Test-Driven Simulation of Robots Controlled by Enzymatic Numerical P Systems Models

Radu Traian Bobe[1(✉)], Marian Gheorghe[2], Florentin Ipate[1], and Ionuţ Mihai Niculescu[1]

[1] Department of Computer Science, Faculty of Mathematics and Computer Science, University of Bucharest, Str Academiei 14, 010014 Bucharest, Romania
radu.bobe@s.unibuc.ro, florentin.ipate@unibuc.ro
[2] Faculty of Engineering and Informatics, University of Bradford, Bradford, West Yorkshire BD7 1DP, UK
m.gheorghe@bradford.ac.uk
https://www.ifsoft.ro/~florentin.ipate/,
https://www.bradford.ac.uk/staff/mgheorghe,
https://ionutmihainiculescu.ro/

Abstract. The simulation of robots behavior and the use of robust models are very important for building controllers. Testing is an important aspect in this process. In this paper, a test-driven approach for designing robot controllers based on enzymatic numerical P systems models is introduced. Four such models are defined and tested using three distinct scenarios. The paper reveals an effective way of using modelling, simulation and testing in a coherent way.

Keywords: Membrane computing · Numerical P systems · Enzymatic numerical P systems · Robot controllers · Simulation · Search-based software testing

1 Introduction

Membrane computing is a branch of a more general research area, called *natural computing*. Natural computing investigates computational models inspired by phenomena and processes occurring in nature. Membrane computing, introduced by Gh. Păun [13], is a biological-inspired computational paradigm, investigating models that abstract out from the structure and functionality of the living cells. The field evolved rapidly in its first decade, focusing on theoretical developments of several classes of membrane systems (or P systems), various applications in computer science, graphics, economics and biology, as well as on relationships with other classes of computational models. A handbook reporting the key developments in membrane computing at the end of the first decade of research in the field has been published [15]. The main classes of membrane systems (P systems), *cell-like*, *tissue-like* and *neural-like* P systems, reflect the structure

J.-L. Guisado-Lizar et al. (Eds.): SIMUtools 2023, LNICST 519, pp. 56–69, 2024.
https://doi.org/10.1007/978-3-031-57523-5_5

of living cell, tissue and brain, respectively. Real-life applications of membrane computing in various areas have been investigated in [24].

A new class of membrane systems, namely *numerical P systems*, has been introduced, with the aim of modelling economics phenomena, in a nature-inspired computational setting [14]. Later on an extension of this model has been considered, *enzymatic numerical P system*, which looks appropriate for modelling robot controllers [11]. Some robot controllers have been designed based on enzymatic numerical P systems [12,22]. A simulator, called Pep [17], for running enzymatic numerical P systems has been provided and utilized in various controllers [3,12,22].

Software applications tend to have a considerable role in solving problems in various aspects of the life. Given the importance of these applications, it is important to ensure the product quality and functionality and this is mostly addressed through software testing. This becomes an important part of the software development life-cycle, aiming to validate that the requirements of the software product are fulfilled, by identifying undesired behavior. A largely used testing approach is search-based testing [8]. A relatively new tool for search-based testing, especially for autonomous systems, is AmbieGen [6].

In this paper, we propose an approach for building simulators for a robot controller designed to handle enzymatic numerical P system models. Four models are built using the formalism provided in [11]. The development process of designing the models is driven by a number of scenarios, one of these, the most complex, uses the testing tool AmbieGen for generating use cases and for testing the model-based systems. In some sense, our approach is similar to *"test-driven development"* [5], improvements being added only after studying the performance of the models during the previous tests. The behavior of the controller and the enzymatic numerical P system models designed for an *E-puck* educational robot [10], is visualized with Webots, a robot simulation tool.

The paper is structured as follows: Sect. 2 presents the definition of the enzymatic numerical P system model used in the paper. Section 3 introduces the working environment, including the tools used. Section 4 describes the four enzymatic numerical P system models, while Sect. 5 illustrates the testing approach along with the results for three scenarios. Finally, Sect. 6 presents the conclusions and future work.

2 Enzymatic Numerical P System Definition

The enzymatic numerical P systems are special classes of membrane systems, that share with the rest of the models only the membrane structure, in the form of a tree. The compartments contain *variables* instead of objects and their values are processed by *programs* replacing the rewriting and communication rules [11]. As in any membrane system, the compartments are delimited by membranes. Subsequently, we use them interchangeably. A global clock controls the systems through discrete time units.

