International Journal of Imaging and Robotics,
[Formerly known as the "International Journal of Imaging" (ISSN 0974-0627)]
Volume 17; Issue No. 1; Year 2017; Int. J. Imag. Robot.
ISSN 2231–525X;

Robot Motion Planning Inside a Grid Using Membrane Computing

Williams Sureshkumar, Kalpana Mahalingam, Raghavan Rama *

Department of Mathematics,
Indian Institute of Technology, Madras, Chennai-36.
wisureshkumariit@gmail.com, kmahalingam@iitm.ac.in, ramar@iitm.ac.in

ABSTRACT

The motion of a robot inside a rectangular grid is simulated using Isotonic Array P System (IAPS)defined by Sureshkumar and Rama in 2015. A polynomial time algorithm has been proposed using this grammatical model to find the shortest path with and/or without obstacles. The obstacles are usually assumed be of polygonal shape in the rectangular grid.

Keywords: Isotonic array grammar, Robot motion planning, Collision free path, Rectangular grid, Priority.

Subject Classification: 68Q05, 68Q42, 68Q45.

1 Introduction

P system is a parallel distributed computing model that can hold any data structure in its membranes and evolve them. The branch of study of several variants of P systems is popularly known as *Membrane Computing* (Păun, Rozenberg and Salomaa, 2010). Membranes are the main structures in P systems holding the evolution that is taking place. Most of the P system variants are computationally universal, exhibiting the power of such systems.

One of the important problem in robotics is to determine the path of a robot in either 2-dimensional or 3-dimensional space from its initial to its target position. The problem becomes challenging if the space contains obstacles. Miloš et al, proposed a method using rectilinear Voronoi diagrams to find a collision free path of a robot in eight directions. An algorithm based on consensus among mobile robot agents to the problem of collision avoidance of virtual robots in 2-dimensional and 3-dimensional space has been proposed in (Soriano, Bernabeu, Valera and Vallés, 2014). Saudi et al, identifies a potential path for mobile robot motion using Laplace's equation (Saudi and Sulaiman, 2013).

*This work was funded by the Project No: MAT/15-16/046/DSTX/KALP, Department of Science and Technology, Government of India.

The task of finding a free space of a mobile robot can be formulated in many ways depending on various conditions. In this paper, we focus on a special case of motion planning in 2D for a completely known scene with static polygonal obstacles.

In this paper we propose a grammatical model using P systems with isotonic arrays as data stucture to simulate the movement of a robot in 4-direction in a rectangular grid. The diagonal movements are avoided in our model. The concept of $IAPS$ was earlier introduced in (Sureshkumar and Rama, 2016b). In (Sureshkumar and Rama, 2016a), we defined labeled $IAPS$ where the robot motion is studied. The set of labels of the rules decides the path. In this paper we use $IAPS$ for the same purpose. However the present model is more general and extracts the shortest path.

We recall the necessary basic notions required for this work in section 2. The procedure and simulation of robot motion using $IAPS$ is presented in section 3 and section 4. Three different cases of robot motion are discussed. Some pointers towards future directions research is given as concluding remark.

2 Preliminary

We refer the reader to (Rozenberg and Salomaa, 1997) for basic notions, notations and results for the concepts of formal language theory. We also define Isotonic Array Grammar (IAG) and Isotonic Array P systems ($IAPS$), which will be used in this paper.

2.1 *Isotonic Array Grammar*

Definition 2.1. (Rosenfeld, 1971; Cook and Wang, 1978) An Isotonic Array Grammar is a quintuple $G = (N,\ T,\ \#,\ P,\ S)$, where:

- N is a finite non-empty set of symbols called non-terminals;
- T is a finite non-empty set of symbols called terminals such that it is disjoint from N that is $N \cap T = \emptyset$;
- $\# \notin (N \cup T)$ is the background or blank symbol;
- P is a finite non-empty set of productions (or rewriting rules) of the form $\alpha \to \beta$, where α and β are identical in shape (identical except for the symbols involved) over $N \cup T \cup \{\#\}$; α not all #'s;
- $S \in N$ is the start symbol;

We say that an array v is directly derived from an array u, denoted by $u \Rightarrow v$, if there is a production $\alpha \rightarrow \beta$, for a sub array α contained in u, and v is obtained from u by replacing α with β. The array v is identical in shape to u. The set of all arrays of terminal symbols and #'s that can be generated from the start symbol S is called the language generated by an isotonic array grammar G, and is denoted by $L(G)$. It consists only of arrays over terminals. The empty array is denoted by Λ.

