About APCol Systems

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

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

[2]Department of Algorithms and Their Applications, Faculty of Informatics,
Eötvös Loránd University, Budapest, Hungary
csuhaj@inf.elte.hu

BWMC 15, January 31 - February 4, 2017, Sevilla

Outline

- **APCol Systems** introduced (in [1])
  - Agents - objects embedded in one membrane;
  - Agents acts on symbols placed in a string (tape),
  - Rules are evolution and communication,
  - e is environmental object, it plays special role inside the rules,
  - With every agent a set of final states can be associated,
  - APCol systems have always capacity 2,
  - APCol system can work in accepting or generating mode.

[1] Cienciala, L., Ciencialová, L., Csuhaj-Varjú, E.: P Colonies Processing Strings. Fundamenta Informaticae 134(2014),pp 51-65.

Communication rules

- when applying program with communication rule the agent consume one symbol from string and replace it by one of its symbols (it is given by rule)
  - $a \longleftrightarrow b$ - b is consumed and a is placed into the string instead of b

	Agent	String
Before:	ax	ubv
After:	bx	uav

  - $a \longleftrightarrow e$ - a is placed into the string in random position

	Agent	String
Before:	ax	uv
After:	ex	uav

  - $e \longleftrightarrow b$ - b is consumed and deleted from the string

	Agent	String
Before:	ex	ubv
After:	bx	uv

Communication rules

- when applying program with communication rule the agent consume one symbol from string and replace it by one of its symbols (it is given by rule)
  - $a \longleftrightarrow b$ - b is consumed and a is placed into the string instead of b

	Agent	String
Before:	ax	ubv
After:	bx	uav

  - $a \longleftrightarrow e$ - a is placed into the string in random position

	Agent	String
Before:	ax	uv
After:	ex	uav

  - $e \longleftrightarrow b$ - b is consumed and deleted from the string

	Agent	String
Before:	ex	ubv
After:	bx	uv

Communication rules

- when applying program with communication rule the agent consume one symbol from string and replace it by one of its symbols (it is given by rule)
  - $a \longleftrightarrow b$ - b is consumed and a is placed into the string instead of b

	Agent	String
Before:	ax	ubv
After:	bx	uav

  - $a \longleftrightarrow e$ - a is placed into the string in random position

	Agent	String
Before:	ax	uv
After:	ex	uav

  - $e \longleftrightarrow b$ - b is consumed and deleted from the string

	Agent	String
Before:	ex	ubv
After:	bx	uv

Communication rules

- when applying program with communication rule the agent consume one symbol from string and replace it by one of its symbols (it is given by rule)
  - $a \longleftrightarrow b$ - b is consumed and a is placed into the string instead of b

	Agent	String
Before:	ax	ubv
After:	bx	uav

  - $a \longleftrightarrow e$ - a is placed into the string in random position

	Agent	String
Before:	ax	uv
After:	ex	uav

  - $e \longleftrightarrow b$ - b is consumed and deleted from the string

	Agent	String
Before:	ex	ubv
After:	bx	uv

Communication rules

- In the case of two communication rules in one program - the order of rules plays role:
  - $a \longleftrightarrow b; c \longleftrightarrow d$ - bd is consumed and ac is placed into the string instead of bd

	Agent	String
Before:	ac	$ubdv$
After:	bd	$uacv$

  - $c \longleftrightarrow d; a \longleftrightarrow b$ - db is consumed and ca is placed into the string instead of db

	Agent	String
Before:	ac	$udbv$
After:	bd	$ucav$

Communication rules

- In the case of two communication rules in one program - the order of rules plays role:
  - $a \longleftrightarrow b; c \longleftrightarrow d$ - bd is consumed and ac is placed into the string instead of bd

	Agent	String
Before:	ac	$ubdv$
After:	bd	$uacv$

  - $c \longleftrightarrow d; a \longleftrightarrow b$ - db is consumed and ca is placed into the string instead of db

	Agent	String
Before:	ac	$udbv$
After:	bd	$ucav$

Communication rules

- In the case of two communication rules in one program - the order of rules plays role:
  - $a \longleftrightarrow b; c \longleftrightarrow d$ - bd is consumed and ac is placed into the string instead of bd

	Agent	String
Before:	ac	$ubdv$
After:	bd	$uacv$

  - $c \longleftrightarrow d; a \longleftrightarrow b$ - db is consumed and ca is placed into the string instead of db

	Agent	String
Before:	ac	$udbv$
After:	bd	$ucav$

The power of APCol Systems (in [2])

Accepting mode

Let Σ be an alphabet and let $L \subseteq \Sigma^*$ be a recursively enumerable language. Let $L' = S \cdot L \cdot E$, where $S, E \notin \Sigma$. Then there exists an APCol system Π with two agents such that $L' = L(\Pi)$ holds.