The enzymatic numerical P system (EN P system) is defined by the tuple:

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0))) \quad (1)$$

where:

- $m \geq 1$ is degree of the system Π (the number of membranes);
- H is an alphabet of labels;
- μ is membrane structure (a tree);
- Var_i is a set of variables from membrane $i, 1 \leq i \leq m$;
- $Var_i(0)$ is the initial values of the variables from region $i, 1 \leq i \leq m$;
- Pr_i is the set of programs from membrane $i, 1 \leq i \leq m$.

The program $Pr_{l_i,i}$, $1 \leq l_i \leq m_i$ has one of the following forms:

i) non-enzymatic

$$F_{l_i,i}(x_{1,i}, \ldots, x_{k,i}) \rightarrow c_{1,i}|v_1 + c_{2,i}|v_2 + \cdots + c_{m_i,i}|v_{m_i}$$

where $F_{l_i,i}(x_{1,i}, \ldots, x_{k,i})$ is the production function, $c_{1,i}|v_1 + c_{2,i}|v_2 + \cdots + c_{m_i,i}|v_{m_i}$ is the repartition protocol, and $x_{1,i}, \ldots, x_{k,i}$ are variables from Var_i. Variables $v_1, v_2 \ldots v_{m_i}$ belong to the compartment where the programs are located, and to its upper and inner compartments, for a particular compartment i. If a compartment contains more than one program, only one will be non-deterministically chosen.

ii) enzymatic

$$F_{l_i,i}(x_{1,i}, \ldots, x_{k,i})|_{e_i} \rightarrow c_{1,i}|v_1 + c_{2,i}|v_2 + \cdots + c_{m_i,i}|v_{m_i}$$

where e_i is an enzymatic variable from Var_i, $e_i \notin \{x_{1,i}, \ldots, x_{k,i}, v_1, \ldots, v_{m_i}\}$. The program can be applied at time t only if $e_i > min(x_{1,i}(t), \ldots, x_{k,i}(t))$. The programs that meet this condition in a compartment will be applied in parallel.

When the program is applied by the system at time $t \geq 0$, the computed value

$$q_{l_i,i}(t) = \frac{F_{l_i,i}(x_{1,i}(t), \ldots, x_{k,i}(t))}{\sum_{j=1}^{n_i} c_{j,i}}$$

representing the *unitary portion* that will be distributed to the variables $v_1, \ldots, v_n$, proportional to coefficients $c_{1,i}, \ldots, c_{m_i,i}$, where $c_{j,i} \in \mathbf{N}^+$ and the received values will be $q_{l_i,i}(t) \cdot c_{1,i}, \ldots, q_{l_i,i}(t) \cdot c_{m_i,i}$.

The value of each of the variables from $t-1$, occurring in the production functions is *consumed*, reset to zero, and its new value is the sum of the proportions distributed to variable through the repartition protocols, if it appears in them or remain at value zero.

3 Research Environment and Tools

The research environment allowing to implement our simulation approach consists of several tools that serve the purpose of this investigation.

EN P system models are simulated with a tool called Pep [17] allowing to define them in domain specific language and then execute the systems in accordance with the semantics of the models. This tool allows us to implement the robot controller environment and execute it.

The controller is meant to equip an E-puck robot [10]. Important for our study is the fact that the robot has two motors attached to the body along with two wheels; the speed value is changeable and handled by the controller. It also includes eight infrared proximity sensors placed around the body and a GPS attached to the turret slot in order to assess the coordinates at each step and use them for the simulation. For the purpose of our investigation a simulator is used instead of the E-puck robot. In this case, Webots, a robot simulation tool is used, allowing to construct complex environments [9]. A special mechanism, called PROTO [16], allows to build special objects within the environment.

A third tool utilized in this investigation is AmbieGen, an open-source tool relying on evolutionary search methods for the generation of test scenarios [6,7]. The software is written in Python and uses evolutionary search [23] and multi-objective algorithms for search-based test generation [2].

We now briefly described how these tools are integrated.

PeP simulator containing EN P system model is integrated with E-puck controller.

The pseudocode version of the main loop in our controller, taken from [1], appears below.

Algorithm 1. Simulation steps performing algorithm

```
1: repeat
2:     for i=1 to number_of_sensors do
3:         sensor_membrane(i) ← value(i)
4:     run one simulation step
5:     read lw, rw  from P system
6:     leftMotor ← lw
7:     rightMotor ← rw
8: until  the end of the road or E-puck goes out of the road
```

AmbieGen provides the test generation scenarios with roads, as *.json* files, plus test outcome, maximum curvature coefficient etc.

In the Webots graphic interface, the simulation can be visualized on inputs provided by AmbieGen. More details about the tools and the way they work together are available from [1].

