Generating APCol systems, mainly deterministic ones

Lucie Ciencialová[1] Luděk Cienciala[1] Erzsébet Csuhaj-Varjú[2]

17th Brainstorming Week on Membrane Computing

February 5-8, 2019, Sevilla, Spain

[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 ludek.cienciala@fpf.slu.cz

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

Outline

APCol systems (Automaton-like P colonies)

were introduced in[1] as an extension of P colonies (introduced in[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 - 2.

Programs

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

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.

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

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

Result of computation - 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 of $w_i \in F_i$ for $1 \leq i \leq n$.

Result of computation - accepting mode

In the case of accepting mode, a computation is called accepting if and only if at least one agent is in final state and the string obtained is ε. The string ω is accepted by the APCol system Π if there exists a computation by Π such that it starts in the initial configuration $(\omega; \omega_1, \ldots, \omega_n)$ and the computation ends by halting in the configuration $(\varepsilon; w_1, \ldots, w_n)$, where at least one of $w_i \in F_i$ for $1 \leq i \leq n$.

Determinism

Configurations and multisets of programs

Let $c = (w_1, \ldots, w_n; w_E)$ be a configuration of APCol system. If the system is deterministic then there is only one maximal multiset of applicable programs M_P- at least one for each agent.

We can construct n-tuple x_c of strings of length 2 $a_i b_i$ corresponding to string that agent i consumes from environmental string by applying program from M_P. If there is rewriting rule in the program e appears in the string $a_i b_i$. If some agent has no applicable program there is ee in the x_c.

$$u_0 \ a_{i_1} b_{i_1} \ u_1 \ a_{i_2} b_{i_2} \ u_2 \ldots u_{n-1} \ a_{i_n} b_{i_n} \ u_n = w_E$$

Deterministic APCol system in accepting mode

Let M be two-way k-headed deterministic finite automaton ($2DFA(k)$) then there exists deterministic APCol system A working in accepting mode such that L(M)=L(A).

Deterministic APCol system in generating mode

Let M be deteministic register machine then there exists deterministic APCol system A working in generating mode and with two agents such that N(M)=N(A).

Idea - how to do zero-check ($l_1 : (SUB(r), l_2, l_3)$)

Content of register r is represented by number of a_r in the environmental string. The string is in the form

$$\#a_1 \dots a_1 \; a_2 \dots a_2 \; a_3 \dots a_n \#'$$

n is number of registers.

If agent need to erase some a_r it place mark $\uparrow$ just after $\#$ and move it through the string. If there is any a_r agent erase it and generate label l_2. If there is no a_r, agent consumes $\uparrow$ together with a_s $(s > r)$ or $\#'$ it generates label l_3.

APCol systems

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

- Restricted APCol systems with two agents working in generating mode can accept any recursively set of natural numbers.

$$NAPCol_{gen}R(2) = NRE$$

- 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.

$$NRM_{pb} \subseteq NAPCol_{gen}R(1)$$

[4] 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.

What can I do? I am alone. . . and nondeterministic

$CS \subseteq APCol_{gen}(1)$

To every context-sensitive grammar G in Kuroda normal form there exists APCol system A with one agent working in generating mode such that L(G)=L(A).

$APCol_{gen}(1) \subseteq CS$

To every APCol system A with one agent working in generating mode there exists context-sensitive grammar G such that L(A)=L(G).

CS grammar $\rightarrow$ APCol system

Initialization

Agent	Program	String
(ee)	$\langle e \rightarrow S; e \rightarrow X' \rangle$	ε
(SX')	$\langle S \leftrightarrow e; X' \rightarrow X \rangle$	ε
(eX)		S

CS grammar $\rightarrow$ APCol system

$p_i : A \rightarrow BC$

Agent	Program	String
(eX)	$\langle e \rightarrow p_i; X \rightarrow X' \rangle$	$u\ A\ v$
(p_iX')	$\langle p_i \rightarrow p'_i; X' \leftrightarrow A \rangle$	$u\ A\ v$
(p'_iA)	$\langle p'_i \rightarrow p''_i; A \rightarrow B \rangle$	$u\ X'\ v$
(p''_iB)	$\langle p''_i \leftrightarrow X'; B \leftrightarrow e \rangle$	$u\ X'\ v$
$(X'e)$	$\langle X' \leftrightarrow p''_i; e \rightarrow e \rangle$	$u\ Bp''_i\ v$
(p''_ie)	$\langle p''_i \rightarrow p'''_i; e \rightarrow C \rangle$	$u\ BX'\ v$
(p'''_iC)	$\langle p'''_i \rightarrow p'''_i; C \leftrightarrow X' \rangle$	$u\ BX'\ v$
(p'''_iX')	$\langle p'''_i \rightarrow e; X' \rightarrow X \rangle$	$u\ BC\ v$

APCol system $\rightarrow$ CS grammar

Idea

String	Rules	Program
$\boxed{ab}\, a_1 a_2 \ldots a_n$		
	$\boxed{ab}\, a_i \rightarrow a_i \boxed{ab}$	
	$a_i \boxed{ab} \rightarrow \boxed{ab}\, a_i$	
	$\boxed{ab}\, c \rightarrow a \boxed{cd}'$	$< a \leftrightarrow c; b \leftrightarrow d >$
	$\boxed{cd}'\, d \rightarrow b \boxed{cd}$	

Thanks!

I would like to thank:

- my colleagues for their ideas, work and patience,
- You, the audience, for your attention,
- and, finally, Sevilla team for organizing this perfect event.