Proc. BWMC 2014, Sevilla, 2014*. Sevilla, Spain: Fénix Editora, 2014, pp. 102–118.

[5] Alexander Meduna and Petr Zemek. "Jumping Finite Automata.". In: *Int. J. Found. Comput. Sci.* 23.7 (2012), pp. 1555–1578.

APCol systems

Generating mode

The string w_F is generated by Π iff

- there exists computation starting in an initial configuration $(\varepsilon; \omega_1, \ldots, \omega_n)$ and
- the computation ends by halting in the configuration $(w_F; w_1, \ldots, w_n)$,
- where at least one agent is in its final state.

APCol systems

The results about generative power of APCol systems[6]:

- Restricted APCol systems with only two agents working in generating mode can accept any recursively set of natural numbers.
- A family of sets of natural numbers acceptable by partially blind register machine can be generated by an APCol system with one agent with restricted programs.

[6] Luděk Cienciala, Lucie Ciencialová, and Erzsébet Csuhaj-Varjú. "A class of restricted P colonies with string environment". In: *Natural Computing* 15.4 (2016), pp. 541–549. ISSN: 1572-9796. DOI: 10.1007/s11047-016-9564-3. URL: http://dx.doi.org/10.1007/s11047-016-9564-3.

Verifying strings

Verifying strings

Verifying mode

The string w_E is verified by Π iff

- there exists computation starting in an initial configuration $(w_E; \omega_1, \ldots, \omega_n)$ and
- the computation ends by halting in the configuration $(w_F; w_1, \ldots, w_n)$,
- for every i, $1 \leq i \leq m$ - supposed that the length of the input string is m - each agent rewrites some symbol at position i in some of the time variant of the environmental string occurring in the computation process.

Verifying strings

Some notes

- The length of the environmental string stay constant during computation
- Agents can use deletion programs - if one agent erases object from the string some other agent should insert new object in the place of erased object.
- An agent can insert an object into the string only if another agent erase object in demanded place.

Verifying strings

Some notes

- The length of the environmental string stay constant during computation
- Agents can use deletion programs - if one agent erases object from the string some other agent should insert new object in the place of erased object.
- An agent can insert an object into the string only if another agent erase object in demanded place.

Verifying strings

Some notes

- The length of the environmental string stay constant during computation
- Agents can use deletion programs - if one agent erases object from the string some other agent should insert new object in the place of erased object.
- An agent can insert an object into the string only if another agent erase object in demanded place.

Verifying strings

Some notes

- The length of the environmental string stay constant during computation
- Agents can use deletion programs - if one agent erases object from the string some other agent should insert new object in the place of erased object.
- An agent can insert an object into the string only if another agent erase object in demanded place.

ae $\langle a \leftrightarrow b;\ e \leftrightarrow c \rangle$

xx $\langle x \rightarrow y;\ x \leftrightarrow e \rangle$

Verifying strings

Some notes

- The length of the environmental string stay constant during computation
- Agents can use deletion programs - if one agent erases object from the string some other agent should insert new object in the place of erased object.
- An agent can insert an object into the string only if another agent erase object in demanded place.

ae $\langle a \leftrightarrow b;\ e \leftrightarrow c \rangle$

xx $\langle x \leftrightarrow e;\ x \leftrightarrow d \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

A_2

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \to a'; b \to b' \rangle$
$\langle a' \to \$'; b' \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \to \$'', \$ \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$'', \$' \to \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

$\langle e \to \$; e \to \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \to \$''; \bar{\$} \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$''; \$' \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

A_2

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \to a'; b \to b' \rangle$
$\langle a' \to \$'; b' \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \to \$'', \$ \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$'', \$' \to \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

$\langle e \to \$; e \to \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \to \$''; \bar{\$} \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$''; \$' \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

A_2

$\$''\$''$

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

$a\$$

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \to a'; b \to b' \rangle$
$\langle a' \to \$'; b' \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \to \$'', \$ \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$'', \$' \to \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

A_2

$\$b$

$\langle e \to \$; e \to \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \to \$''; \bar{\$} \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$''; \$' \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \rightarrow a'; b \rightarrow b' \rangle$
$\langle a' \rightarrow \$'; b' \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \rightarrow \$'', \$ \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$'', \$' \rightarrow \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

$\langle e \rightarrow \$; e \rightarrow \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \rightarrow \$''; \bar{\$} \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \rightarrow \$''; \$' \rightarrow \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \rightarrow \$'; b \rightarrow \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \rightarrow T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \rightarrow T \rangle$

