An Application of the PCol Automata in Robot Control

Miroslav Langer[1], Luděk Cienciala[1], Lucie Ciencialová[1], Michal Perdek[1], and Alice Kelemenová[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, alice.kelemenova}@fpf.slu.cz

Summary. The P colonies were introduced in [6] as a variant of the bio-inspired computational models called membrane systems or P systems. In [2] we divided agents into the groups according the function they provide; we introduced the modularity on the P colonies. PCol automata are an extension of the P colonies by the tape (see [1]). This is an accepting computational device based on the very simple computational units. In this paper we combine the approach of the modules in the P colonies and of the PCol automata and we introduce the PCol automaton driven robot.

1 Introduction

Recently, the robotics has been more and more expanding and intervening in various branches of science like biology, psychology, genetics, engineering, cognitive science, neurology etc. An effort to create robots with an artificial intelligence which are able to cogitate or solve various types of problems refers to hardware and software limits. Many of these limits are managed to be eliminated by the interdisciplinary approach which allows creating new concepts and technics suitable for the robot control and facture of the new hardware.

Very robot control is often realised by the classical procedures known from the control theory (see [9]), concepts inspired by the biology, evolution concepts (see [5]) or with use of the decentralized approaches (see [8]).

The autonomous robot's behaviour and its control are realized by the control unit. Robots are equipped with the various types of sensors, cameras, gyroscopes and further hardware which all together represents the robots perception. These hardware components provide to the control unit the information about the actual state of the environment in which the robot is present and also the information about the internal states of the robot. After the transformation of these inputs there are generated a new data which are forwarded to the actuators like the wheels, robotic arm etc. Thus the robot can pass the obstacle by using the sensors and adjusting the speed of the particular wheels. So the objective of the control unit is to transform input signals to the output signals which consequently affect the behaviour of the robot. These changes in the behaviour cause that the robot interacts with the environment and in the sight of the observer the robot seems to be intelligent. Transformation of these signals can be done computationally in various ways with use of the knowledge or fuzzy knowledge systems, artificial neural networks, or just with use of the membrane systems or the P systems as it will be shown in this paper.

The development, design and the realization of the new approaches and techniques through which is possible to realize function of the control unit is one of the key subject of the development of the artificial intelligence and the robotics.

P colonies were introduced in 2004 as abstract computing devices composed from independent single membrane agents, reactively acting and evolving in a shared environment. P colonies reflect motivation from colonies of grammar systems, i.e. the idea of the devices composed from as simple as possible agents placed in a common environment; the system, which produce nontrivial emergent behaviour, using the environment only as the communication medium with no internal rules. P colonies consist of single cells “floating” in a common environment. Sets of rules of cells are structured to simple programs in P colonies. Rules in a program have to act in parallel in order to change all objects placed into the cell, in one derivation step. Objects are grouped into cells or they can appear in their completely passive environment in which these cells act. We assume that the environment contains several copies of the basic environmental objects (denoted in the formal definition of P colonies by e), as many as needed to perform a computation. Moreover the environment can contain also finite number of non-basic objects, and each entity contains a fixed (intuitively small) number of (possibly identical) objects.

Cells as basic computing agents of P colonies are of as much as possible restricted complexity and the capability. Each agent is associated with a small number of objects present inside it and with a set of rules forming programs for processing these objects.

Two types of rules are considered, namely the evolution rules acting inside agents, and the communication rules providing elementary interactions between the agents and the environment.