Remark 1. All the models used in this experiment and test scenarios along with the results are available in our GitHub project [18].

4 Enzymatic Numerical P System Models

In this section we will present our models used to control the robot. The controller receives data from proximity sensors, that measure distances to obstacles from the environment, to determine the direction of movement of a differential two wheeled robot, E-puck, in our case.

The proximity sensor has a range of 4 cm; if the obstacles are further than this limit the sensor returns the value of 0. The proximity sensors are placed on the left and right side of the robot in the direction of its movement at different angles.

The basic model was taken from [3] and adapted to make a rotation move when an obstacle is near, in order to avoid it and continue the movement. The equations that calculate the linear and angular velocity are shown below:

$$leftSpeed = cruiseSpeed + \sum_{i=1}^{n} weightLeft_i \cdot prox_i \tag{2}$$

$$rightSpeed = cruiseSpeed + \sum_{i=1}^{n} weightRight_i \cdot prox_i \tag{3}$$

The $leftSpeed$ and $rightSpeed$ are the speeds of the two wheels of the robot, whilst n is the number of sensors. Each sensor has constant weight values, empirically chosen to conduct the robot to the desired behavior.

This basic model encountered considerable difficulties to pass the road tests generated by AmbieGen, being capable to move without hitting the margins just on straight roads or very little curved roads. The model was formally described and more experiments have been done in our previous work [1], where it is referred as Π_{M_1}.

Considering the limitations of Π_{M_1}, the model we present next is an improvement on the first one.

4.1 Basic Model with Rotation

Analyzing the initial model, we observed that the speed of the wheel on the side with an obstacle increased reported to weights. Thereby, this caused a sort of rotation in the opposite direction of the obstacle but not sufficient to pass the test.

The main change of our model was to introduce the compartment w, calculating the product of the travel speed and the sum of the weights. In this way, after rotating in the opposite direction when detecting an obstacle, the robot will continue the test, moving forward with a constant velocity.

This is certainly an improvement, especially when the robot is challenged on roads, but as presented next, there were still some issues that aimed us to introduce another model.

Let us consider the following function:

$$f(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases}$$

This function will be used in the equations describing the behavior of the model and in the production functions from the programs.

The equations describing the behavior are:

$$weightLeft = \sum_{i=1}^{n} weightLeft_i \cdot prox_i \tag{4}$$

$$weightRight = \sum_{i=1}^{n} weightRight_i \cdot prox_i \tag{5}$$

$$leftSpeed = cruiseSpeed \cdot weightLeft + f(weightLeft) \cdot cruiseSpeed \tag{6}$$

$$rightSpeed = cruiseSpeed \cdot weightRight + f(weightRight) \cdot cruiseSpeed \tag{7}$$

The model is defined as follows:

$$\Pi_{M_2} = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0))) \tag{8}$$

where:

- $m = 3k + 3, k = 6$;
- $H = \{s, w, s_c\} \cup \bigcup_{i=1}^{k} \{c_i, s_i, w_i\}$;
- $\mu = [[[[]_{s_1}[]_{w_1}]_{c_1} \cdots [[]_{s_k}[]_{w_k}]_{c_k}[]_{s_c}]_w]_s$;
- $Var_s = \{x_{s_l}, x_{s_r}\}, Var_w = \{x_{w_l}, x_{w_r}, e_w\}, Var_{s_c} = \{x_{s_c}\}$,
  $Var_{c_i} = \{x_{c_i,s_l}, x_{c_i,s_r}, x_{c_i,w_l}, x_{c_i,w_r}, e_{c_i}\}, 1 \leq i \leq k$,
  $Var_{s_i} = \{x_{s_i,i}\}, 1 \leq i \leq k$,
  $Var_{w_i} = \{x_{w_i,w_l}, x_{w_i,w_r}, e_{w_i}\}, 1 \leq i \leq k$;
- $Var_i(0) = 0, 1 \leq i \leq m$;
- $Pr_s = \{0 \cdot x_{s_l} \cdot x_{s_r} \to 1 | x_{s_l} + 1 | x_{s_r}\}$;
  $Pr_w = \{x_{s_c} \cdot x_{w_l} + f(x_{w_l}) \cdot x_{s_c}|_{e_w} \to 1 | x_{w_l},$
  $x_{s_c} \cdot x_{w_r} + f(x_{w_r}) \cdot x_{s_c}|_{e_w} \to 1 | x_{w_r}\}$;
  $Pr_{s_c} = \{x_{s_c} \to 1 | x_{s_c}\}$;
  $Pr_{c_i} = \{x_{c_i,s_l} \cdot x_{c_i,w_l}|_{e_{c_i}} \to 1 | x_{s_l},$
  $x_{c_i,s_r} \cdot x_{c_i,w_r}|_{e_{c_i}} \to 1 | x_{s_r}\}, 1 \leq i \leq k$;
  $Pr_{s_i} = \{3x_{s_i,i} \to 1 | x_{s_i,i} + 1 | x_{c_i,s_l} + 1 | x_{c_i,s_r}\}, 1 \leq i \leq k$;
  $Pr_{w_i} = \{2x_{w_i,w_l}|_{e_{w_i}} \to 1 | x_{w_i,w_l} + 1 | x_{c_i,w_l},$
  $2x_{w_i,w_r}|_{e_{w_i}} \to 1 | x_{w_i,w_r} + 1 | x_{c_i,w_r}\}, 1 \leq i \leq k$;