Verifying strings

Example

Let $\Pi = (O, e, A_1, A_2)$ be an APCol system where the object alphabet is $O = \{a, a', b, b', \$, \bar{\$}, \$', \$'', T\}$, and

A_1

A_2

$\langle \bar{\$} \leftrightarrow a; \bar{\$} \leftrightarrow b \rangle$
$\langle a \to a'; b \to b' \rangle$
$\langle a' \to \$'; b' \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$ \rangle$
$\langle a \to \$'', \$ \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$''; \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$'', \$' \to \$'' \rangle$
$\langle \$' \leftrightarrow \$', \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

$\langle e \to \$; e \to \$ \rangle$
$\langle \$ \leftrightarrow \bar{\$}; \$ \leftrightarrow \bar{\$} \rangle$
$\langle \bar{\$} \to \$''; \bar{\$} \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$; \$'' \leftrightarrow b \rangle$
$\langle \$ \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow a; \$' \leftrightarrow \$' \rangle$
$\langle a \to \$''; \$' \to \$'' \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \leftrightarrow b \rangle$
$\langle \$'' \to \$'; b \to \$' \rangle$
$\langle \$' \leftrightarrow \$'; \$' \to T \rangle$
$\langle \$'' \leftrightarrow \$'', \$'' \to T \rangle$

This system is able to verify strings of the form $a^n b^n$, $n \geq 1$.

Simulation of $1NFA(k)$

One-way multihead finite automaton - $1NFA(k)$[7]

A non-deterministic one-way k-head finite automaton is a construct $M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$, where

- Q is the finite set of states,
- Σ is the set of input symbols,
- $k \geq 1$ is the number of heads,
- $\triangleright \notin \Sigma$ and $\triangleleft \notin \Sigma$ are the left and the right endmarkers, respectively,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states,

[7] Markus Holzer, Martin Kutrib, and Andreas Malcher. "Complexity of multi-head finite automata: Origins and directions". In: *Theoretical Computer Science* 412.1 (2011). Complexity of Simple Programs, pp. 83–96. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.08.024.

One-way multihead finite automaton - $1NFA(k)$[7]

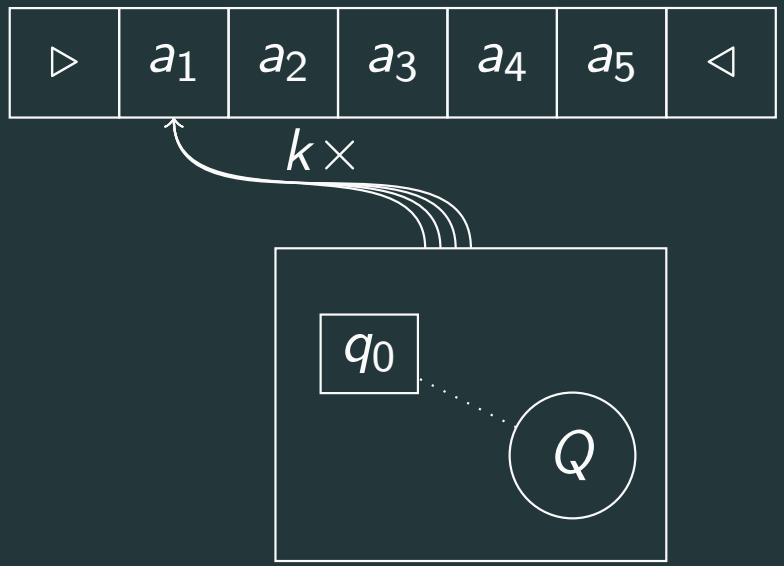
A non-deterministic one-way k-head finite automaton is a construct $M = (Q, \Sigma, k, \delta, \triangleright, \triangleleft, q_0, F)$, where

- δ is the partial transition function which maps $Q \times (\Sigma \cup \{\triangleright, \triangleleft\})^k$ into subsets of $Q \times \{0, 1\}^k$, where 1 means that the head moves one tape cell to the right and 0 means that it remains at the same position.

[7] Markus Holzer, Martin Kutrib, and Andreas Malcher. "Complexity of multi-head finite automata: Origins and directions". In: *Theoretical Computer Science* 412.1 (2011). Complexity of Simple Programs, pp. 83–96. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.08.024.