Each of the evolution rules is able to rewrite one object in the agent into another object which will remain inside this agent. Evolution rule is denoted by $a \rightarrow b$. The communication rules consist in the mutual exchanging of one object inside the agent, and one object in its environment. We denote communication rule by $c \leftrightarrow d$, where the object appearing in the agent is written in the left side of the relation $\leftrightarrow$. Moreover checking rules are considered to extend the abilities of agents follows: assume that communication rule can be chosen from two possibilities with the first one having higher priority. The rule associated with the agent with greater priority has to be active. The agent checks the possibility to execute the communication rule having higher priority. Otherwise, the second communication rule can be treated. We denote a checking rule being a pair $c \leftrightarrow d/c^{'} \leftrightarrow d^{'}$. A *P colony with checking rules will be called also a P colony with priority.*

The program of an agent allows changing all objects in the cell simultaneously and deterministically by different rules, so the number of objects in an agent is identical with the number of rules in each of its programs.

P colony starts a computation with given objects in the environment and in each agent. We associate a result with a halting computation, in the form of the number of copies of a distinguished object in the environment. Both parallel as well as sequential computational mode of P colonies is discussed depending on the amount of agents acting in one derivation step. In the first case, each agent which can apply any of its programs has to choose one non-deterministically, and apply it; in the sequential case one agent, non-deterministically chosen, is allowed to act. P colonies are computationally complete, i.e. all the number sets computable by Turing machines are computable also by P colonies. This gives the interpretation that the environment is essential as a medium for communication and for storing information during the computation, even with no structure and no information in the environment at the beginning of the computation. The power of cooperating agents of a very restricted form can be dramatically different from the power of individual agents. For overview on P colonies we refer to [7], [3], [4].

Pcol automaton was introduced in order to describe the situation, when P colonies behave according to the direct signals from the environment (see [1]). This modification of the P colony is constructed in order to recognize input strings. In addition to the writing and communicating rules usual for a P colony cells in Pcol automata have also tape rules. Tape rules are used for reading next symbol on the input tape and changing an object in cell(s) to the read symbol. Depending on the way how tape rules and other rules can take a part in derivation process several computation modes are treated. After reading the whole input word, computation ends with success if the Pcol automaton reaches one of its accepting configurations. So, in accordance with finite automata, Pcol automata are string accepting devices based on the P colony computing mechanisms.

Agents can be grouped to different modules specified for some activities as illustrated in [2] for P colonies. This approach will be used in the present paper for Pcol automata illustrated in the case of robot control.
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 colony is a computing device composed from the environment and the independent organisms called agents or cells. The agents live in the environment. Each agent is represented by a collection of objects embedded in a membrane, which is constant during the computation. A set of programs, which are composed from the rules, is associated with each agent. The rule can be either evolution rule or communication rule. 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 communication rules are of the form $c \leftrightarrow d$. When the communication rule is performed, the object c inside the agent and the object d in the environment swap their places. Thus after execution of the rule, the object d appears inside the agent and the object c is placed in the environment.

In [6] the set of programs was extended by the checking rules. These rules give an opportunity to the agents 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 environment contains several copies of the basic environmental object denoted by e. The number of the copies of e in the environment is sufficient, it means that each agent which wants to receive the symbol e from the environment using the communication rule will receive it.

We will handle parallel model of P colonies with checking rules (denoted by $NPCOL_{par}K$) in this paper. At each step of the parallel computation each agent tries to apply one usable program. If the number of applicable programs is higher than one, then the agent chooses one of the rules nondeterministically and the maximal possible number of agents is active at each step of the computation. 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

The research of the P colonies suggested that particular agents are providing the same function during the computation. This served as the inspiration to introduce the modules in the P colonies. In the [2] we grouped agents of the P colony simulating computation of the register machine into the modules. The agents providing subtraction were classified into the subtraction module, agents providing addition were sorted 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.

The inspiration to introduce modularity was found in living organisms where group of cells providing one function evolved into the organs and whole organism is composed by these organs.

In this paper, this approach to define modules will be used for the robot control. One module for each module of the robot (sensors, actuators) will be defined. For planning the robots action will be used the tape; PCol automaton.