An isotonic production without #'s does not allow the array to grow. So to avoid this class of productions, the background symbol # has been introduced so that the non background portion of the array can expand into the background. The purpose of background symbol here is similar to that of the blank symbol on the tape of the Turing machine (Rozenberg and Salomaa, 1997).

As in the case of string grammar, a hierarchy of grammars with isotonic rules can be defined by imposing restrictions in the form of productions. The image of an array square on the left side of the production is the symbol in the corresponding square on the right side. An array is connected if there is a path of non-# symbols between every pair of non-# squares.

Definition 2.2. (Rosenfeld, 1971) Let $G = (N,\ T,\ \#,\ P,\ S)$ be an isotonic array grammar.

1. G is of type 0 if there is no restriction on P.

2. G is of type 1 or *monotonic* if both sides of each production is connected and the image of each left side symbol in $N \cup T$ is in $N \cup T$; that is #'s cannot be created.

3. G is of type 2 or *context-free* if the right side of each production is connected, the left side contains exactly one non-terminal in a field of #'s, and the image has every symbol in $N \cup T$. e.g.
$$\begin{matrix} \# \\ A\# \end{matrix} \rightarrow \begin{matrix} A \\ aB \end{matrix}, \quad \begin{matrix} \# \\ \# \\ A\# \end{matrix} \rightarrow \begin{matrix} A \\ B \\ AB \end{matrix},$$
where A and B are nonterminal symbols and a is a terminal symbol.

4. G is of type 3 or *regular* if every production is of the form
$$\#A \rightarrow Ba, \quad A\# \rightarrow aB, \quad \begin{matrix} \# \\ A \end{matrix} \rightarrow \begin{matrix} B \\ a \end{matrix}, \quad \begin{matrix} A \\ \# \end{matrix} \rightarrow \begin{matrix} a \\ B \end{matrix} \quad or \quad A \rightarrow a,$$
where A and B are nonterminal symbols and a is a terminal symbol.

Let IAG, $IMAG$, $ICFAG$ and $IRAG$ denote the isotonic, isotonic monotonic, isotonic context-free and isotonic regular array grammar respectively.

Now, we define the isotonic *restricted monotonic* array grammar G.

Definition 2.3. (Sureshkumar and Rama, 2016b) An isotonic grammar G is said to be *restricted monotonic* if for each isotonic array rule $\alpha \to \beta$, α contains exactly one non-terminal symbol and α may contain #'s and terminal symbols. Image of each non-# symbol of α is a non-# symbol whereas image of each # symbol is either a # symbol or a non-# symbol. Hence, each symbol of β is in $N \cup T \cup \{\#\}$. Note that #'s cannot be created. We call this new restricted grammar by Isotonic Restricted Monotonic Array Grammar($IRMAG$). For example rules look like

$$\begin{matrix} 0\,0 \\ \#A \end{matrix} \to \begin{matrix} 0\,0 \\ A\,1 \end{matrix}, \quad \begin{matrix} \#\,0 \\ \#\,B \end{matrix} \to \begin{matrix} \#\,0 \\ \#\,1 \end{matrix}.$$

Throughout this paper we only use isotonic restricted monotonic rules. Hereafter if we mention isotonic rules, it means isotonic restricted monotonic rules.

2.2 Isotonic Array P Systems

Definition 2.4. (Sureshkumar and Rama, 2016b) An Isotonic Array P Systems ($IAPS$) of degree m (≥ 1) is a construct $\Pi = \big(V,\ T,\ \#,\ \mu,\ I_1,\ldots,I_m,\ (R_1,\rho_1),\ldots,(R_m,\rho_m),\ i_o\big)$, where V is the alphabet of terminals and non-terminals, $T \subseteq V$ is the terminal alphabet, $\# \notin V$ is the blank or background symbol. Here μ represents the membrane structure and the membranes are numbered as $1,2,\ldots,m$. Membrane 1 is the outermost membrane or skin membrane. The set of initial arrays are I_i associated with region i, $1 \leq i \leq m$. For the definition of membrane region the reader is referred to (Păun, 2000). R_i is the set of isotonic array rules associated with region i, $1 \leq i \leq m$. ρ_i describes the priority among the rules in R_i, $1 \leq i \leq m$. The rules are written as $\mathscr{C} \to \mathscr{D}\ (tar)$, where $\mathscr{C},\mathscr{D}$ are isotonic arrays and $tar \in \{here, out, in\}$ which means that the array $\mathscr{C}$ is replaced by an identical (in shape) array $\mathscr{D}$ and the resultant modified array is sent to a membrane indicated by tar. Here, i_o is the output membrane.

