P Colony Robot Controller

Miroslav Langer, Luděk Cienciala[1], Lucie Ciencialová[1], Michal Perdek[1], Vladimír Smolka[1]

Institute of Computer Science and Research Institute of the IT4Innovations Centre of Excellence, Silesian University in Opava, Czech Republic
{{miroslav.langer, ludek.cienciala, lucie.ciencialova, michal.perdek, vladimir.smolka}@fpf.slu.cz

Summary. P colonies were introduced in 2004 (see [7]) as an abstract computing device composed of independent single membrane agents, reactively acting and evolving in a shared environment. Each agent is equip with set of rules which are structured into simple programs.

We use this very simple symbol processing computational device to build complex robot controllers. Moreover, we group agents into the modules (see [1]). Each module fulfils particular function. This allows us to easily extend our controller or change its function without rebuilding whole P colony. In this paper we introduce simple controller for passing the maze using right-hand rule.

1 Introduction

One of the main tasks of the robotics is to design or program the robot controller. Robot is controlled by controller, which aims to transform outputs from the sensors into the inputs for actuators. The controller can receive input signals from various kinds of sensors, cameras or through communication channels and it has to decide, how this information affect the activity of internal states of the robot and actuators. Robot controller can use different ways to achieve this activity. The controller can be based on the rule-based systems, (fuzzy) expert systems, artificial neural networks and many other techniques.

Recently, we often meet with hybrid controllers, which combine multiple technics to achieve more suitable reaction of robotic systems, or improvement of artificial intelligence. Each of applicable techniques for control has its advantage, but also disadvantage, whether it concerns the complexity, adaptability or scalability etc. We often meet with alternative control methods, which are the result of interdisciplinary convergence of different science disciplines such as biology, genetics, cognitive science, neuroscience, psychology and more.

The concepts of using knowledge of biology can be found in e.g. [4]. Concepts taken from biology such as membrane systems or P-systems will allow us to take these advantages such as parallelization and distributivity of each parts of the controller.

Controller based on P colonies described in this paper is drawn up as a group of cooperating agents who live in shared environment, through which agents can communicate. Such controller can be used for wide range of tasks associated with control issues.
Throughout the paper we assume that the reader is familiar with the basics of the formal language theory.

2 Preliminaries on the P colonies

P colonies were introduced in 2004 (see [7]) as an abstract computing device composed of independent single membrane agents, reactively acting and evolving in a shared environment. This model is inspired by structure and function of a community of living organisms in a shared environment.

The independent organisms living in a P colony are called agents or cells. Each agent is represented by a collection of objects embedded in a membrane. The number of objects inside each agent is constant during the computation. With each agent is associated a set of simple programs. Each program is composed from the rules which can be of two types. The first type of rules, called the evolution rules, are of the form $a \rightarrow b$. It means that the object a inside the agent is rewritten (evolved) to the object b. The second type of rules, called the communication rules, are of the form $c \leftrightarrow d$. If the communication rule is performed, the object c inside the agent and the object d outside the agent swap their places. Thus after executing the rule, the object d appears inside the agent and the object c is placed outside the agent.

In [6], the set of programs was extended by the checking rules. These rules give the agents an opportunity to opt between two possibilities. The rules are in the form r_1/r_2. If the checking rule is performed, then the rule r_1 has higher priority to be executed over the rule r_2. It means that the agent checks whether the rule r_1 is applicable. If the rule can be executed, then the agent is compulsory to use it. If the rule r_1 cannot be applied, then the agent uses the rule r_2. The program determines the activity of the agent. The agent can change the content of itself or of the environment by programs and it can affect the behaviour of other agents through the environment.

The environment contains several copies of the basic environmental object denoted by e. The environmental object e appears in arbitrary large number of copies in the environment.

This interaction between agents is a key factor in functioning of the P colony. In each moment each object inside the agent is affected by executing the program.

For more information about P systems see [11] or [12].