2.2 PCol automata

By extending the P colony by the input tape we obtain a string accepting/ recognizing device; the PCol automaton (see [1]). The input tape contains the input string which can be read by the agents. The input string is the sequence of the symbols. To access the tape the agents use special tape rules (T-rules). The rules not accessing the tape are called non-tape rules (N-rules). The computation and use of the T-rules is very similar to the use of the rules in the P colonies. Once any of the agents uses its T-rule, the actual symbol on the tape is considered as read. The only difference between the tape and the environmental symbol is that the tape symbol can access arbitrary many agents at the same time.

Definition 1. *PCol automaton of the capacity k and with n agents, $k, n \geq 1$ is a construct*

$$\Pi = (A, e, V_E, (O_1, P_1), \dots (O_n, P_n), F), \text{ where}$$

- *A is a finite set, an alphabet of the PCol automaton, its elements are called objects;*
- *e is an environmental object, $e \in A$;*
- *V_E is a multiset over $A - \{e\}$ defining the initial content of the environment;*
- *$(O_i, P_i), 1 \leq i \leq n$ is an i-th agent*
  - *O_i is a multiset over A defining the initial content of the agent, $|O_i| = k$,*
  - *P_i is a finite set of the programs,$P_i = T_i \cup N_i,\ T_i \cap N_i = \emptyset$, where every program is formed from k rules of the following types:*
    - *the tape rules (T-rules for short)*
      - *$a \overset{T}{\rightarrow} b$ are called the rewriting T-rules;*
      - *$a \overset{T}{\leftrightarrow} b$ are called the communicating T-rules;*
    - *the non-tape rules (N-rules for short)*
      - *$a \rightarrow b$ are called the rewriting N-rules;*
      - *$c \leftrightarrow d$ are called the communicating N-rules;*
    - *T_i is a set of tape programs (T-programs for short) consisting from one T-rule and $k-1$ N-rules.*
    - *N_i is a set of non-tape programs (N-programs for short) consisting only from N-rules.*
- *F is a set of accepting configurations of the PCol automaton, each state is a $(n+1)$-tuple $(v_E; v_1, \dots, v_n)$, where:*
  - *$v_E \subseteq (A - \{e\})^*$ is a multiset of the objects different from the object e placed in the environment;*
  - *$v_i,\ 1 \leq i \leq n$ is a content of the i-th agent;*

The configuration of the PCol automaton is $(n+2)$-tuple $(w_T; w_E; w_1, \dots, w_n)$, where $w_T \in A^*$ the unprocessed (unread) part of the input string, $w_E \in (A - \{e\})^*$ is a multiset of the objects different from the object e placed in the environment of the PCol automaton and $w_i,\ 1 \le i \le n$ is a content of the i-th agent.

The computation starts in the starting configuration defined by the input string on the tape the initial content of the environment and the initial content of the agents. Actual symbol on the input tape we consider as read iff at least one agent uses its T-program in the particular derivation step.

We define the rule r in following way:

$$r = \left(a \xrightarrow{T/\llcorner\!\lrcorner} b\right) \Rightarrow \begin{cases} left\,(r) = a \\ right\,(r) = b \\ export\,(r) = \varepsilon \\ import\,(r) = \varepsilon \end{cases}$$

$$r = (c \overset{T/\llcorner\!\lrcorner}{\longleftrightarrow} d) \Rightarrow \begin{cases} left\,(r) = \varepsilon \\ right\,(r) = \varepsilon \\ export\,(r) = c \\ import\,(r) = d \end{cases}$$

For each configuration $(w_E, w_1, \dots, w_n)$ we define set of applicable programs $P_{(w_E, w_1, \dots, w_n)}$:

- $\forall p, p' \in P, p \neq p', p \in P_i, p' \in P_j \Rightarrow i \neq j$
- for each $p \in P$ and $p \in P_i$ $left(p) \cup export(p) = w_i$
- $\bigcup\limits_{p \in P} import(p) \subseteq w_E$

For each configuration $(w_E, w_1, \dots, w_n)$ we define set of all sets of applicable programs $\mathcal{P}_{(w_E, w_1, \dots, w_n)}$