The meaning of the variables from the model is the following:

- x_{s_l} and x_{s_r} from the region s represent $leftSpeed$ and $rightSpeed$, the sum of the products are accumulated in s;

∘ x_{s_c} from the compartment s_c is $cruiseSpeed$;
∘ each pair of weights, $weightLeft_i$ and $weightRight_i$, resides in the regions w_i, $1 \leq i \leq k$;
∘ for each proximity sensor, $prox_i$, a compartment is defined, namely s_i, containing a single variable, $x_{s_i,i}$, $1 \leq i \leq k$;
∘ the products are calculated by two distinct programs, $weightLeft_i \cdot prox_i$, and $weightRight_i \cdot prox_i$, $1 \leq i \leq k$, in the compartments c_i.

4.2 Refined Model

During the test phase of the above presented model, we observed that even though the robot avoids the obstacles (road borders, in case of generating roads with AmbieGen), it tends to have a "zig-zag" motion going from the proximity of a border to the proximity of the other one. Considering this, an immediate adjustment was to recenter the robot after avoiding an obstacle and reaching the center of the road, so the robot will go straight until it encounters a new obstacle.

We made this adjustment by introducing a new membrane, called *Direction*. The membrane has seven variables, called *directionLeft, directionRight, angle, state, distance, angleStep, distanceStep*, which will be detailed when giving the formal definition of the model. In order to obtain the desired behavior, we used differential drive kinematics equations [20]. The *state* variable aims to reproduce a finite state machine inside the production function of the membrane, with the following states:

a) state 0 - the robot is moving in a straight line
b) state 1 - the robot is moving in the presence of an obstacle
c) state 2 - the robot is moving to approximately the center of the lane
d) state 3 - the robot is recentering on the lane

Before introducing the mathematical definition of the described model, we firstly defined four functions needed in the production functions (Table 1):

Table 1. Functions used in the model

sgn(x)	$sgn(x) = \begin{cases} 1, & \text{if } x = 1 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x = -1 \end{cases}$
not(x,y)	$not(x,y) = \begin{cases} 1, & \text{if } x \neq y \\ 0, & \text{otherwise} \end{cases}$
gt(x,y)	$gt(x,y) = \begin{cases} 1, & \text{if } x > y \\ 0, & \text{otherwise} \end{cases}$
eq(x,y)	$eq(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases}$

Also, we used the constant *len* which represents the axle length of the robot. For E-puck this is equal to 52 mm.

The model is defined as follows:

$$\Pi_{M_3} = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0))) \quad (9)$$

where:

- $m = 3k + 4, k = 6$;
- $H = \{s, d, w, s_c\} \cup \bigcup_{i=1}^{k} \{c_i, s_i, w_i\}$;
- $\mu = [[[[[]_{s_1}[]_{w_1}]_{c_1} \cdots [[]_{s_k}[]_{w_k}]_{c_k}[]_{s_c}]_w]_d]_s$;
- $Var_s = \{x_{s_l}, x_{s_r}\}, Var_d = \{x_{d_l}, x_{d_r}, x_a, x_{st}, x_{dst}, x_{as}, x_{ds}, e_{ds}, e_{dw}, e_d\}$,
  $Var_w = \{x_{w_l}, x_{w_r}, e_w\}, Var_{s_c} = \{x_{s_c}\}$,
  $Var_{c_i} = \{x_{c_i,s_l}, x_{c_i,s_r}, x_{c_i,w_l}, x_{c_i,w_r}, e_{c_i}\}, 1 \leq i \leq k$,
  $Var_{s_i} = \{x_{s_i,i}\}, 1 \leq i \leq k$,
  $Var_{w_i} = \{x_{w_i,w_l}, x_{w_i,w_r}, e_{w_i}\}, 1 \leq i \leq k$;
- $Var_i(0) = 0, i \in \{s, w, s_c\} \cup \bigcup_{i=1}^{k} \{c_i, s_i, w_i\},\ Var_{d_{e_d}}(0) = 100, Var_{d_i}(0) = 0,$
  $i \in Var_d \setminus \{e_d\}$;
- $Pr_s = \{0 \cdot x_{s_l} \cdot x_{s_r} \to 1|x_{s_l} + 1|x_{s_r}\}$;
- $Pr_d = \{not(x_{st}, 3) \cdot x_{d_l} + eq(x_{st}, 3) \cdot ((x_{d_l} + |x_{as}|) \cdot sgn(x_{as}))|_{e_{ds}} \to 1|x_{s_l},$
  $not(x_{st}, 3) \cdot x_{d_r} + eq(x_{st}, 3) \cdot ((x_{d_r} + |x_{as}|) \cdot sgn(-x_{as}))|_{e_{ds}} \to 1|x_{s_r},$
  $gt(x_{as}, 0) \cdot x_{as}|_{e_{ds}} \to 1|x_{as},$
  $x_{d_l}|_{e_{dw}} \to 1|x_{s_l},$
  $x_{d_r}|_{e_{dw}} \to 1|x_{s_r},$
  $e_{ds} \cdot e_{dw} \cdot 0|_{e_d} \to 1|e_{ds} + 1|ed_w,$
  $not(x_{d_r} - x_{d_l}, 0) \cdot \frac{x_{d_r} - x_{d_l}}{2len} + eq(x_{d_r} - x_{d_l}, 0) \cdot x_a + x_a \cdot 0|_{e_{dw}} \to 1|x_a,$
  $not(x_{d_r} - x_{d_l}, 0) \cdot \frac{x_{d_r} - x_{d_l}}{4} + eq(x_{d_r} - x_{d_l}, 0) \cdot x_{as}|_{e_{dw}} \to 1|x_{as},$
  $not(x_{d_r} - x_{d_l}, 0) \cdot \lfloor |\frac{4}{2len}| \rfloor \cdot 3 + eq(x_{d_r} - x_{d_l}, 0) \cdot x_{ds} + x_{ds} \cdot 0|_{e_{dw}} \to 1|x_{ds},$
  $eq(x_{st}, 1) \cdot 2 + eq(x_{st}, 2) \cdot not(x_{dst}, 0) \cdot 2 + eq(x_{st}, 2) \cdot eq(x_{dst}, 0) \cdot 3 +$
  $eq(x_{st}, 3) \cdot gt(x_{dst}, 0) \cdot 3 + eq(x_{st}, 3) \cdot eq(x_{dst}, 0) \cdot 0|_{e_{ds}} \to 1|x_{st},$
  $x_{st} \cdot 0 + 1|_{e_{dw}} \to 1|x_{st},$
  $x_{dst} \cdot 0 + 180|_{e_{dw}} \to 1|x_{dst},$
  $eq(x_{st}, 2) \cdot gt(x_{dst}, 0) \cdot (x_{dst} - 1) + eq(x_{st}, 1) \cdot x_{dst}|_{e_{ds}} \to 1|x_{dst},$
  $eq(x_{st}, 3) \cdot gt(x_{ds}, 0) \cdot (x_{ds} - 1) + not(x_{st}, 3) \cdot x_{ds}|_{e_{ds}} \to 1|x_{ds}$;
  $Pr_w = \{x_{s_c} \cdot x_{w_l}|_{e_w} \to 1|x_{d_l},$
  $f(x_{w_l}) \cdot x_{s_c}|_{e_w} \to 1|x_{d_l}\},$
  $x_{s_c} \cdot x_{w_r}|_{e_w} \to 1|x_{d_r},$
  $f(x_{w_r}) \cdot x_{s_c}|_{e_w} \to 1|x_{d_r}\},$
  $(f(x_{w_l}) \cdot f(x_{w_r})) \cdot x_{s_c} \cdot 100|_{e_w} \to 1|e_{ds},$
  $(not(x_{w_l}, 0) \cdot not(x_{w_r}, 0)) \cdot x_{s_c} \cdot 100|_{e_w} \to 1|e_{dw}$;
  $Pr_{s_c} = \{x_{s_c} \to 1|x_{s_c}\}$;
  $Pr_{c_i} = \{x_{c_i,s_l} \cdot x_{c_i,w_l}|_{e_{c_i}} \to 1|x_{s_l},$
  $x_{c_i,s_r} \cdot x_{c_i,w_r}|_{e_{c_i}} \to 1|x_{s_r}\}, 1 \leq i \leq k;$

