P COLONIES WITH DYNAMIC ENVIRONMENT

Petr Sosík Luděk Cienciala Lucie Ciencialová

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

BWMC 14, February 1-5, 2016, Sevilla

Outline

P colonies

- One-membrane agents acting in a shared environment;
- The environment is changing only by intervention of the agents;
- The computation is maximally parallel (or sequential);
- The result is assigned only to halting computation and it is the number of copies of special object placed in the environment in the moment of halting;

Rules

- Two sets of programs (rules):
- The first set : rewriting and communication rules;
  $a \longrightarrow b$; $c \longleftrightarrow d$;
- The second set : programs for agent sender and consumer.

P colonies

- One-membrane agents acting in a shared environment;
- The environment is changing only by intervention of the agents;
- The computation is maximally parallel (or sequential);
- The result is assigned only to halting computation and it is the number of copies of special object placed in the environment in the moment of halting;

Rules

- Two sets of programs (rules):
- The first set : rewriting and communication rules;
  $a \rightarrow b$; $c \longleftrightarrow d$;
- The second set : programs for agent sender and consumer.

P colonies

- One-membrane agents acting in a shared environment;
- The environment is changing only by intervention of the agents;
- The computation is maximally parallel (or sequential);
- The result is assigned only to halting computation and it is the number of copies of special object placed in the environment in the moment of halting;

Rules

- Two sets of programs (rules):
- The first set : rewriting and communication rules;
  $a \rightarrow b$; $c \longleftrightarrow d$;
- The second set : programs for agent sender and consumer.

- **Eco-P Colonies** introduced in [**ecoP2**] and in [**ecoP1**]
  - A team of one membrane agents placed in dynamical environment
  - Eco-P colony has only one alphabet – a set of objects,
  - e is environmental object and f is final object,
  - A mechanism of changes of the environment is based on 0L scheme
  - Agents are working according to generating and consuming programs

- **[ecoP1]** L. Ciencialová, E. Csuhaj-varjú, A. Kelemenová and G. Vaszil, Variants of P colonies with very simple cell structure, *Int. J. of Computers, Communications & Control*, **3** (IV), 224–233, 2009.
- **[ecoP2]** L. Cienciala and L. Ciencialová, Eco-P colonies. In G. Păun, M.J. Pérez-Jiménez and A. Riscos-Núñez (Eds.), *Pre-Proceedings of the 10th Workshop on Membrane Computing*, 201–209, 2009.

- **Eco-P Colonies** introduced in [**ecoP2**] and in [**ecoP1**]
  - A team of one membrane agents placed in dynamical environment
  - Eco-P colony has only one alphabet – a set of objects,
  - e is environmental object and f is final object,
  - A mechanism of changes of the environment is based on 0L scheme
  - Agents are working according to generating and consuming programs

- **[ecoP1]** L. Ciencialová, E. Csuhaj-varjú, A. Kelemenová and G. Vaszil, Variants of P colonies with very simple cell structure, *Int. J. of Computers, Communications & Control*, **3** (IV), 224–233, 2009.
- **[ecoP2]** L. Cienciala and L. Ciencialová, Eco-P colonies. In G. Păun, M.J. Pérez-Jiménez and A. Riscos-Núñez (Eds.), *Pre-Proceedings of the 10th Workshop on Membrane Computing*, 201–209, 2009.

- **Rules**
  Let ab be a configuration of the eco-P colony
  - Sender – $\langle a \rightarrow cd; bout \rangle$ –
    object a is rewritten to two objects – c, d and object b is sent to the environment
  - Consumer – $\langle ab \rightarrow c; din \rangle$ –
    objects ab are rewritten to one object – c and object d is consumed from the environment

- **Rules**
  Let ab be a configuration of the eco-P colony
  - Sender – $\langle a \rightarrow cd; bout \rangle$ –
    object a is rewritten to two objects – c, d and object b is sent to the environment
  - Consumer – $\langle ab \rightarrow c; din \rangle$ –
    objects ab are rewritten to one object – c and object d is consumed from the environment