Definition 1. *The P colony of the capacity k is a construct*
$$\Pi = (A, e, f, V_E, B_1, \ldots, B_n), \text{ where}$$

- *A is an alphabet of the colony, its elements are called objects,*
- *$e \in A$ is the basic object of the colony,*
- *$f \in A$ is the final object of the colony,*
- *V_E is a multiset over $A - \{e\}$, it determines the initial state (content) of the environment,*
- *B_i, $1 \leq i \leq n$, are agents, each agent is a construct $B_i = (O_i, P_i)$, where*
  - *O_i is a multiset over A, it determines the initial state (content) of the agent, $|O_i| = k$,*
  - *$P_i = \{p_{i,1}, \dots, p_{i,k_i}\}$ is a finite multiset of programs, where each program contains exactly k rules, which are in one of the following forms each:*
    - *$a \rightarrow b$, called the evolution rule,*
    - *$c \leftrightarrow d$, called the communication rule,*
    - *r_1/r_2, called the checking rule; r_1, r_2 are the evolution rules or the communication rules.*

An initial configuration of the P colony is an $(n+1)$-tuple of strings of objects present in the P colony at the beginning of the computation. It is given by the multiset O_i for $1 \leq i \leq n$ and by the set V_E. Formally, the configuration of the P colony Π is given by $(w_1, \dots, w_n, w_E)$, where $|w_i| = k$, $1 \leq i \leq n$, w_i represents all the objects placed inside the i-th agent, and $w_E \in (A - \{e\})^\circ$ represents all the objects in the environment different from the object e.

We will use the parallel model of P colonies for the robot controller. That means that each agent tries to find one usable program at each step of the parallel computation. If the number of applicable programs is higher than one, then the agent nondeterministically chooses one of the programs. The maximal possible number of agents is active at each step of the computation.

The configuration is called halting if the set of program labels P satisfying the mentioned conditions above is empty. A set of all possible halting configurations is denoted by H.

Each P colony is characterised by three characteristics; the capacity k, the degree n and the height h; denoted by $NPCOL_{par}K(k, n, h)$. The capacity k is the number of the objects inside each agent, the degree n is the number of agents in the P colony, the height h is the maximal number of programs associated with the agent of the P colony.

2.1 Modularity in the therms of P colonies

In our research of the P colonies we observed that some agents are performing the same function during the computation. In the [1] we grouped agents of the P colony simulating computation of the register machine into the modules. The agents providing subtraction were grouped into the subtraction module, agents providing addition were grouped into the addition module, agents controlling the computation were grouped into the control module, etc. The program of simulated register machine is stored in the control module, so changing the program of the register machine does not mean reprograming all the agents of the P colony but the change of the control module.

We use this approach to design robot controller. Each individual robot's module like infrared sensors, actuators etc. is represented by the one P colony module. All the P colony modules are driven by the controlling module, which contains controlling logic and drives robot's behaviour.

3 P colony robot controller

P colonies are very simple, but computationally complete string processing computational device working in parallel manner. Data are processed by very primitive computational units using very simple rules formed into the programs. Collaterally working autonomous units sharing common environment provides fast computation device. Dividing agents into the modules allows us to compound agents controlling single robot sensors and actuators. All the modules are controlled by the main controlling unit. Controlling unit collect information from the sensors and send the instructions to the actuators. All the communication is done via the environment.

We construct a P colony with four modules: *Controlling unit*, *Left actuator controller*, *Right actuator controller* and *Infra-red receptor*. Entire colony is amended by the *input* and *output filter*. The input filter codes signals from the robots receptors and spread the coded signal into the environment. In the environment there is the coded signal used by the agents. The output filter decodes the signal from the environment which the actuator controllers sent into it. Decoded signal is forwarded to the robots actuators.

The infra-red receptors consume all the symbols released into the environment by the input filter. It releases actual information from the sensors on demand of the control unit. The infra-red receptors remove unused data from the environment.

The actuator controllers wait for the activating signal from the control unit. After obtaining the activating signal the controllers try to provide demanded action by sending special objects - coded signal for the output filter into the environment. When the action is performed successfully the actuators send the announcement of the successful end of the action to the control unit, the announcement of the unsuccessful end of the action otherwise.

The controlling unit ensures the computation. It controls the behaviour of the robot, it asks the data from the sensors and it sends instructions to the actuators by sending particular symbols to the environment. The controlling unit contains set of programs which provides fulfilling set goal, in this case pass through the maze using the right-hand rule. If the exit from the maze is found; symbol G is appears in the environment, the controlling unit releases into the environment special symbol H, which stops the infra-red receptors and the P colony stops and so the robot.