For the configuration $(w_E, w_1, \dots, w_n)$ and the input symbol a we define:

- **t-transition,** $\Rightarrow_t^a$: If there is at least one set of applicable programs $P \in \mathcal{P}$ such that each $p \in P$ is the T-program with T-rule of the form $x \overset{T}{\rightarrow} a$ or $x \overset{T}{\leftrightarrow} a, x \in A$ and P is the maximal set (there does not exists other set $P' \in \mathcal{P}$ where $|P'| > |P|$ fulfilling stated conditions).
- **n-transition,** $\Rightarrow_n$: If there is at least one set of applicable programs $P \in \mathcal{P}$ such that each $p_i \in P$ is the N-program and P is the maximal set.
- **tmin-transition** $\Rightarrow_{tmin}^a$: If there is at least one set of applicable programs $P \in \mathcal{P}$ such that there exists at least one program P is the T-program and it is in the form $x \overset{T}{\rightarrow} a$ or $x \overset{T}{\leftrightarrow} a, x \in A$, it can contain also the N-programs and P is the maximal set.
- **tmax-transition** $\Rightarrow_{tmax}^a$: If there is at least one set of applicable programs $P \in \mathcal{P}$ such that $P = P_N \cup P_T$ where P_N is a set of nontape programs and P_T is a maximal set of applicable tape programs of the form $x \overset{T}{\rightarrow} a$ or $x \overset{T}{\leftrightarrow} a, x \in A$, and $P = P_N \cup P_T$ is maximal;

We denote

$(w_E, w_1, \ldots, w_n) \Rightarrow^{a/\llcorner}_{trans} (w'_E, w'_1, \ldots, w'_n)$,
$trans = \{t, n, tmin, tmax\}$
where: for each j, $1 \leq j \leq n$ for which there exists $p \in P \wedge p \in P_j$, $w'_j = right(p) \cup import(p)$, if there does not exists $p \in P \wedge p \in P_j$ so $w'_j = w_j$; $w'_E = w_E - \bigcup_{p \in P} import(p) \cup \bigcup_{p \in P} export(p)$.

PCol automaton works in the *t(tmax, tmin) mode of computation* if it uses only t- (tmax-, tmin-) transitions. PCol automaton works in the *nt (ntmax or ntmin) mode of computation* if at any computation step it may use a t- (tmax- or tmin-) transition or an n-transition. A special case of the *nt* mode is called *initial*, denoted by *init*, if the computation of the automaton is divided in two phases: first it reads the input strings using t-transitions and after reading all the input symbols it uses n-transitions to finish the computation.

The computation ends by (types of acceptance):

halting (halt) - there does not exist an applicable set of programs corresponding to the computation mode. Computation is successful if it ends and the whole input tape is read.

reading the last input symbol (lastsym) - the computation (successfully) ends if the last input symbol is read and there does not exist set of applicable programs corresponding to the computation mode. The computation is unsuccessful if it ends before reading the last symbol from the input tape.

final state reached (finstate) - the computation ends whenever the last symbol is read as far as the automaton would not stop further. The computation is successful if the input tape is read and PCol automaton reaches the configuration from the set of the final states F.

The language accepted by the PCol automaton Π is defined as a set of the words for which there exist successful computation in particular mode and type of acceptance.

Definition 2. $L(\Pi, mod, acc) = \{w \in A^* | w$ *is accepted by the computation in the mode mod with type of acceptance acc* $\}$,
where $mod \in \{t, nt, tmax, ntmax, tmin, ntmin, init\}$ *and* $acc \in \{halt, lastsym, finstate\}$.

3 Robot control using the PCol automaton

Main advantage of using PCol automaton in the controlling robot behaviour is the parallel proceeding of the data done by very primitive computational units using very simple rules.