The evolution in the above defined system happens in a parallel distributed manner. However, each array will be computed by only one rule at a time. The evolution of the arrays may also be dominated by priorities. An evolution on the array may or may not always halt, there will be no rules that are applicable to the array any more. Such an output condition is in general ignored or avoided by suitable array rewriting rules. Hence, the output of this system will be only the collection of arrays that appear in a designated output membrane for which the system halts. Moreover, the output is a collection of arrays over terminal symbols.

The set of arrays generated by an $IAPS$ Π is denoted by $A(\Pi)$. The family of sets $A(\Pi)$ generated by $IAPS$ of degree at most m with or without priority rules is denoted by $AIAP_m$. If the number of membranes is unbounded, then the subscript m is replaced by $\star$.

Example 2.1. *In this example the $IAPS$ Π with two membranes is given as,*
$\Pi = \Big(\Big\{A,B,C,0,1\Big\},\Big\{0,1\Big\},\#,[_1[_2]_2]_1,I_1,I_2,R_1,R_2,2\Big)$*, where* $I_1 = \Big\{A\Big\}, I_2 = \emptyset$ *and*

$$R_1 = \Bigg\{ 1)\ \ A\#\# \ \to\ 11B\ ,$$

$$R_2 = \left\{ \begin{array}{ll} 2) & B\# \rightarrow 1B \ , \\ 3) & \left.\begin{array}{c} B \\ \# \end{array} \rightarrow \begin{array}{c} 1 \\ C \end{array}, in_2 \right\}, \\ & \begin{array}{ll} 1\,1 & 1\,1 \\ \#B & C\,1 \\ \#\# & 0\,1 \\ \#\# & 1\,1 \end{array} \\ 4) & \rightarrow \ , \\ & \begin{array}{lll} \#\,1\,1 & \#1\,1 & 1\,1 \\ \#C & 1\,0 & \#\,C \\ \#0 & 1\,0 & \#\,0 \\ \#1 & 1\,1 & \#\,1 \end{array} \\ 5) & \begin{array}{l} \\ \rightarrow \\ \\ \end{array} \begin{array}{l} \\ 1\,0 \\ \\ \end{array} > 6) \begin{array}{l} \\ \rightarrow \\ \\ \end{array} \left.\begin{array}{l} \\ C\,0 \\ 0\,0 \\ \\ \end{array}\right\}. \end{array} \right.$$

In this example $IAPS$ Π generates the set of all solid rectangles (with fixed breadth) whose edges are marked with 1 and all the internal pixels are marked with 0 (see Figure 1).

$$\begin{array}{cccc} 1 & 1 & 1 & \\ 1 & 0 & 1 & \\ 1 & 0 & 1 & , \\ 1 & 1 & 1 & \end{array} \quad \begin{array}{ccccc} 1 & 1 & 1 & 1 & \\ 1 & 0 & 0 & 1 & \\ 1 & 0 & 0 & 1 & , \\ 1 & 1 & 1 & 1 & \end{array} \quad \begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array} , \ldots$$

Figure 1: The arrays generated by $IAPS$ in Example 2.1

The working of $IAPS$ Π is as follows: Starting with axiom A, the isotonic array rules in region 1 generate a horizontal string of the form 1^n (n is the length of the rectangle) which will be the top row of the solid rectangle to be generated and expel the array to region two. In region 2, apply rule '4' once and the higher priority rule '5' will halt the computation. So after application of rule '6', $(n-3)$ times followed by an application of rule '5', the process stops.

3 $IAPS$ for Robot Motion Planning

The objective of this section is to find a collision free path which simulates the movement of a robot in a square grid using $IAPS$. We also assume that the grid contains obstacles which are polygonal in shape. We assume the following:

i) The robot is assumed to move on a 2-dimensional grid of dimension $n \times n$, where $n \in \mathbb{N}$.

ii) The robot can move only in the following four directions (see Figure 2):

   a) upward
   b) downward
   c) leftward
   d) rightward

iii) The grid may or may not contain obstacles.

iv) The obstacles are polygonal in shape.