Let us introduce the formal definition of the mentioned P colony:
$\Pi = (A, e, V_E, (O_1, P_1), \ldots (O_5, P_5), \emptyset)$, where

$A \;= \{1_L, 1_R, -1_L, -1_R, A_L, A_R, F, F_F, F_O, F^F_F, F^F_O, G, I_F, I_R, H, H_1, H_2, H_3,$
$\quad H_4, L, M_F, R, R_F, R_O, R^F_F, R^F_O, W_i\}$
$V_E = \{e\}$,

Let us describe the meaning of the particular objects:

$1_L, -1_L$	Signal for the output filter - move the left wheel forward/bacward.
$1_R, -1_R$	Signal for the output filter - move the right wheel forward/bacward.
F, L, R	Signal from the control unit to the actuator controllers - move forward, turn left/right.
F^F_F, R^F_F	Signal from the input filter - no obstacle in front/on the right.
F^F_O, R^F_O	Signal from the input filter - obstacle in front/on the right.
F_F, R_F	Signal from the IR module to the control unit - no obstacle in front/on the right.
F_O, R_O	Signal from the IR module to the control unit - obstacle in front/on the right.
I_F, I_R	Signal from the control unit to the IR module - is there an obstacle in front/on the right?
G	Maze exit.
H	Halting symbol.

Remaining objects are used for inner representation of the actions and as complementary objects.

Particular modules and agents which they contain are defined as follows:

Control Unit:

$O_1 = \{\, I_F, I_R, W_i\}$,7

$$\begin{array}{lll}
P_1 = \{ & < ee{\leftrightarrow}A_LA_R; e \leftrightarrow G/e \rightarrow e >; & < A_LA_Re \rightarrow I_FI_RW_i >; \\
& < I_FI_R{\leftrightarrow}ee; W_i \rightarrow W_i >; & < W_i{\rightarrow}W_i; ee \leftrightarrow R_FF_F >; \\
& < W_i{\rightarrow}W_i; ee \leftrightarrow R_OF_F >; & < W_i{\rightarrow}W_i; ee \leftrightarrow R_OF_O >; \\
& < W_i{\rightarrow}W_i; ee \leftrightarrow R_FF_O >; & < R_FF_Oe{\rightarrow}RRM_F >; \\
& < R_OF_Fe{\rightarrow}FFe >; & < R_FF_Fe{\rightarrow}RRM_F >; \\
& < R_OF_Oe{\rightarrow}LLe >; & < FF{\leftrightarrow}ee; e{\rightarrow}e >; \\
& < LL{\leftrightarrow}ee; e{\rightarrow}e >; & < RR{\leftrightarrow}ee; M_F{\rightarrow}M_F >; \\
& < M_Fee{\rightarrow}FFe >; & < A_LA_RG{\rightarrow}HHH >; \\
& < HH{\leftrightarrow}ee; H \rightarrow H_1 >; & < H_1ee{\rightarrow}HHH_1 >; \\
& < HH{\leftrightarrow}ee; H_1 \rightarrow H_2 >; & < H_2ee{\rightarrow}HHH_2 >; \\
& < HH{\leftrightarrow}ee; H_2 \rightarrow H_3 >; & < H_3ee{\rightarrow}HHH_3 >; \\
& < HH{\leftrightarrow}ee; H_3 \rightarrow H_4 >; & < H_4ee{\rightarrow}HHH_4 >; \\
& < HH{\leftrightarrow}ee; H_4 \rightarrow e >; &
\end{array}$$

Control Unit contains program which controls robots behaviour. According to the data obtained from the IR module it sends instructions to the Actuator controllers. It passes the maze using the right-hand rule until it finds symbol G which represents exit from the maze. While it founds the exit the Control unit releases symbol H into the environment which stops the computation.

Left Actuator controller:
$O_2 = \{\ e, e, e\}$,
$P_2 = \{ < e{\leftrightarrow}F; e \rightarrow 1_L; e \rightarrow A_L >;$
$< F{\rightarrow}e; 1_L \leftrightarrow e; A_L \leftrightarrow e >;$
$< e{\leftrightarrow}R; e \rightarrow 1_L; e \rightarrow A_L >;$
$< R{\rightarrow}e; 1_L \leftrightarrow e; A_L \leftrightarrow e >;$
$< e{\leftrightarrow}L; e \rightarrow -1_L; e \rightarrow A_L >;$
$< L{\rightarrow}e; -1_L \leftrightarrow e; A_L \leftrightarrow e >;$

Right Actuator controller:
$O_3 = \{\ e, e, e\}$,
$P_3 = \{ < e{\leftrightarrow}F; e \rightarrow 1_R; e \rightarrow A_R >;$
$< F{\rightarrow}e; 1_R \leftrightarrow e; A_R \leftrightarrow e >;$
$< e{\leftrightarrow}R; e \rightarrow -1_R; e \rightarrow A_R >;$
$< R{\rightarrow}e; -1_R \leftrightarrow e; A_R \leftrightarrow e >;$
$< e{\leftrightarrow}L; e \rightarrow 1_R; e \rightarrow A_R >;$
$< L{\rightarrow}e; 1_R \leftrightarrow e; A_R \leftrightarrow e >;$

Right and left Actuator controllers wait for the activating signal from the Control unit. According to signal they move the robot in required direction by sending appropriate signal to the output filter.