- **Rules**
  Let ab be a configuration of the eco-P colony
  - Sender – $\langle a \rightarrow cd; bout \rangle$ –
    object a is rewritten to two objects – c, d and object b is sent to the environment
  - Consumer – $\langle ab \rightarrow c; din \rangle$ –
    objects ab are rewritten to one object – c and object d is consumed from the environment

The generative power of eco-P colonies

- $NEPCOL_{sc,pass}(3,*) = NRE$ in [**ecoP1**],
  - one sender and two consumers can generate every set from NRE
- $NRM_{pb} \subseteq NEPCOL_{sc,pass}(2,*)$ in [**ecoP2**],
  - one sender and one consumer can generate every set from NRM_{pb}
- $NEPCOL_{c,activ,ini}(2,*) = NRE$ in [**ecoP2**].
  - two consumers in eco-P colony with "active" 0L scheme and initial content of the environment and the agents can generate every set from NRE

Theorem

$NRM_{pb} \subseteq NEPCOL_{c,active,ini}(1,*)$

- The rules of environment ensure process of the computation - adding, generation of the labels of the next instruction.
- SUB-instruction – the agent consumes a_r when the subtraction is needed;
- SUB-instruction – if the agent consumes instead of a_r another objects, it comes to loop ($\langle eF \rightarrow F; ein \rangle$) – because of non-determinism there exists computation with correct subtraction;
- The agent must work in every step of computation; when it stops, computation ends $\Rightarrow$ the environment generates special object D to be consumed by the agent in every second step (except of subtraction).

Theorem

$NRM_{pb} \subseteq NEPCOL_{c,active,ini}(1,*)$

- The rules of environment ensure process of the computation - adding, generation of the labels of the next instruction.
- SUB-instruction – the agent consumes a_r when the subtraction is needed;
- SUB-instruction – if the agent consumes instead of a_r another objects, it comes to loop ($\langle eF \rightarrow F; ein \rangle$) – because of non-determinism there exists computation with correct subtraction;
- The agent must work in every step of computation; when it stops, computation ends $\Rightarrow$ the environment generates special object D to be consumed by the agent in every second step (except of subtraction).

Theorem

$NRM_{pb} \subseteq NEPCOL_{c,active,ini}(1,*)$

- The rules of environment ensure process of the computation - adding, generation of the labels of the next instruction.
- SUB-instruction – the agent consumes a_r when the subtraction is needed;
- SUB-instruction – if the agent consumes instead of a_r another objects, it comes to loop ($\langle eF \rightarrow F; ein \rangle$) – because of non-determinism there exists computation with correct subtraction;
- The agent must work in every step of computation; when it stops, computation ends $\Rightarrow$ the environment generates special object D to be consumed by the agent in every second step (except of subtraction).

Theorem

$NRM_{pb} \subseteq NEPCOL_{c,active,ini}(1,*)$

- The rules of environment ensure process of the computation - adding, generation of the labels of the next instruction.
- SUB-instruction – the agent consumes a_r when the subtraction is needed;
- SUB-instruction – if the agent consumes instead of a_r another objects, it comes to loop ($\langle eF \rightarrow F; ein \rangle$) – because of non-determinism there exists computation with correct subtraction;
- The agent must work in every step of computation; when it stops, computation ends $\Rightarrow$ the environment generates special object D to be consumed by the agent in every second step (except of subtraction).

Theorem

$NRM_{pb} \subseteq NEPCOL_{c,active,ini}(1,*)$

- The rules of environment ensure process of the computation - adding, generation of the labels of the next instruction.
- SUB-instruction – the agent consumes a_r when the subtraction is needed;
- SUB-instruction – if the agent consumes instead of a_r another objects, it comes to loop ($\langle eF \rightarrow F; ein \rangle$) – because of non-determinism there exists computation with correct subtraction;
- The agent must work in every step of computation; when it stops, computation ends $\Rightarrow$ the environment generates special object D to be consumed by the agent in every second step (except of subtraction).