$$Pr_{s_i} = \{3x_{s_i,i} \to 1|x_{s_i,i} + 1|x_{c_i,s_l} + 1|x_{c_i,s_r}\}, 1 \leq i \leq k;$$
$$Pr_{w_i} = \{2x_{w_i,w_l}|_{e_{w_i}} \to 1|x_{w_i,w_l} + 1|x_{c_i,w_l},$$
$$2x_{w_i,w_r}|_{e_{w_i}} \to 1|x_{w_i,w_r} + 1|x_{c_i,w_r}\}, 1 \leq i \leq k;$$

As observed, the main difference in the membranes structure is made by the new region called *direction*. Next we present the meaning of the variables used in it:

- x_{d_l} and x_{d_r} represent $directionLeft$ and $directionRight$;
- x_a and x_{as} represent $angle$ and $angleStep$;
- x_{dst} and x_{ds} represent $distance$ and $distanceStep$;
- x_{st} represents the $state$ of the simulated finite state machine

4.3 Extended Refined Model

Experiments carried out with the second model proved that when the robot approaches perpendicularly the obstacle, it remains locked into that obstacle. This limitation is caused by the values of weights, which have values of opposite sign, and sum is cancel each other out. This situation can be easily explained by the position of the proximity sensors that are facing the obstacle perpendicularly.

To solve this problematic behavior we defined two more production functions inside *Weight* compartment. These functions are distributed to *directionRight* and *directionLeft* variables, as follows:

$$\begin{aligned} directionLeft = {} & eq(|weightLeft|, |weightRight|) \cdot gt(|weightLeft|, 0) \cdot \\ & \cdot gt(|weightRight|, 0) \cdot weightLeft \cdot cruiseSpeed \cdot 0 \\ directionRight = {} & eq(|weightLeft|, |weightRight|) \cdot gt(|weightLeft|, 0) \cdot \\ & \cdot gt(|weightRight|, 0) \cdot weightRight \cdot 0 + cruiseSpeed \end{aligned}$$

In this way, we ensure that the robot will not get stuck when perpendicularly facing an obstacle, as the speed of the right wheel (guided by *directionRight* variable) will move the robot to the left.

An interesting step would be to automatically decide what direction to follow in this situation (e.g., if an obstacle is near on the left, a better decision should be to activate the *directionLeft* variable, thus moving the robot to the right).

5 Simulation Results

In this section we will present some scenarios designed to challenge the robot as well as their results. Considering the way we defined and integrated the models, these simulation scenarios guided us to refine intermediate variants of models based on the results.

In this experiment we opted for three scenarios and all of them are defined by the type of the area E-puck needs to go through. These are introduced as follows:

a) *corridors*
b) *a square*
c) *roads generated by AmbieGen*

The first two scenarios were created in a simple manner, using the areas that E-puck should cover being defined using Webots embedded shapes. The last scenario is the most complex one, taking the roads generated by AmbieGen and integrating them in a Webots world.

As stated above, our approach combines robot testing and simulation, using AmbieGen for road generation. The tool offers flexibility in choosing the parameters needed by the genetic algorithm. Some of them are set as constant values in the source code and other aspects (e.g., allowed out of bound percentage, map size, generation time) can be set easily from the command line, when launching the tool. Taking into consideration the default values from AmbieGen, we performed some trials and noticed that the same would fit our needs in road generation. More details about these values and how to use the tool were presented in [1,4].

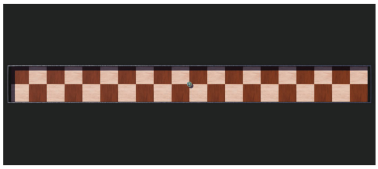
As mentioned before, this approach encapsulates three main types of scenarios, *a corridor, a square* and *roads generated by AmbieGen*. For *corridor* and *square* we simulated two situations. In the first situation, the robot starts in a straight line from the middle of the corridor or from the middle of the square. For simplicity, let us call this types of tests *corridor straight* and *square straight*.

The other situation assumes that the robot also starts from the middle of the corridor, respectively square, but with an angle of 15°. Let us refer to these test types as *corridor angle* and *square angle*.

Table 2 contains the simulation results for each model, starting with the basic one, Π_{M_1}. It includes the test types presented in the above paragraph and four roads generated by AmbieGen and imported to Webots. We chose these roads from a larger suite, opting for different curvatures in order to better observe the behavior of each model.