By conjunction modularity and PCol automaton we obtain a powerful tool to control robot behaviour. PCol automaton is parallel computation device. Collaterally working autonomous units sharing common environment provide 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. Input tape gives us an opportunity to plan robot actions. Each input symbol represents a single instruction which has to be done by the robot, so the input string is the sequence of the actions which guides the robot to reach his goal; performing all the actions.

We construct a PCol automaton with four modules: *Control unit*, *Left actuator controller*, *Right actuator controller* and *Infra-red receptor*. Entire automaton 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 control unit is the main module which controls the computation. It reads the sequence of the actions from the input tape. According to the type of the action read from the tape it asks the data from the sensors modules by sending particular objects into the environment. If the answer from the sensors allows to perform the action, the control unit sends the command to the actuator controllers to perform demanded action. After sending the command to the actuator controllers the control unit waits for the announcement of the successful or the unsuccessful performance of the demanded action from the actuator controllers. If the action was fulfilled then the control unit continues in reading the input tape and performs following action.

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 left and right 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.

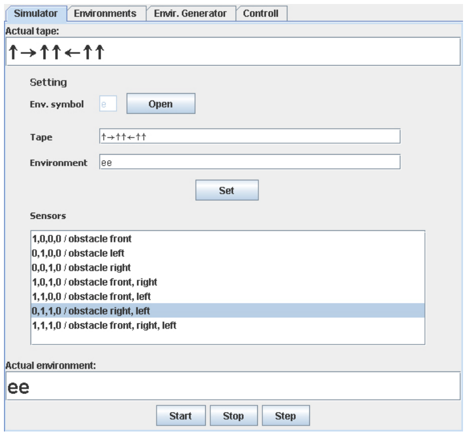
Let us introduce the formal definition of the mentioned PCol automaton: $\Pi = (A, e, V_E, (O_1, P_1), \dots (O_7, P_7), \emptyset)$, where
$A \;= \{0_L, 0_R, 1_L, 1_R, e, F_F, \overline{F_F}, F_L, \overline{F_L}, F_R, \overline{F_R}, G_F, G_L, G_R, I_F, I_L, I_R, M_F,$
$M_L, M_R, N_F, \overline{N_F}, N_L, \overline{N_L}, N_R, \overline{N_R}, R, R_T, W_F, W_L, W_R, W_T\}$,
$V_E = \{e\}$,
Control Unit:

$O_1 = \{\ T, e\}$,
$P_1 = \{ < R_T \xrightarrow{T} M_F; e \to e >; < G_F M_F \to M_F M_F >; < M_F M_F \to ee >;$
$< e \leftrightarrow G_F/e \leftrightarrow R; M_F \to M_F >; < RM_F \to eW_T >;$
$< R_T \xrightarrow{T} M_L; e \to e >; < G_L M_L \to M_L M_L >; < M_L M_L \to ee >;$
$< e \leftrightarrow G_L/e \leftrightarrow R; M_L \to M_L >; < RM_L \to eW_T >;$
$< R_T \xrightarrow{T} M_R; e \to e >; < G_R M_R \to M_R M_R >; < M_R M_R \to ee >;$
$< e \leftrightarrow G_R/e \leftrightarrow R; M_R \to M_R >; < RM_R \to eW_T >;$
$< ee \to eW_T >; < W_T \to e; e \leftrightarrow R_T > \}$
$O_2 = \{\ T, e\}$,
$P_2 = \{ < R_T \xrightarrow{T} M_F; e \to I_F >; < I_F \leftrightarrow e; M_F \to W_F >;$
$< W_F \to e; e \leftrightarrow F_F/e \leftrightarrow N_F >; < F_F \to G_F; e \to e >;$
$< G_F \leftrightarrow e; e \to W_T >; < R_T \xrightarrow{T} M_L; e \to I_L >;$
$< I_L \leftrightarrow e; M_L \to W_L >; < W_L \to e; e \leftrightarrow F_L/e \leftrightarrow N_L >;$
$< F_L \to G_L; e \to e >; < G_L \leftrightarrow e; e \to W_T >; < R_T \xrightarrow{T} M_R; e \to I_R >;$
$< I_R \leftrightarrow e; M_R \to W_R >; < W_R \to e; e \leftrightarrow F_R/e \leftrightarrow N_R >;$
$< F_R \to G_R; e \to e >; < G_R \leftrightarrow e; e \to W_T >; < R \leftrightarrow e; e \to W_T >;$
$< W_T \to e; e \leftrightarrow R_T > \}$