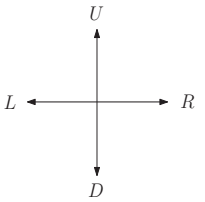


Figure 2: Valid directions of robot motion

The problem is to find a collision free path of the robot from the source to the target indicated in the grid by small disk. The path will be a sequence of cells between the source and the target. The initial grid representations are given in Figure 4a, Figure 6a and Figure 10b. The collision free path will be indicated in the output grid (see Figure 4b, Figure 6b and Figure 10c). We always consider a square grid of size n. The cardinality of the search space will then be $4^{2n} = 2^{4n}$. Hence, even for reasonably small values of n, there will be an intractable amount of possible paths. Clearly achieving an optimal path in a reasonable amount of time becomes difficult. We address this issue by considering a parallel distributed computing method. A P system is defined for this purpose.

The size of the robot is assumed to be smaller than the cell size of the grid. The information about the scene (or grid) is made known initially.

We discuss the three different cases of scenes for robot motion,

1. scene without obstacles.
2. scene with obstacles distributed everywhere in the grid.
3. scene with polygonal obstacles in the grid.

The robot is placed latter in the scene by means of little disk, (i.e.,) the initial and the target positions of the robot are indicated by little disks.
The scene is transferred as a 2D array as follows for computational purposes.

- the boundary of the square array corresponding to the scene is filled with 1's.
- the initial and the target positions are denoted by a non-terminal A and a $\$$ symbol in the array respectively.
- $+$ denotes the stretch of the obstacles.

- d is a symbol used to represent the stretch of the array cells not covered by obstacles.

- The collision free path is a sequence of $\star$'s or 0's connecting the source 'A' with a target '\$'.

3.1 Robot motion on a scene without obstacles

The first movement of the robot is done on a scene without obstacles. The initial scene, the robot's initial and target positions are transferred as an array as given in Figure 3. For this situation, it is enough to move the robot only in 2-directions (rightward and upward). A grid representation of a plane without obstacles is shown in Figure 4a.

The $IAPS$ Π with one membrane is constructed for this situation as follows:

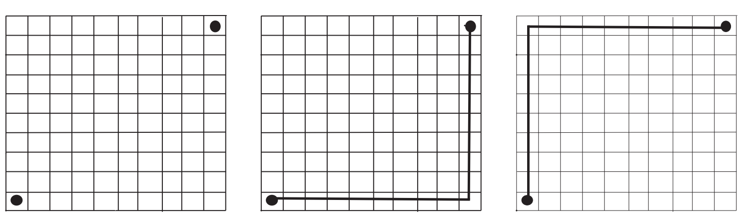
$\Pi = \left(\{A,\star,1,d,\$\},\{\star,1,d,\$\},\#,[_1]_1,I_1,R_1,1\right)$, where

$R_1 = \left\{ 1)\ \begin{matrix} d \\ A \end{matrix} \rightarrow \begin{matrix} A \\ \star \end{matrix}\ ,\ 2)\ Ad \rightarrow \star A\ ,\ 3)\ \begin{matrix} \$ \\ A \end{matrix} \rightarrow \star\star\ ,\ 4)\ A\$ \rightarrow \star\star \right\}$ and I_1 is given in Figure 3.

1	1	1	1	1	1	1	1	1	1	1	1
1	d	d	d	d	d	d	d	d	d	\$	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	A	d	d	d	d	d	d	d	d	d	1
1	1	1	1	1	1	1	1	1	1	1	1

Figure 3: Initial array representation of Π

Π generates all possible paths from the starting to the target positions. Two typical paths from the starting to the target positions on a grid of size 10×10 are given in Figure 4b and 4c.



(a) a) Grid without obstacles (b) b)Path from source to target (c) c) Another path

Figure 4: Robot motion without obstacles

It is obvious that the problem is of combinatorial in nature and its time complexity depends on the granularity of the grid and distribution of obstacles (in this case no obstacle is present). Even though the paths have fixed lengths, the complexity remains exponential.

Here the square grid is of size $n \times n$ and the robot can move only in two directions (right and upward). The cardinality of the search space is equal to 2^{2n}, which, even for not very high values of n, leads to a rather intractable amount of possible paths. For example, when $n = 60$ we get $2^{2(60)} = 2^{120} = (2^{10})^{12} = (1024)^{12} > (10)^{36}$ paths, which gives no chance to find a successful path in a reasonable amount of time.