Generative mode

- $NAPCol_{gen}R(2) = NRE$.
- $NRM_{pb} \subseteq NAPCol_{gen}R(1)$.
- $APCol_{gen}R(1) \subseteq MAT^{\varepsilon}$.

[2] Cienciala, L., Ciencialová, L., Csuhaj-Varjú, E.: P Colonies Processing Strings. Fundamenta Informaticae 134(2014),pp 51-65.

APCol systems with teams

- Agents of the APCol systems are grouped into teams
- during a computation only one team of agents can work
- all agents from this team must be active
- APCol systems can work in endless-accepting mode - colours (red, green) are assigned to teams - the same condition for accepting as in Red-Green CM.

Red-Green Counter Machine

- Red–green counter machine has its set of instructions divided into two subsets - Red and Green, and M operates without halting (no HALT instruction).
- SUB and ADD instructions, READ instruction $l_1 : (READ(a), l_2)$
-
- We call an infinite run of the counter machine on input w recognizing if and only if
  - no red instruction is performed infinitely often and
  - some green instructions (one or more) are performed infinitely often.

Results

APCol systems with R-G teams can simulate any R-G Counter machine.

- Let L be language accepted by M; then language $\#L$ can be accepted by Π.
- Idea: To every instruction there are two teams of agents with the same colour as instruction has.
- $\#$ the sign of beginning of string.
- at the first phase of computation one team generate after symbols for registers - $\#_1\#_2 \ldots \#_n\$w$

Results

APCol systems with R-G teams can simulate any R-G Counter machine.

- Let L be language accepted by M; then language $\#L$ can be accepted by Π.
- Idea: To every instruction there are two teams of agents with the same colour as instruction has.
- $\#$ the sign of beginning of string.
- at the first phase of computation one team generate after symbols for registers - $\#_1\#_2\ldots\#_n\$w$

Results

APCol systems with R-G teams can simulate any R-G Counter machine.

- Let L be language accepted by M; then language $\#L$ can be accepted by Π.
- Idea: To every instruction there are two teams of agents with the same colour as instruction has.
- $\#$ the sign of beginning of string.
- at the first phase of computation one team generate after symbols for registers - $\#_1\#_2\ldots\#_n\$w$

Results

APCol systems with R-G teams can simulate any R-G Counter machine.

- Let L be language accepted by M; then language $\#L$ can be accepted by Π.
- Idea: To every instruction there are two teams of agents with the same colour as instruction has.
- $\#$ the sign of beginning of string.
- at the first phase of computation one team generate after symbols for registers - $\#_1\#_2\ldots\#_n\$w$

Results

APCol systems with R-G teams can simulate any R-G Counter machine.

- Let L be language accepted by M; then language $\#L$ can be accepted by Π.
- Idea: To every instruction there are two teams of agents with the same colour as instruction has.
- $\#$ the sign of beginning of string.
- at the first phase of computation one team generate after symbols for registers - $\#_1\#_2 \ldots \#_n\$w$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems with R-G teams can simulate any R-G Counter machine.

One-Agent Teams

- One instruction is performed by two or three teams
- the number of teams (agents) is $|H|+5$

Teams with Two Agents

- The number of agents in the team is maximally 2
- One instruction is performed by one team only
- The system has $|H|+3$ teams and the number of agents is $2|H|+3$

APCol systems and P-M Colonies

P-M colony $\Pi = (E, \#, N, >, R_1, \ldots, R_n)$

- E - alphabet, $\#$ - boundary symbol, N - set of names of agents, R_i - set of rules associated to name of agent, $>$ - priorities between agents;
- the rules are Deletion, Insertion, Substitution, Move and Death;
- Agent acts according to left and right context.
- when two agents are close to each other only one can act, it is given by priority, if there is no priority set between these two agents, they are in conflict and computation aborts.