Infra-red module:
$O_3 = \{\ e, e\}$,
$P_3 = \{ < e \leftrightarrow \overline{F_F}; e \to F_F >; < F_F \leftrightarrow I_F/F_F \to e; \overline{F_F} \to e >;$
$< e \leftrightarrow \overline{N_F}; e \to N_F >; < N_F \leftrightarrow I_F/N_F \to e; \overline{N_F} \to e >;$
$< I_F e \to ee; >\}$
$O_4 = \{\ e, e\}$,
$P_4 = \{ < e \leftrightarrow \overline{F_L}; e \to F_L >;$
$< F_L \leftrightarrow I_L/F_L \to e; \overline{F_L} \to e >; < e \leftrightarrow \overline{N_L}; e \to N_L >;$
$< N_L \leftrightarrow I_L/N_L \to e; \overline{N_L} \to e >; < I_L e \to ee; >\}$
$O_5 = \{\ e, e\}$,
$P_5 = \{ < e \leftrightarrow \overline{F_R}; e \to F_R >;$
$< F_R \leftrightarrow I_R/F_R \to e; \overline{F_R} \to e >; < e \leftrightarrow \overline{N_R}; e \to N_R >;$
$< N_R \leftrightarrow I_R/N_R \to e; \overline{N_R} \to e >; < I_R e \to ee; >\}$

Left Actuator controller:
$O_6 = \{\ e, e\}$,
$P_6 = \{ < e \leftrightarrow M_F; e \to 1_L >; < M_F \to R_T; 1_L \leftrightarrow e >;$
$< e \leftrightarrow M_R; e \to 1_L >; < M_R \to R_T; 1_L \leftrightarrow e >;$
$< e \leftrightarrow M_L; e \to 0_L >; < M_L \to R_T; 0_L \leftrightarrow e >;$
$< R_T \leftrightarrow e; e \to e >\}$

Right Actuator controller:
$O_7 = \{\ e, e\}$,
$P_7 = \{ < e \leftrightarrow M_F; e \to 1_R >; < M_F \to R_T; 1_R \leftrightarrow e >;$
$< e \leftrightarrow M_R; e \to 0_R >; < M_R \to R_T; 0_R \leftrightarrow e >;$
$< e \leftrightarrow M_L; e \to 1_R >; < M_L \to R_T; 1_R \leftrightarrow e >;$
$< R_T \leftrightarrow e; e \to e >\}$



Actual tape:
↑→↑↑←↑↑

Setting
Env. symbol e
Tape ↑→↑↑←↑↑
Environment ee

Sensors
1,0,0,0 / obstacle front
0,1,0,0 / obstacle left
0,0,1,0 / obstacle right
1,0,1,0 / obstacle front, right
1,1,0,0 / obstacle front, left
0,1,1,0 / obstacle right, left
1,1,1,0 / obstacle front, right, left