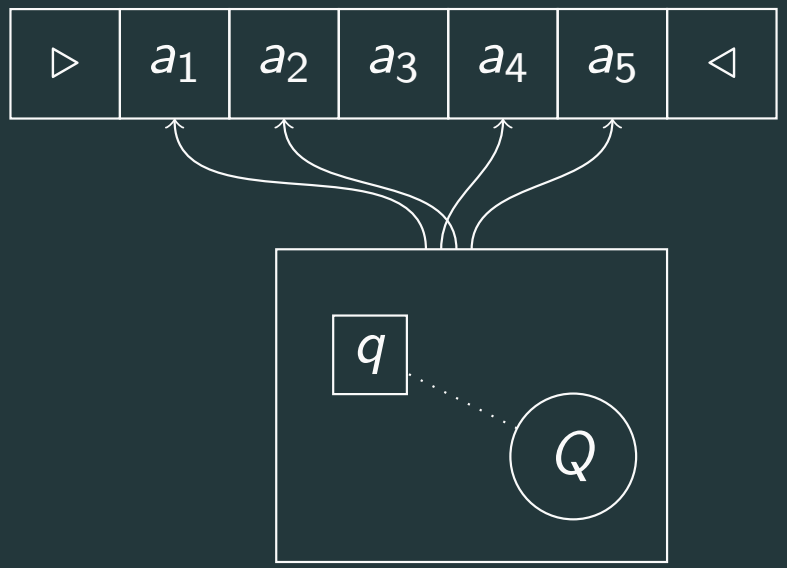
Computing with $1NFA(k)$

an initial configuration



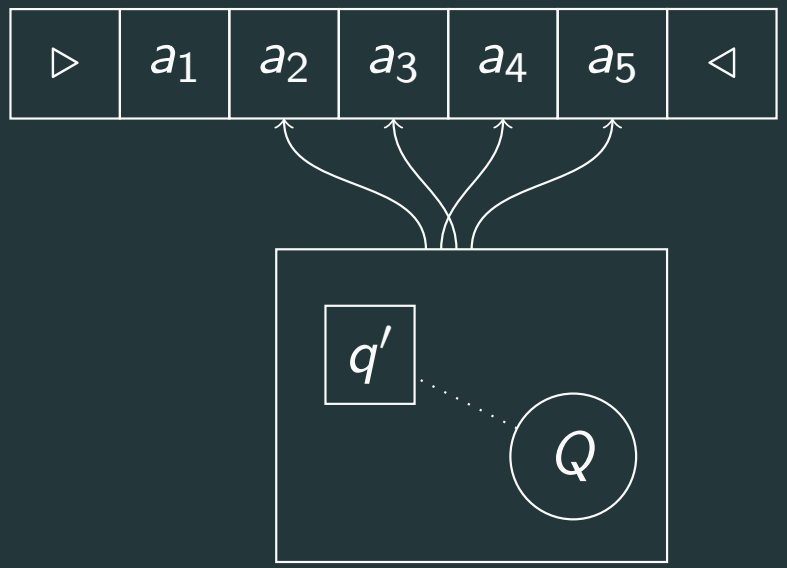
Computing with $1NFA(k)$

$$\delta(q, a_1, a_4, a_2, a_5) = (q', 1, 0, 1, 0)$$



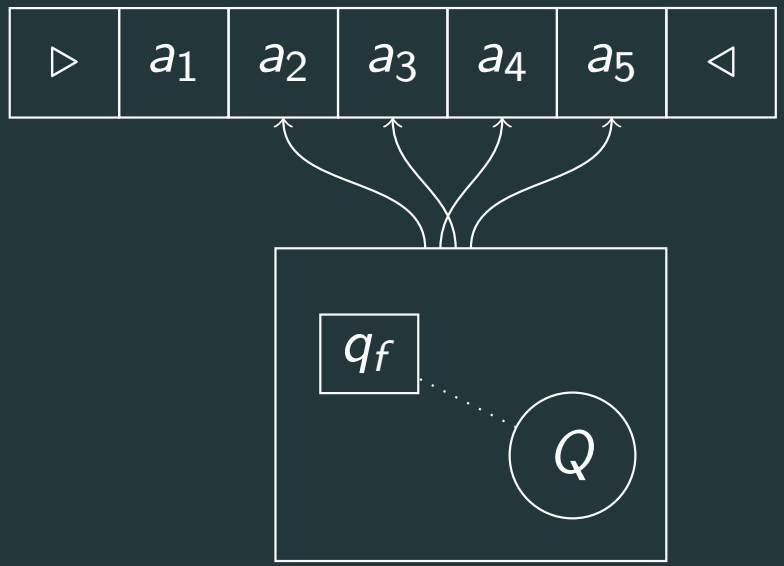
Computing with $1NFA(k)$

$$\delta(q, a_1, a_4, a_2, a_5) = (q', 1, 0, 1, 0)$$



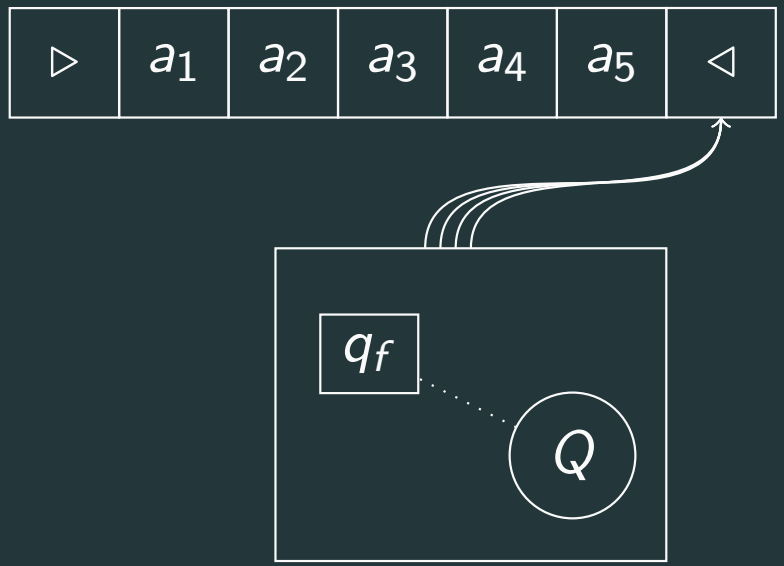
Computing with $1NFA(k)$

the end of a computation



Computing with $1NFA(k)$

the end of a computation



Simulation of $1NFA(k)$

Theorem

Let $M = (Q, \Sigma, n, \delta, \triangleright, \triangleleft, q_0, F)$, $n \geq 1$, be a one-way non-deterministic n-head finite automaton. Then we can construct an APCol system Π with $n + 2$ agents such that any word w that can be accepted by M can be verified by Π.

We construct an APCol system
$\Pi = (O, e, A_{ini,1}, A_{ini,2}, A_1, \ldots, A_n)$.

If a was scanned by reading head 1, then this fact will be indicated by having symbol $a^{(1)}$ instead of a.

The verifying process in Π corresponds to
an accepting process in M.

Simulation of $1NFA(k)$

Initialization

Two agents will initialize computation by rewriting