Front Infra-red module:
$O_4 = \{\ e, e, e\}$,
$P_4 = \{ < e \leftrightarrow H/e \leftrightarrow F_F^F; ee \rightarrow eF_F >;$
$< e \leftrightarrow H/e \leftrightarrow F_O^F; ee \rightarrow eF_O >;$
$< F_F \leftrightarrow I_F/F_F \leftrightarrow e; F_F^F e \rightarrow ee >;$
$< F_O \leftrightarrow I_F/F_O \leftrightarrow e; F_O^F e \rightarrow ee >;$
$< I_F ee \rightarrow eee; >\}$

Right Infra-red module:
$O_5 = \{\ e, e, e\}$,
$P_5 = \{ < e \leftrightarrow H/e \leftrightarrow R_F^F; ee \rightarrow eR_F >;$
$< e \leftrightarrow H/e \leftrightarrow R_O^F; ee \rightarrow eR_O >;$
$< R_F \leftrightarrow I_R/R_F \leftrightarrow e; R_F^F e \rightarrow ee >;$
$< R_O \leftrightarrow I_R/R_O \leftrightarrow e; R_O^F e \rightarrow ee >;$
$< I_R ee \rightarrow eee; >\}$

IR modules consume all the symbols send by the input filter into the environment. They send actual data to the Control unit on demand.

Robot driven by this P colony is able to pass through simple mazes that are possible to pass using the right-hand rule.

4 Conclusion

We have shown the basic possibilities of controlling the robot using the P colonies and modular approach. We constructed P colony with five simple modules for passing the maze using right-hand rule. With respect to the fact that P colonies are computationally complete devices will the further research be dedicated to the more precise control and fulfilling more difficult and complicated tasks like processing the information from distance sensors.

Remark 1. Article has been made in connection with project IT4Innovations Centre of Excellence, reg. no. CZ.1.05/1.1.00/02.0070.

Research was also supported by the SGS/24/2013 Project of the SU Opava.

References

1. L. Cienciala, L., Ciencialová,, Langer, M.: *Modularity in P colonies with Checking Rules.* In: Revised Selected Papers 12 th International Conference CMC 2011 (George, M., Păun, Gh., Rozenber, G., Salomaa, A., Verlan, S. eds.), Springer, LNCS 7184, 2012, pp. 104 120.
2. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, Gh., Vaszil, G.: *Cells in environment: P colonies*, Multiple-valued Logic and Soft Computing, 12, 3-4, 2006, pp. 201–215.
3. Csuhaj-Varjú, E., Margenstern, M., Vaszil, G.: *P colonies with a bounded number of cells and programs.* Pre-Proceedings of the 7th Workshop on Membrane Computing (H. J. Hoogeboom, Gh. Păun, G. Rozenberg, eds.), Leiden, The Netherlands, 2006, pp. 311–322.
4. Floreano, D. and Mattiussi, C. (2008). Bio-inspirated Artificial Inteligence: Theories, Methods, and Technologies. MIT Press.
5. Freund, R., Oswald, M.: *P colonies working in the maximally parallel and in the sequential mode.* Pre-Proceedings of the 1st International Workshop on Theory and Application of P Systems (G. Ciobanu, Gh. Păun, eds.), Timisoara, Romania, 2005, pp. 49–56.
6. Kelemen, J., Kelemenová, A.: *On P colonies, a biochemically inspired model of computation.* Proc. of the 6th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest TECH, Hungary, 2005, pp. 40–56.
7. Kelemen, J., Kelemenová, A., Păun, Gh.: *Preview of P colonies: A biochemically inspired computing model.* Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems, ALIFE IX (M. Bedau at al., eds.) Boston, Mass., 2004, pp. 82–86.
8. Minsky, M. L.: *Computation: Finite and Infinite Machines.* Prentice Hall, Engle-wood Cliffs, NJ, 1967.
9. Păun, Gh.: *Computing with membranes.* Journal of Computer and System Sciences 61, 2000, pp. 108–143.
10. Păun, Gh.: *Membrane computing: An introduction.* Springer-Verlag, Berlin, 2002.
11. Păun, Gh., Rozenberg, G., Salomaa, A.: *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2009.
12. P systems web page: http://psystems.disco.unimib.it