The Restricted APCol Systems

Erzsébet Csuhaj-Varjú, Luděk Cienciala, Lucie Ciencialová

Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
csuhaj@inf.elte.hu
and
Institute of Computer Science
and
Research Institute of the IT4Innovations Centre of Excellence,
Silesian University in Opava, Czech Republic
{ludek.cienciala, lucie.ciencialova}@fpf.slu.cz

February 4, 2015

Outline

Ideas

- P Colonies + Automata $\Rightarrow$ APCol systems
- *One-membrane agents are "floating" in the environment around a string to be processed*
- *Programs - ordered rules - communication or evolution - in current version only two rules in the program*

Ideas

- P Colonies + Automata $\Rightarrow$ APCol systems
- One-membrane agents are "floating" in the environment around a string to be processed
- *Programs - ordered rules - communication or evolution - in current version only two rules in the program*

Ideas

- P Colonies + Automata $\Rightarrow$ APCol systems
- One-membrane agents are "floating" in the environment around a string to be processed
- Programs - ordered rules - communication or evolution - in current version only two rules in the program

How it works

Informally

- Agents communicate with input string, there is only sufficient amount of copies of the environmental object around the string.
- *For the case that program is formed from communication rules the order of rules in program is important.*
  - *In the case of program $\langle a \leftrightarrow b; c \leftrightarrow d \rangle$, a substring bd of the input string is replaced by string ac.*
  - *If the program is of the form $\langle c \leftrightarrow d; a \leftrightarrow b \rangle$, then a substring db of the input string is replaced by string ca.*
- *We call APCol system restricted if all the programs are formed from one evolution and one communication rule.*

How it works

Informally

- Agents communicate with input string, there is only sufficient amount of copies of the environmental object around the string.
- For the case that program is formed from communication rules the order of rules in program is important.
  - *In the case of program $\langle a \leftrightarrow b; c \leftrightarrow d \rangle$, a substring bd of the input string is replaced by string ac.*
  - *If the program is of the form $\langle c \leftrightarrow d; a \leftrightarrow b \rangle$, then a substring db of the input string is replaced by string ca.*
- *We call APCol system restricted if all the programs are formed from one evolution and one communication rule.*

How it works

Informally

- Agents communicate with input string, there is only sufficient amount of copies of the environmental object around the string.
- For the case that program is formed from communication rules the order of rules in program is important.
  - In the case of program $\langle a \leftrightarrow b; c \leftrightarrow d \rangle$, a substring bd of the input string is replaced by string ac.
  - If the program is of the form $\langle c \leftrightarrow d; a \leftrightarrow b \rangle$, then a substring db of the input string is replaced by string ca.
- *We call APCol system restricted if all the programs are formed from one evolution and one communication rule.*

How it works

Informally

- Agents communicate with input string, there is only sufficient amount of copies of the environmental object around the string.
- For the case that program is formed from communication rules the order of rules in program is important.
  - In the case of program $\langle a \leftrightarrow b; c \leftrightarrow d \rangle$, a substring bd of the input string is replaced by string ac.
  - If the program is of the form $\langle c \leftrightarrow d; a \leftrightarrow b \rangle$, then a substring db of the input string is replaced by string ca.
- We call APCol system restricted if all the programs are formed from one evolution and one communication rule.

Symbol Insertion and Deletion

Communication rule can be in the form:

- $a \leftrightarrow e$ – insert a to the string and consume e or
- $e \leftrightarrow a$ – delete a from the string – exchange it by ε

Where was (is) the object e?

Changes in Communication rules

We replace the arrow in communication rule by the structure

$$\underset{FROM}{\overset{TO}{\rightleftharpoons}}$$

where *FROM* and *TO* are S – string, E – environment.

Example

- $a \underset{E}{\overset{S}{\rightleftharpoons}} e$ – get e from the environment and put a to the string – insert a to the string.
- $e \underset{E}{\overset{S}{\rightleftharpoons}} e$ – get a from the string and put e to the environment – erase a from the string.

  Only e can be put or taken from the environment.
- $a \underset{S}{\overset{S}{\rightleftharpoons}} b$ – exchange b in the string by a.

Generating and Accepting Mode

Accepting mode

The string ω is accepted by the Automaton-like P colony 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 $1 \leq i \leq n$.

Generating mode

The situation is slightly different when the APCol system works in the generating mode. A computation is called successful if only if it is halting and at least one agent is in final state. 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 $1 \leq i \leq n$.

Accepting mode[1]

Theorem (1)

The family of languages accepted by Automaton-like P colonies with one agent properly includes the family of languages accepted by jumping finite automata.

Theorem (2)

Any recursively enumerable language can be obtained as a projection of a language accepted by an Automaton-like P colony with two agents.

---

[1] In: Cienciala, L., Ciencialová, L., Csuhaj-Varjú, E.: P Colonies Processing Strings. Fundamenta Informaticae 134(2014), IOS Press, pp 51-65. DOI 10.3233/FI-2014-1090.

The power of restricted generating APCol systems I

Theorem

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

- For register machine M with m registers we construct a APCol system $\Pi = (O, e, A_1, A_2)$ simulating the computations of register machine M.
- The APCol system Π starts computation in the initial computation with empty tape.
- It starts simulation of register machine M with instruction labelled l_0 and it proceeds simulation in according to instructions of register machine.