$$\triangleright a_1 a_2 \dots a_m \triangleleft \qquad \Longrightarrow \qquad q_{0,t,1} a_1^{(12\dots n)} a_2^{()} \dots a_m^{()} \triangleleft^{()}$$

$q_0 \in Q;\ \ t : (p, (d_1, \dots, d_n)) \in \delta(q_0, (b_1, \dots, b_n))$

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $= a_1 a_2 a_3 a_4 a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

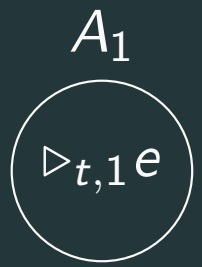
$q_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $= a_1 a_2 a_3 a_4 a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$q_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

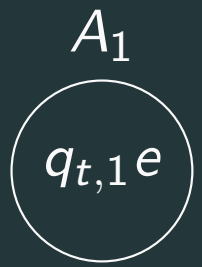


Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $= a_1 a_2 a_3 a_4 a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

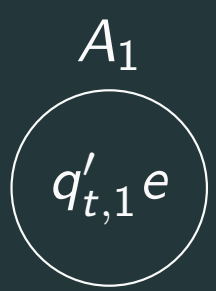


Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

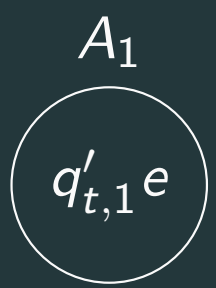
$\triangleright_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$



Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $= a_1 a_2 a_3 a_4 a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$



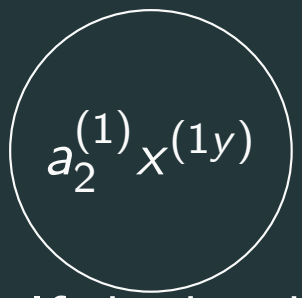
Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$q'_{t,1}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