APCol systems and P-M Colonies

P-M colony $\Pi = (E, \#, N, >, R_1, \ldots, R_n)$

- E - alphabet, $\#$ - boundary symbol, N - set of names of agents, R_i - set of rules associated to name of agent, $>$ - priorities between agents;
- the rules are Deletion, Insertion, Substitution, Move and Death;
- Agent acts according to left and right context.
- when two agents are close to each other only one can act, it is given by priority, if there is no priority set between these two agents, they are in conflict and computation aborts.

APCol systems and P-M Colonies

P-M colony $\Pi = (E, \#, N, >, R_1, \ldots, R_n)$

- E - alphabet, $\#$ - boundary symbol, N - set of names of agents, R_i - set of rules associated to name of agent, $>$ - priorities between agents;
- the rules are Deletion, Insertion, Substitution, Move and Death;
- Agent acts according to left and right context.
- when two agents are close to each other only one can act, it is given by priority, if there is no priority set between these two agents, they are in conflict and computation aborts.

APCol systems and P-M Colonies

P-M colony $\Pi = (E, \#, N, >, R_1, \ldots, R_n)$

- E - alphabet, $\#$ - boundary symbol, N - set of names of agents, R_i - set of rules associated to name of agent, $>$ - priorities between agents;
- the rules are Deletion, Insertion, Substitution, Move and Death;
- Agent acts according to left and right context.
- when two agents are close to each other only one can act, it is given by priority, if there is no priority set between these two agents, they are in conflict and computation aborts.

Theorem

To every P-M colony there exists APCol system simulating behaviour of the P-M colony.

- The idea is that there will be token going through the string;
- The first phase - Agents rewrite all agents names in the string to triples;
  $\ldots aAb \ldots \Rightarrow \ldots a\ [aAb]b \ldots$
- The second phase - conflict check - agent will be active or non-active or computation of P-M colony is aborted (APCol systems go to endless cycle)
- The third phase - execution of the rules, rewriting the triples to agents names again.

Theorem

To every P-M colony there exists APCol system simulating behaviour of the P-M colony.

- The idea is that there will be token going through the string;
- The first phase - Agents rewrite all agents names in the string to triples;
  $\ldots aAb \ldots \Rightarrow \ldots a\ [aAb]b \ldots$
- The second phase - conflict check - agent will be active or non-active or computation of P-M colony is aborted (APCol systems go to endless cycle)
- The third phase - execution of the rules, rewriting the triples to agents names again.

Theorem

To every P-M colony there exists APCol system simulating behaviour of the P-M colony.

- The idea is that there will be token going through the string;
- The first phase - Agents rewrite all agents names in the string to triples;
  $\ldots aAb \ldots \Rightarrow \ldots a\ [aAb]b \ldots$
- The second phase - conflict check - agent will be active or non-active or computation of P-M colony is aborted (APCol systems go to endless cycle)
- The third phase - execution of the rules, rewriting the triples to agents names again.

Theorem

To every P-M colony there exists APCol system simulating behaviour of the P-M colony.

- The idea is that there will be token going through the string;
- The first phase - Agents rewrite all agents names in the string to triples;
  $\ldots aAb \ldots \Rightarrow \ldots a\ [aAb]b \ldots$
- The second phase - conflict check - agent will be active or non-active or computation of P-M colony is aborted (APCol systems go to endless cycle)
- The third phase - execution of the rules, rewriting the triples to agents names again.

Theorem

To every P-M colony there exists APCol system simulating behaviour of the P-M colony.

- The idea is that there will be token going through the string;
- The first phase - Agents rewrite all agents names in the string to triples;
  $\ldots aAb\ldots \Rightarrow \ldots a\ [aAb]b\ldots$
- The second phase - conflict check - agent will be active or non-active or computation of P-M colony is aborted (APCol systems go to endless cycle)
- The third phase - execution of the rules, rewriting the triples to agents names again.

Ideas

- What about reachability?
- Can we define languages Garden-of-Eden, Life, Doomsday and Non-life?

Agent creation and dissolution

- agent (after it reaches special state) can create its copy (with evolved content)
- agent can die due to it contents
- for 2D P colonies - good tool for simulation of spread of living organisms
- Will it increase computational power?

Thanks for your attention.