The power of restricted generating APCol systems II

- After M reaches halting instruction, the agent A_2 in the APCol system Π erases from the tape all the symbols except symbols 1 and then APCol systems halts.
- The length of the word placed on the tape in last configuration corresponds to the number stored in the first register of M at the end of its computation.

Restricted APCol systems with one agent I

If the APCol system is formed from one agent only there are some limitation for generated languages.

Theorem

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

The proof is similar to previous one.

Theorem

$APCol_{gen}R(1) \subseteq MAT^{\lambda}$

Let $\Pi = (O, e, A)$ be the restricted APCol system with one agent. We construct the matrix grammar $G = (N, T, S, M)$ simulating Π.

Restricted APCol systems with one agent II

- The symbol on the tape $a \neq e$ is represented by $\boxed{a} \in N$ and at the end of simulation it can be rewritten to $a \in T$.
- The contain of the agent is represented by one non-terminal symbol $\boxed{AB} \in N$, where $a, b \in O$ are object placed inside the agent.
- The first applied matrix is $\left(S \to C\boxed{ee}\right)$.
- For every program of the type $\langle a \to b; c \leftrightarrow d \rangle$, $c, d \neq e$ there is one matrix in M:

$$\left(C \to C, \boxed{AC} \to \boxed{BD}, \boxed{d} \to \boxed{c}\right)$$

Restricted APCol systems with one agent III

- For every program of the type $\langle a \to b; e \leftrightarrow d \rangle$, $d \neq e$ there is one matrix in M:

$$\left(C \to C, \boxed{AC} \to \boxed{BD}, \boxed{d} \to \varepsilon\right)$$

- For every program of the type $\langle a \to b; e \leftrightarrow e \rangle$ there is one matrix in M:

$$\left(C \to C, \boxed{AE} \to \boxed{BE}\right)$$

- For every program of the type $\langle a \to b; c \leftrightarrow e \rangle$, $c \neq e$ there is one matrix in M:

$$\left(C \to C, \boxed{AC} \to \boxed{BE} c\right)$$

Restricted APCol systems with one agent IV

- and the set of matrices generating $\boxed{c}$ somewhere in the string and deleting c:
  $\left(C \to C, \boxed{x} \to \boxed{x}\,\boxed{c}, c \to \varepsilon\right)$,
  $\left(C \to C, \boxed{x} \to \boxed{c}\,\boxed{x}, c \to \varepsilon\right)$, $\forall \boxed{x}$ such that $x \in T$
- When APCol system reaches the halting configuration the matrix grammar generates corresponding string. The string is formed from non-terminals only. Matrix grammar has to rewrite rammed terminal symbols to real terminals and delete non-terminal representing content of agent and non-terminal C. The halting configuration can be presented by a string $AB \cdot w$, where $|w|_a = 1$ for all $a \in T$ such that a is present in this halting configuration and AB is content of the agent such that $ab \in F$. The set of such a representations is finite.

Restricted APCol systems with one agent V

- For each representation $AB \cdot a_1 a_2 \dots a_p,\ p \leq |T|$, we add following matrices to matrix grammar:

  $(C \rightarrow [\,\boxed{AB}a_1 a_2 \dots a_p])$

  $\left([\,\boxed{AB}a_1 a_2 \dots a_q] \rightarrow [\,\boxed{AB}a_1 a_2 \dots a_q], \boxed{a_q} \rightarrow a_q\right),$

  $\left([\,\boxed{AB}a_1 a_2 \dots a_q] \rightarrow [\,\boxed{AB}a_1 a_2 \dots a_{q-1}]\right), 1 < q \leq p$

  $\left([\,\boxed{AB}a_1] \rightarrow [\,\boxed{AB}a_1], \boxed{a_1} \rightarrow a_1\right),\ \left([\,\boxed{AB}a_1] \rightarrow [\,\boxed{AB}\,]\right)$

  $\left([\,\boxed{AB}\,] \rightarrow \varepsilon, \boxed{AB} \rightarrow \varepsilon\right)$
- All non-terminal symbols are rewritten to corresponding to terminals and non-terminal corresponding to contain of agent is deleted.

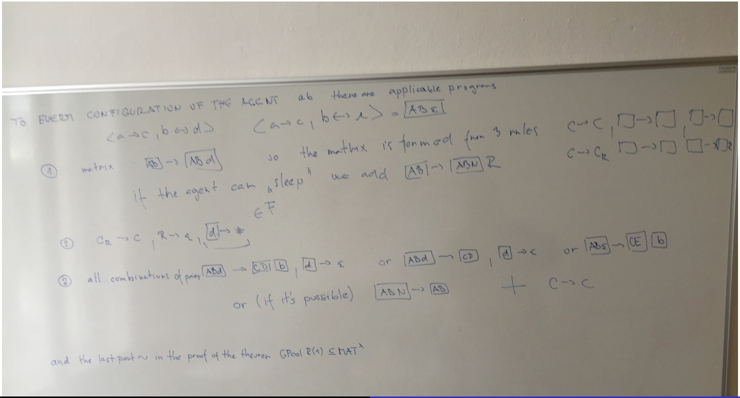
Restricted APCol systems with one agent VI

- If restricted APCol system Π generates the string ω than matrix grammar can generate it too. If APCol system halts and the agent is not in final state, the matrix grammar cannot generate string only from terminals.

Almost finished

Theorem

$APCol_{gen}R(2) \subseteq MAT$



Thank you for your attention.