Actual environment:
ee

Fig. 1. Simulator

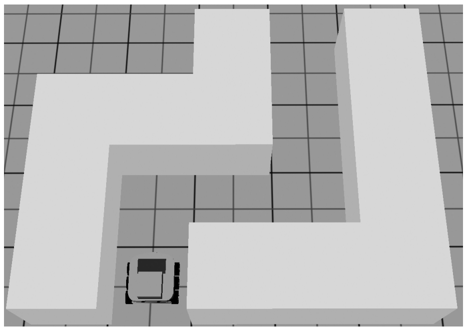


Fig. 2. Starting position

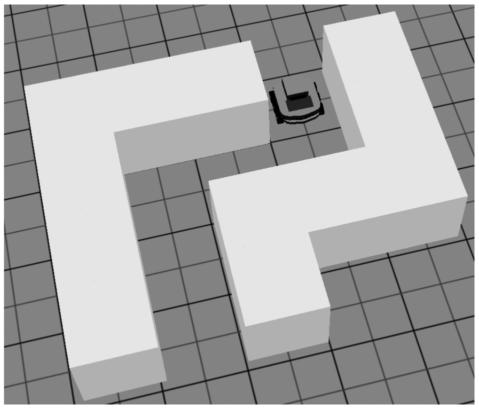


Fig. 3. Ending position

The robot driven by this very simple PCol automaton is able to follow the instruction on the tape safely without crashing into any obstacle. If the instruction cannot be proceeded, the robot stops. This solution is suitable for known robots environment. Following the instructions on the tape (picture 1) the robot can move from its starting position (picture 2) to the final destination (picture 3). If the environment is changed before or during the journey, the robot cannot reach the final place but it also will not crash.

4 Conclusion

We have shown the basic possibilities of controlling the robot using the PCol automaton and modular approach. With respect to the fact that P colonies are computationally complete device (see [2]) the further research will be dedicated to the more precise control and the possibilities of processing the information from other sensors, especially from the camera. Fulfilling more complex tasks and more autonomous behaviour (e.g. attempt go round the obstacle if it is not possible to go in demanded direction, skipping unrealizable tasks, etc.) is also direction of the further research.

By extending the robot by the acting modules like e.g. mechanic tongs the robot can fulfil more complicated tasks. To control such a device by the PCol automaton we just need to add a new control module and extend set of programs of the main control unit. Such an extension is thanks to the modularity very easy.

Remark 1.
This work was partially supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), by SGS/7/2011 and by project OPVK no. CZ.1.07/2.2.00/28.0014.

References

1. Cienciala, L., Ciencialová, L. Csuhaj-Varjú, Vazsil, G.: *PCol Automata: Recognizing Strings with P colonies.* Eight Brainstormung Week on Membrane Computing (In Martínez del Amor, M. A., Păun, G., Hurtado de Mendoza, I. P., Riscon-Núnez, A. (eds.)), Sevilla, 2010, pp. 65–76.
2. Cienciala, L., Ciencialová, L., Langer, M.: *Modularity in P Colonies with Checking Rules.* In: Revised Selected Papers 12 th International Conference CMC 2011 (Gheorge, M., Păun, Gh., Rozenber, G., Salomaa, A., Verlan, S. eds.), Springer, LNCS 7184, 2012, pp. 104-120.
3. 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.
4. Csuhaj-Varjú, E., Margenstern, M., Vaszil, G.: *P Colonies with a bounded number of cells and programs.* Pre-Proceedings of the 7^{th} Workshop on Membrane Computing (H. J. Hoogeboom, Gh. Păun, G. Rozenberg, eds.), Leiden, The Netherlands, 2006, pp. 311–322.

5. Floreano, D. and Mattiussi, C.: *Bio-inspirated Artificial Inteligence: Theories, Methods, and Technologies*, MIT Press, 2008.
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. Kelemenová, A.: *P Colonies.* In: The Oxford Handbook of Membrane Computing eds. Gh.Paun, G. Rozengerg, A. Salomaa Oxford University Press, Oxford, 2009, 584–593
8. Weiss, G.: *Multiagent systems. A modern approach to distributed artificial intelligence*, MIT Press, Cambridge, Massachusetts, 1999
9. Wit, C. C., Bastin, G., Siciliano, B.: *Theory of Robot Control*, Springer-Verlag New York, 1996.