The number of paths from starting to target position is exponential. This is because the moves of a robot in each cell is non-deterministic (either right or up). By imposing priority among the rules, the non-determinism can be avoided. Now we construct the *IAPS* Π_1 with seven membranes which generate a collision free path (in case of obstacles present) in polynomial time as follows:

Let an *IAPS* $\Pi_1 = \Big(\{A, B_3, B_4, B_5, B_6, D, E, \star, +, 0, 1, d, \$\}, \{\star, +, 0, 1, d, \$\}, \#, [_1[_2[_3]_3[_4]_4[_5]_5[_6]_6[_7]_7]_2]_1, I_1, I_2, I_3, I_4, I_5, I_6, I_7, R_1, R_2, R_3, R_4, R_5, R_6, R_7, 7\Big)$, where $I_1 = \{\emptyset\}$, $I_2 = \{D\}, I_7 = \{\emptyset\}$ and $I_3 = I_4 = I_5 = I_6 = \{\mathscr{A}, D\}$, where $\mathscr{A}$ is an initial array and

$R_1 = \Big\{\emptyset\Big\}$

$R_2 = \Big\{1)\ E \rightarrow \delta\ > 2)\ B_3 \rightarrow \star\ , in_7 > 3)\ B_4 \rightarrow \star\ , in_7 > 4)\ B_5 \rightarrow \star\ , in_7\ >$

$5)\ B_6 \rightarrow \star\ , in_7\Big\}$

$R_3 = \Big\{6)\ \begin{matrix}\$\\A\end{matrix} \rightarrow \begin{matrix}B_3\\\star\end{matrix}\ , out\ > 7)\ A\,\$ \rightarrow \star B_3\ , out\ > 8)\ \$\,A \rightarrow B_3 \star\ , out\ >$

$9)\ \begin{matrix}A\\\$\end{matrix} \rightarrow \begin{matrix}\star\\B_3\end{matrix}\ , out\ > 10)\ \begin{matrix}d\\A\end{matrix} \rightarrow \begin{matrix}A\\\star\end{matrix}\ > 11)\ A\,d \rightarrow \star A\ >$

$12)\ d\,A \rightarrow A \star\ > 13)\ \begin{matrix}A\\d\end{matrix} \rightarrow \begin{matrix}\star\\A\end{matrix}\ > 14)\ \begin{matrix}A\\\star\end{matrix} \rightarrow \begin{matrix}0\\A\end{matrix}\ >$

$15)\ \star A \rightarrow A\,0\ > 16)\ A \star \rightarrow \star B\ > 17)\ \begin{matrix}\star\\A\end{matrix} \rightarrow \begin{matrix}A\\0\end{matrix}\ >$

$18)\ D \rightarrow E\ , out\Big\}$

$R_4 = \Big\{19)\ \begin{matrix}\$\\A\end{matrix} \rightarrow \begin{matrix}B_4\\\star\end{matrix}\ , out\ > 20)\ \$\,A \rightarrow B_4 \star\ , out\ > 21)\ A\,\$ \rightarrow \star B_4\ , out\ >$

$22)\ \begin{matrix}A\\\$\end{matrix} \rightarrow \begin{matrix}\star\\B_4\end{matrix}\ , out\ > 23)\ \begin{matrix}d\\A\end{matrix} \rightarrow \begin{matrix}A\\\star\end{matrix}\ > 24)\ d\,A \rightarrow A \star\ >$

$$
\begin{aligned}
&25)\ A\,d \rightarrow \star A \quad > \quad 26)\ \begin{matrix} A \\ d \end{matrix} \rightarrow \begin{matrix} \star \\ A \end{matrix} \quad > \quad 27)\ \begin{matrix} A \\ \star \end{matrix} \rightarrow \begin{matrix} 0 \\ A \end{matrix} \quad > \\
&28)\ A \star \rightarrow \star B \quad > \quad 29)\ \star A \rightarrow A\,0 \quad > \quad 30)\ \begin{matrix} \star \\ A \end{matrix} \rightarrow \begin{matrix} A \\ 0 \end{matrix} \quad > \\
&31)\ D \rightarrow E\ , out \Big\}
\end{aligned}
$$