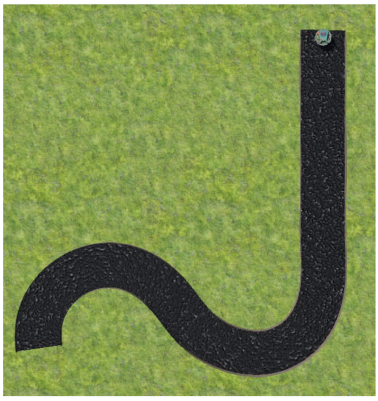
Table 2. Experimental results

Test type	Π_{M_1}	Π_{M_2}	Π_{M_3}	Π_{M_4}
Corridor straight	Failed	Failed	Failed	Passed
Corridor angle	Failed	Passed	Failed	Passed
Square straight	Failed	Failed	Passed	Passed
Square angle	Failed	Passed	Passed	Passed
Road 1	Failed	Passed	Passed	Failed
Road 2	Failed	Passed	Passed	Passed
Road 3	Failed	Passed	Passed	Failed
Road 4	Passed	Passed	Passed	Failed

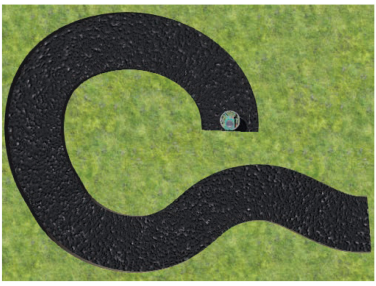


Corridor

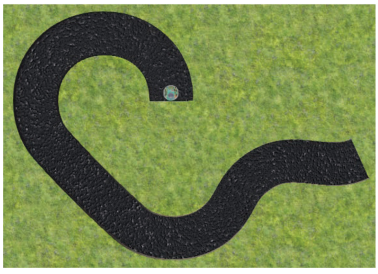
Square



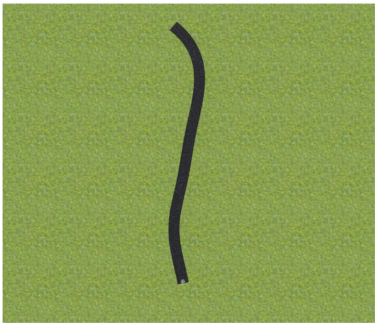
Road 1



Road 2



Road 3



Road 4

Fig. 1. Test cases as represented in Webots

When analyzing the experimental results, we noticed that the basic model, Π_{M_1}, has the worst performance compared to the other three models. This basic model passed just one test, Road 4, which is the simplest one (as presented in Fig. 1).

Another observation that can be extracted from the experimental results recorded in Table 2 is that the second model, Π_{M_2}, has an improved performance in terms of passing the road tests. The improvement consists in a rotation that is performed when the road is curved (i.e., the proximity sensors detect an obstacle). Additional road tests were added to [19].

The refocusing movement added to the behavior of the third model, Π_{M_3}, conducted to a natural movement on the roads, all the roads being also passed for this model. Nevertheless, it can be observed that this modification of the membrane structure (with the addition of the membrane *direction*) made Π_{M_3} to fail the *corridor angle* challenge, but pass the *square straight*, failed by Π_{M_2}.. In the end, the last model, Π_{M_4}, came with an improvement in passing the corridor and square tests, due to the property of moving even if the obstacle is in the front of the robot (placed perpendicularly), the situation when the robot gets stuck in front of an obstacle being handled. However, the adjustments made came with a few disadvantages in passing road tests, the robot moving to the left and failing the test at the moment the robot direction is perpendicular to one of the borders. In many instances, this scenario was frequently observed, serving as the underlying cause for the model's failure to pass three out of the four road test examples.

Figure 1 illustrates the experimental tests discussed above. A video representation of each model performance on these tests can be found at [21].

6 Conclusions and Future Work

This paper illustrated our approach to model, simulate and test a robot controller based on enzymatic numerical P systems. As we already introduced two models and the working environment in our previous work [1], we formally described each additional model and the reasons which made us to implement them.

As presented above, each model was constructed taking into consideration the limitations discovered during the testing phase of the previous ones. We also emphasize the capabilities of each model in a comparative manner, testing them in three challenging scenarios with different features.

Concerning our future work, we intend to involve in our experiments other types of P systems in order to obtain a more natural functioning of the robot. We analyze the possibility to dynamically assign weights values depending on the situation (e.g., taking a decision when the robot is close to an obstacle and there is another obstacle nearby). The palette of different P systems types is encouraging but there are some limitations in the area of available tools for simulating them, this constituting another aspect that we investigate.