A_1



If the head 1 have to move according to t

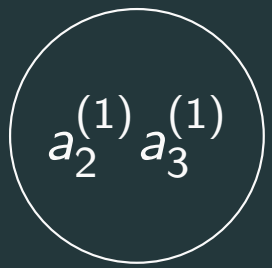
Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$q'_{t,1}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

A_1



If the head 1 have to move according to t

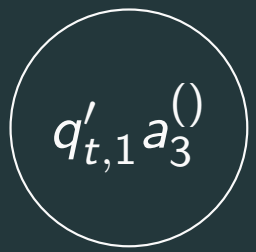
Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{(1)}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

A_1



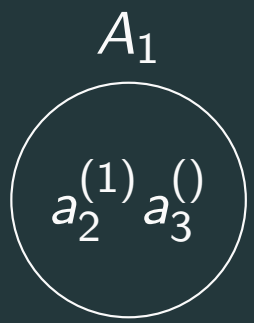
If the head 1 have to move according to t

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$q'_{t,1}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$



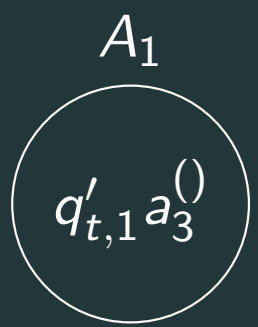
If the head 1 have to stay on the same position according to t

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$



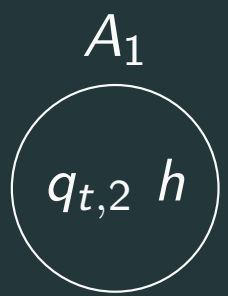
If the head 1 have to stay on the same position according to t

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $= a_1 a_2 a_3 a_4 a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,1}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

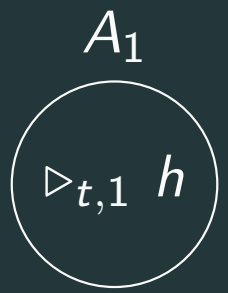


Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$q_{t,2}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$



Agent rewrites $\triangleright_{t,1}$ h to ee

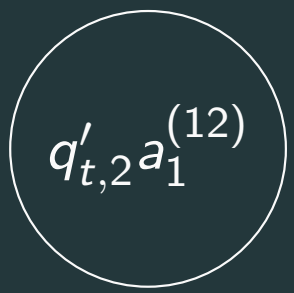
Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,2}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$

A_2



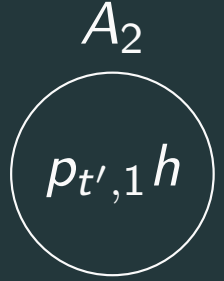
Simulation of the second head move is done

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $= a_1 a_2 a_3 a_4 a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,2}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$



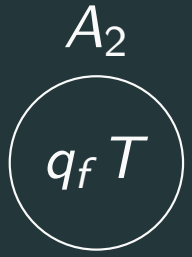
$t' : (p', (d_1, \ldots, d_n)) \in \delta(p, (b_1, \ldots, b_n))$

Simulation of $1NFA(k)$

For example: Let $1NFA$ M

- has 2 heads
- works on input string $=a_1a_2a_3a_4a_5$
- is in the configuration $(q, 2, 1)$
- $t : (p, (d_1, \ldots, d_n)) \in \delta(q, (b_1, \ldots, b_n))$

$\triangleright_{t,2}$	$a_1^{(12)}$	$a_2^{(1)}$	$a_3^{()}$	$a_4^{()}$	$a_5^{()}$	$\triangleleft^{()}$



$t' : (p', (d_1, \ldots, d_n)) \in \delta(p, (b_1, \ldots, b_n))$

Simulation of $1NFA(k)$

The input string is verified by APCol system only if the computation halts and agents visited each position in the string.

The input string is accepted by $1NFA(k)$ only if the automaton is in the final state and because we set the condition that final state can be reached only if all heads are "parked" in most right symbol of the string, all heads must go through each position in the string (excluding the 0. position).

- To every n-head one-way finite automaton there exists an APCol system Π with $n+2$ agents such that any word w that can be accepted by M can be verified by Π.

Conclusion

- We introduce new computational mode - verifying mode
- We proved that: To every n-head one-way finite automaton there exists an APCoI system Π with $n + 2$ agents such that any word w that can be accepted by M can be verified by Π.

Thanks!

I would like to thank[8]:

- organizers the conference for their amazing work,
- my colleagues for their ideas and patience,
- You, the audience, for your attention.

[8]This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602, by SGS/13/2016 and by Grant No. 120558 of the National Research, Development, and Innovation Office, Hungary.