$$
R_5 = \Bigg\{
\begin{aligned}
&32)\ A\,\$ \rightarrow \star B_5\ , out \quad > \quad 33)\ \begin{matrix} A \\ \$ \end{matrix} \rightarrow \begin{matrix} \star \\ B_5 \end{matrix} , out \quad > \quad 34)\ \begin{matrix} \$ \\ A \end{matrix} \rightarrow \begin{matrix} B_5 \\ \star \end{matrix} , out \quad > \\
&35)\ \$\,A \rightarrow B_5 \star\ , out \quad > \quad 36)\ A\,d \rightarrow \star A \quad > \quad 37)\ \begin{matrix} A \\ d \end{matrix} \rightarrow \begin{matrix} \star \\ A \end{matrix} \quad > \\
&38)\ \begin{matrix} d \\ A \end{matrix} \rightarrow \begin{matrix} A \\ \star \end{matrix} \quad > \quad 39)\ d\,A \rightarrow A \star \quad > \quad 40)\ A \star \rightarrow \star B \quad > \\
&41)\ \begin{matrix} A \\ \star \end{matrix} \rightarrow \begin{matrix} 0 \\ A \end{matrix} \quad > \quad 42)\ \begin{matrix} \star \\ A \end{matrix} \rightarrow \begin{matrix} A \\ 0 \end{matrix} \quad > \quad 43)\ \star A \rightarrow A\,0 \quad > \\
&44)\ D \rightarrow E\ , out \Big\}
\end{aligned}
$$

$$
R_6 = \Bigg\{
\begin{aligned}
&45)\ A\,\$ \rightarrow \star B_6\ , out \quad > \quad 46)\ \begin{matrix} \$ \\ A \end{matrix} \rightarrow \begin{matrix} B_6 \\ \star \end{matrix} , out \quad > \quad 47)\ \begin{matrix} A \\ \$ \end{matrix} \rightarrow \begin{matrix} \star \\ B_6 \end{matrix} , out \quad > \\
&48)\ \$\,A \rightarrow B_6 \star\ , out \quad > \quad 49)\ A\,d \rightarrow \star A \quad > \quad 50)\ \begin{matrix} d \\ A \end{matrix} \rightarrow \begin{matrix} A \\ \star \end{matrix} \quad > \\
&51)\ \begin{matrix} A \\ d \end{matrix} \rightarrow \begin{matrix} \star \\ A \end{matrix} \quad > \quad 52)\ d\,A \rightarrow A \star \quad > \quad 53)\ A \star \rightarrow \star B \quad > \\
&54)\ \begin{matrix} \star \\ A \end{matrix} \rightarrow \begin{matrix} A \\ 0 \end{matrix} \quad > \quad 55)\ \begin{matrix} A \\ \star \end{matrix} \rightarrow \begin{matrix} 0 \\ A \end{matrix} \quad > \quad 56)\ \star A \rightarrow A\,0 \quad > \\
&57)\ D \rightarrow E\ , out \Big\}
\end{aligned}
$$

$$R_7 = \Big\{\emptyset\Big\}.$$

We classify the rules of the $IAPS\ \Pi_1$ as follows:

1. **Ejecting Rule:** The rules 6-9, 19-22, 32-35 and 45-48 in the regions of membranes 3, 4, 5 and 6 respectively are known as ejecting rules. The application of one of the rules in the respective membranes will expel the array present in that membrane to its upper neighbouring membrane.

2. **Moving Rule:** The rules 10-13, 23-26, 36-39 and 49-52 in the regions of membranes 3, 4, 5 and 6 respectively are known as moving rules. These rules are used to simulate the movement of the robot from one cell of the grid to its neighbouring cell towards the target position.

3. **Backtracking Rule:** The rules 14-17, 27-30, 40-43 and 53-56 in the regions of

membranes 3, 4, 5 and 6 respectively are known as backtracking rules. If at stagnation arise during the motion of a robot, then it has to backtrack from its stagnated position. In this situation the simulation is done by backtracking rules.

4. **Switching Rule:** The rule 18 in region 3, rule 31 in region 4, rule 44 in region 5 and rule 57 in region 6 are known as switching rules. These rules are used to identify the expelling array from the membranes 3-6.

5. **Terminating Rule:** The rules 2, 3, 4 and 5 in region 2 are known as terminating rules. Application of one of these rules sends the unique output array to the output membrane 7.

6. **Dissolution Rule:** The rule 1 in region 2 is known as dissolution rule.