References

1. Bobe, R.T., Ipate, F., Niculescu, I.M.: Modelling and search-based testing of robot controllers using enzymatic numerical p systems. In: Cheval, H., Leuştean, L., Sipoş, A. (eds.) Proceedings 7th Symposium on Working Formal Methods, Bucharest, Romania, 21–22 September 2023. Electronic Proceedings in Theoretical Computer Science, vol. 389, pp. 1–10. Open Publishing Association (2023). https://doi.org/10.4204/EPTCS.389.1
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
3. Florea, A.G., Buiu, C.: Modelling multi-robot interactions using a generic controller based on numerical P systems and ROS. In: 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), pp. 1–6 (2017). https://doi.org/10.1109/ECAI.2017.8166411
4. Gambi, A., Jahangirova, G., Riccio, V., Zampetti, F.: SBST tool competition 2022. In: Proceedings of the 15th Workshop on Search-Based Software Testing, pp. 25–32 (2022). https://doi.org/10.1145/3526072.3527538
5. George, B., Williams, L.: A structured experiment of test-driven development. Inf. Softw. Technol. **46**(5), 337–342 (2004). https://doi.org/10.1016/j.infsof.2003.09.011
6. Humeniuk, D., Antoniol, G., Khomh, F.: AmbieGen tool at the SBST 2022 tool competition. In: Proceedings of the 15th Workshop on Search-Based Software Testing, pp. 43–46 (2022). https://doi.org/10.1145/3526072.3527531
7. Humeniuk, D., Khomh, F., Antoniol, G.: AmbieGen: a search-based framework for autonomous systems testing. arXiv preprint arXiv:2301.01234 (2023). https://doi.org/10.48550/arXiv.2301.01234
8. Khari, M., Kumar, P.: An extensive evaluation of search-based software testing: a review. Soft Comput. Fusion Found. Methodol. Appl. **23**(6), 1933–1946 (2019). https://doi.org/10.3233/ICA-190616
9. Michel, O.: Cyberbotics Ltd. Webots^{TM}: professional mobile robot simulation. Int. J. Adv. Robot. Syst. **1**(1), 5 (2004). https://doi.org/10.5772/5618
10. Mondada, F., et al.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco (2009)
11. Pavel, A., Arsene, O., Buiu, C.: Enzymatic numerical P systems - a new class of membrane computing systems. In: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), pp. 1331–1336. IEEE (2010). https://doi.org/10.1109/BICTA.2010.5645071
12. Pérez-Hurtado, I., Martínez-del Amor, M.A., Zhang, G., Neri, F., Pérez-Jiménez, M.J.: A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. Integr. Comput.-Aided Eng. **27**(2), 121–138 (2020). https://doi.org/10.3233/ICA-190616
13. Păun, G.: Membrane Computing: An Introduction. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-642-56196-2
14. Păun, G., Păun, R.: Membrane computing and economics: numerical P systems. Fundam. Inform. **73**(1–2), 213–227 (2006). http://content.iospress.com/articles/fundamenta-informaticae/fi73-1-2-20
15. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford (2010)

16. Webots reference manual. https://cyberbotics.com/doc/reference/proto
17. Florea, A.G., Buiu, C.: PeP - an open-source software simulator of numerical P systems and numerical P systems with enzymes (2017). https://github.com/andrei91ro/pep
18. Github project. https://github.com/radubobe/Research/tree/main/Modelling
19. Github simulation results folder. https://github.com/radubobe/Research/tree/main/Modelling
20. Hellstrom, T.: Kinematics equations for differential drive and articulated steering. Umea University (2011). https://www8.cs.umu.se/kurser/5DV122/HT13/material/Hellstrom-ForwardKinematics.pdf
21. Simulation of E-puck controlled by Enzymatic Numerical P Systems models in Webots. https://youtu.be/FA7snrqaKKs
22. Wang, X., et al.: Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots. Integr. Comput.-Aided Eng. **23**(1), 15–30 (2016). https://doi.org/10.3233/ICA-150503
23. Whitley, D., Rana, S., Dzubera, J., Mathias, K.E.: Evaluating evolutionary algorithms. Artif. Intell. **85**(1–2), 245–276 (1996). https://doi.org/10.1016/0004-3702(95)00124-7
24. Zhang, G., Pérez-Jiménez, M.J., Gheorghe, M.: Real-Life Applications with Membrane Computing. Springer, Cham (2017). https://doi.org/10.1007/s00500-017-2906-y