Theorem

$NUL \subseteq NEPCOL_{c,active,ini}(1,*)$

- UL – the family of unary 0L languages;
- The rules of environment are the same as in UL system, the number of copies of final object equals to the length of the axiom .
- The agent controls computation with two programs $ee \rightarrow e; ein$ and $ee \rightarrow F; ein$.

Theorem

$NUL \subseteq NEPCOL_{c,active,ini}(1,*)$

- UL – the family of unary 0L languages;
- The rules of environment are the same as in UL system, the number of copies of final object equals to the length of the axiom .
- The agent controls computation with two programs $ee \rightarrow e; ein$ and $ee \rightarrow F; ein$.

Theorem

$NUL \subseteq NEPCOL_{c,active,ini}(1,*)$

- UL – the family of unary 0L languages;
- The rules of environment are the same as in UL system, the number of copies of final object equals to the length of the axiom .
- The agent controls computation with two programs $ee \rightarrow e; ein$ and $ee \rightarrow F; ein$.

Theorem

$NUL \subseteq NEPCOL_{c,active,ini}(1,*)$

- UL – the family of unary 0L languages;
- The rules of environment are the same as in UL system, the number of copies of final object equals to the length of the axiom .
- The agent controls computation with two programs $ee \rightarrow e; ein$ and $ee \rightarrow F; ein$.

- P colonies with rewriting and communication rules;
- The environment is equipped with 0L scheme;
- We focuse on P colonies with capacity one without checking rules.
- $PCOL_{par}(1,6,*) = NRE$ in [**ecoP1**].

- P colonies with rewriting and communication rules;
- The environment is equipped with 0L scheme;
- We focuse on P colonies with capacity one without checking rules.
- $PCOL_{par}(1,6,*) = NRE$ in [**ecoP1**].

- P colonies with rewriting and communication rules;
- The environment is equipped with 0L scheme;
- We focuse on P colonies with capacity one without checking rules.
- $PCOL_{par}(1,6,*) = NRE$ in [**ecoP1**].

- P colonies with rewriting and communication rules;
- The environment is equipped with 0L scheme;
- We focuse on P colonies with capacity one without checking rules.
- $PCOL_{par}(1,6,*) = NRE$ in [**ecoP1**].

THEOREM

$NPCOL_{par,active}(1,2,*) = NRE$

- The environment generates labels of the instructions to be executed and object a_r in the case of adding;
- It also generate copies of the object D to be consumed by one of the agent to ensure that the computation will continue;
- The agents helps each other to subtract in the case of SUB-instruction.
- The computation ends after object corresponding to the halting instruction appears in the environment.

THEOREM

$NPCOL_{par,active}(1,2,*) = NRE$

- The environment generates labels of the instructions to be executed and object a_r in the case of adding;
- It also generate copies of the object D to be consumed by one of the agent to ensure that the computation will continue;
- The agents helps each other to subtract in the case of SUB-instruction.
- The computation ends after object corresponding to the halting instruction appears in the environment.

THEOREM

$NPCOL_{par,active}(1,2,*) = NRE$

- The environment generates labels of the instructions to be executed and object a_r in the case of adding;
- It also generate copies of the object D to be consumed by one of the agent to ensure that the computation will continue;
- The agents helps each other to subtract in the case of SUB-instruction.
- The computation ends after object corresponding to the halting instruction appears in the environment.

THEOREM

$NPCOL_{par,active}(1,2,*) = NRE$

- The environment generates labels of the instructions to be executed and object a_r in the case of adding;
- It also generate copies of the object D to be consumed by one of the agent to ensure that the computation will continue;
- The agents helps each other to subtract in the case of SUB-instruction.
- The computation ends after object corresponding to the halting instruction appears in the environment.

THEOREM

$NPCOL_{par,active}(1,2,*) = NRE$

- The environment generates labels of the instructions to be executed and object a_r in the case of adding;
- It also generate copies of the object D to be consumed by one of the agent to ensure that the computation will continue;
- The agents helps each other to subtract in the case of SUB-instruction.
- The computation ends after object corresponding to the halting instruction appears in the environment.

Thanks for your attention.