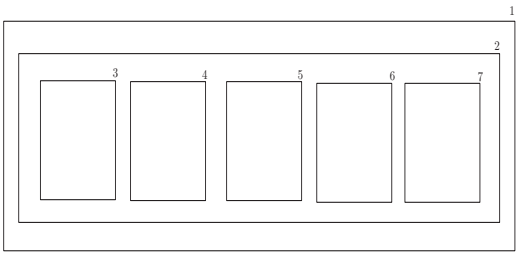
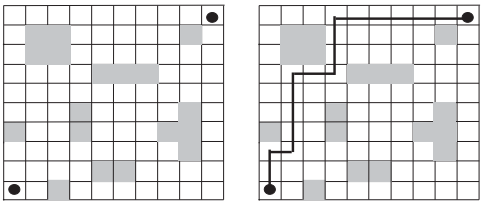


Figure 5: The membrane structure of $IAPS\ \Pi_1$

The working of $IAPS\ \Pi_1$ is as follows: The membrane structure of Π_1 is shown in Figure 5. In the initial configuration, the regions 3, 4, 5 and 6 contains an identical copy of an initial array $\mathscr{A}$ (corresponding to the initial scene) and a symbol D. The regions 1, 2 and 7 do not contain any object. The computing on the initial array begins parallelly in regions 3-6. The rules applied in this situation will be moving rules and backtracking rules. If an ejecting rule becomes applicable in regions 3-6, the evolved arrays is shifted to region 2 with symbol B_i $(3 \leq i \leq 6)$ marked at the target position. In the next step of evolution, while applying termination rules 2-5 on B_i, in region 2, atleast one of the regions 3-6, works on the symbol D by the distinct switching rules present in them respectively. Finally, a unique array indicating the desired shortest path is sent to the output membrane 7. Now in region 2, the presence of symbol E in the array obtained by the previous rewriting step, dissolves membrane 2 by rule 1. On the other hand, the evolution proceeds in the rest of the membranes until a halting configuration is reached. Finally, the unique output array specifying the shortest path of the robot is retained in the output membrane 7.

A typical representation of the initial scene is indicated in Figure 3. The targeted scene corresponding to the array present in the output membrane 7 is given in Figure 4c. The complexity will be $2(n-1)$ parallel steps, where n is the size of the array.



(a) a) Initial state (b) b) Path from source to target

Figure 6: Obstacles distributed everywhere in the grid

3.2 Robot motion on a scene with rectangular obstacles distributed everywhere

In this case the rectangular obstacles are distributed everywhere in the grid as shown in Figure 6a. The corresponding array is modeled as per description in section 3.1 (see Figure 7). The same $IAPS\ \Pi_1$ works on prioritized rules as follows. A copy of the initial array $\mathscr{A}$ present in each of the membranes 3-6 is given in Figure 7.

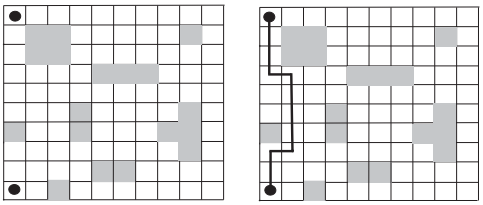
1	1	1	1	1	1	1	1	1	1	1	1
1	d	d	d	d	d	d	d	d	d	\$	1
1	d	+	+	d	d	d	d	d	+	d	1
1	d	+	+	d	d	d	d	d	d	d	1
1	d	d	d	d	+	+	+	d	d	d	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	+	d	d	d	d	+	d	1
1	+	d	d	+	d	d	d	+	+	d	1
1	d	d	d	d	d	d	d	d	+	d	1
1	d	d	d	d	+	+	d	d	d	d	1
1	A	d	+	d	d	d	d	d	d	d	1
1	1	1	1	1	1	1	1	1	1	1	1

Figure 7: Initial array representation of Π_1 in regions 3-6

The evolution of an $IAPS\ \Pi_1$ is exactly as before. The array present in the output membrane 7 indicates the shortest path. Some typical scenes are given in Figures 6a and 8a. This P system will identify the shortest path even in the case of different target positions (6b, 8b). However, the source is fixed. Since the grid has n^2 cells, the collision free path (irrespective of target position) requires less than n^2 moves. In membranes 3-6, the priority rules are modeled depending on the target position of the robot and the distribution of obstacles.

4 Robot motion on a scene with polygonal obstacles

In this case we consider the polygonal obstacles distributed everywhere in the grid. It is indicated by blue color and is shown in Figure 10a. We allowed the polygonal obstacles not only adjacent to the boundary of the grid, but also anywhere in the grid. The scene is modeled as an array as per description in section 3.1. Since the obstacles are polygons, the portion



(a) a) Initial state (b) b) Path from source to target

Figure 8: Obstacles distributed everywhere in the grid with different targets

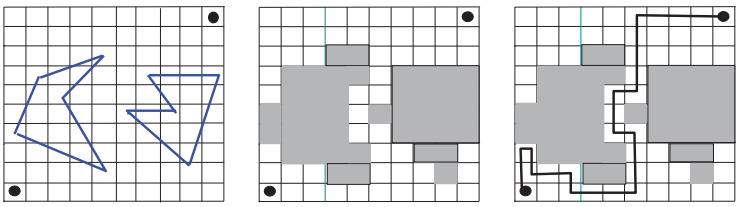
of a polygon present in any cell of the grid will be denoted by a $+$ symbol. A typical array is shown in Figure 9. A copy of the initial array $\mathscr{A}$ present in each of the membranes 3-6 is given in Figure 9.

1	1	1	1	1	1	1	1	1	1	1	1
1	d	d	d	d	d	d	d	d	d	\$	1
1	d	d	d	d	d	d	d	d	d	d	1
1	d	d	d	$+$	$+$	d	d	d	d	d	1
1	d	$+$	$+$	$+$	$+$	d	$+$	$+$	$+$	$+$	1
1	d	$+$	$+$	$+$	d	d	$+$	$+$	$+$	$+$	1
1	$+$	$+$	$+$	$+$	d	$+$	$+$	$+$	$+$	$+$	1
1	$+$	$+$	$+$	$+$	d	d	$+$	$+$	$+$	$+$	1
1	d	$+$	$+$	$+$	$+$	d	d	$+$	$+$	d	1
1	d	d	d	$+$	$+$	d	d	d	$+$	d	1
1	A	d	d	d	d	d	d	d	d	d	1
1	1	1	1	1	1	1	1	1	1	1	1

Figure 9: Initial array representation of Π_1

The $IAPS$ Π_1 generates the collision free path from the starting to the target positions (Figure 10c). Since the grid contains n^2 cells, the number of moves to generate the collision free path requires less than n^2 moves. Hence, in this case as well, the collision free path can be generated in polynomial time (parallel steps).

Remark 4.1. In all the above three cases, we allowed the target position of the robot in the grid to vary, but we assumed that the starting position of the robot is always fixed. That is the bottom left position of the grid. Based on this assumption we have constructed the priority



(a) a) Grid with polygonal obstacles (b) b) Initial state (c) c) Path from source to target

Figure 10: Robot motion with Polygonal Obstacles

rules in the membranes 3-6 of the $IAPS\ \Pi_1$. If we intend to vary the starting position of the robot, then the other possibilities of prioritized rules with suitable number of membranes must be added in $IAPS\ \Pi_1$

5 Conclusion

A grammatical model that uses Isotonic arrays has been proposed to obtain a path (with or without) obstacles in polynomial time. The assumed path is originating from starting position to the identified target position also shown to be the shortest such path. One of the famous interesting study would be to extend the interest of motion planning in the grid where the obstacles are mobile. Further investigation will also include the case when the environment is totally or partially unknown.

References

Cook, C. R. and Wang, P. S. P. 1978. A chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing* **8**(1): 144 – 152.

Păun, G. 2000. Computing with membranes, *J. Comput. System Sci.* **61**(1): 108–143.

Păun, G., Rozenberg, G. and Salomaa, A. (eds) 2010. *The Oxford Handbook of Membrane Computing*, Oxford University Press.

Rosenfeld, A. 1971. Isotonic grammars, parallel grammars, and picture grammars, *Machine Intelligence, 6*, American Elsevier, New York, pp. 281–294.

Rozenberg, G. and Salomaa, A. (eds) 1997. *Handbook of formal languages. Vol. 1*, Springer-Verlag, Berlin. Word, language, grammar.

Saudi, A. and Sulaiman, J. 2013. Indoor path planning for mobile robot using lbbc-eg, *International journal of imaging and robotics* **11**(3): 37–45.

Soriano, Á., Bernabeu, E. J., Valera, Á. and Vallés, M. 2014. Distributed collision avoidance method based on consensus among mobile robotic agents, *International Journal of Imaging and Robotics* **15**(1): 80–90.

Sureshkumar, W. and Rama, R. 2016a. 4-directional combinatorial motion planning via labeled isotonic array p system, *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015*, Springer, pp. 701–709.

Sureshkumar, W. and Rama, R. 2016b. Chomsky hierarchy control on isotonic array p systems, *International Journal of Pattern Recognition and Artificial Intelligence* **30**(02): 1650004.