P Colonie 0.94a – 2D P Colony simulator

User´s Guide

2D P Colony

We briefly summarize the notion of 2D P colonies. A 2D P colony consists of agents and the environment. Both the agents and the environment contain objects. With each agent the set of programs is associated. There are three types of rules, which are grouped in pairs into the programs.

The first rule type, called the evolution rule, is of the form $a \to b$. The second rule type, called the communication rule, is of the form $c \leftrightarrow d$. The third rule type, called the motion rule, is of the form *matrix* $3 \times 3 \to$ *move direction*. Based on the contents of the neighbouring cells, an agent can move one step to the left, right, up or down. If there is object $*$ inside the matrix, the agent does not take into account objects in corresponding cell. A program can contain maximum one motion rule. When there is a motion rule inside a program, there cannot be a communication rule inside the same program.

Definition 1. The 2D P colony is a construct $\Pi = (A, e, Env, B_1, \dots, B_k), k \geq 1$, where
A is an alphabet of the 2D P colony, its elements are called objects,
$e \in A$ is the basic environmental object of the 2D P colony,
Env is a pair $(m \times n, w_E)$, where $m \times n$; $m, n \in N$ is the size of the environment and w_E is the initial contents of environment, it is a matrix of size $m \times n$ of multisets of objects over $A - \{e\}$.
$B_i; 1 \leq i \leq k$, are agents, each agent is a triple $B_i = (O_i, P_i, [x_i, y_i]), 0 \leq x_i < m, 0 \leq y_i < n$, where
O_i is a multiset over A, it determines the initial state (contents) of the agent, $|O_i| = 2$,
$P_i = \{p_{i,1}, \dots, p_{i,l_i}\}, l_i \geq 1, 1 \leq i \leq k$ is a finite set of programs, where each program contains exactly 2 rules, which are in one of the following forms each:
$a \to b$ called the evolution rule, $a, b \in A$;
$c \leftrightarrow d$ called the communication rule, $c, d \in A$;
$[a_{qr}] \to s$; $0 \leq q, r \leq 2, a_{qr} \in A \cup \{*\}, s \in \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\}$ called the motion rule;
The third part of program is natural number $h \in N$, which determine priority level of the program.
$[x_i, y_i]$ are the coordinates of the initial location of the agent B_i in the environment.
$f \in A$ is the final object of the 2D P colony.

The configuration of the 2D P colony is given by the state of the environment - matrix of type $m \times n$ with multisets of objects over $A - \{e\}$ as its elements, and by the state of all agents - pairs of objects from alphabet A and the coordinates of the agents. An initial configuration is given by the definition of the 2D P colony.

The computational step consists of three parts. The first part lies in determining the applicable set of programs according to the actual configuration of the 2D P colony having regard to the priority levels of the programs. In the second part we have to choose one program corresponding to each agent from the set of applicable programs with maximum priority level. The third part is the execution of the chosen programs.

A change of the configuration is triggered by the execution of programs and it involves changing the state of the environment, contents and placement of the agents.

The computation is nondeterministic and maximally parallel. The computation ends by halting when no agent has an applicable program. The result of the computation is the number of copies of the final object placed in the environment at the end of the computation.

Description of simulation environment

The simulation environment has been written in Java and it allows us to load, save and create simulations using XML markup language.

The simulation file is loaded using XML parser and it creates a tree structure of objects using DOM and JAXP. These objects represent parameters of the simulation, which contain a description of the environment and agents in this environment. The information describing the environment includes parameters such as speed of simulation, the size and the contents of the environment. The speed of the simulation determines the time interval between the steps of the simulation. The environment and its contents are represented by a two-dimensional array of objects which is displayed as a 2D grid to the user. The agent is located in this grid and has the ability to move or influence the contents of the environment by using rewriting rules. The environment may contain a special object \#, which represents an obstacle or a position that is inaccessible for agents. The agent in the environment activates one of its applicable programs in each simulation step. Each program has an assigned priority and the selection of applicable programs is based on this priority. If there is a state when several programs can be activated with the same priority, we use pseudo-random selection to choose only one of these programs. Multiple agents can be located on different or identical positions in the simulation environment. Collisions may occur in simulations of several agents in a common shared environment which can cause simulation errors. To avoid these problems, agents need to synchronize their access to the environment and again using a pseudo-random selection to decide the order in which the agents will be on the same positions to activate their programs. Environment changes are stored in the stack from which they are projected into the environment. In this way we can avoid the situation when one agent in the simulation step will affect the neighbourhood of another agent or objects in the position where there are more agents. In these cases it could lead to the use of previously unusable agent programs. However, in one simulation step this is not acceptable. Our simulation tool uses the Swing library for creating graphical user interfaces. It is possible to use change the graphics of the environment, the agents and the obstacles and customize the simulation environment and visualization according to user needs. Users now have an interesting tool for the implementation of the simulation of P colonies with the ability to edit the simulation directly from the simulation environment or from any text editor.

Structure of XML file

The first line determine type of the dokument:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Tag simulation contains definition of 2D P colony, it atribute characterize speed of simulation

```
<Simulation speed="200"> … </Simulation>
```

Description of 2D P colony consists of two parts: definitiv of the environment and definitiv of th agents.

Definition of the environment:

```
<Environment sizex="10"
sizey="10">e,e,e,e,e,e,e,e,e,e;e,e,e,e,e,e,e,e,e,e;e,e,e,e,e,e,e
,e,e,e;e,e,e,e,e,e,e,e,e,e;e,e,e,e,e,e,e,e,e,e;e,e,e,e,e,e,e,e,e
,e;e,e,e,e,e,e,e,e,e,e;e,e,e,e,e,e,e,e,e,e;e,e,e,e,e,e,e,e,e,e;e
,e,e,e,e,e,e,e,e</Environment>
```

Definition of the agent is set within tags:

```
<Agents>…</Agents>
```

Each agents has free atributes: contain, position x and position y:

```
<Agent contain="aa" posx="4" posy="4"></Agent>
```

Agent contains programs with priorities:

```
<Program priority="10"></Program>
```

Program is composed of rules:

1. Moving rule:

```
<Rule alfa="?,?,?;?,e,?;?,?,?" beta="0" op="move"/>
```

2. Rewriting rule

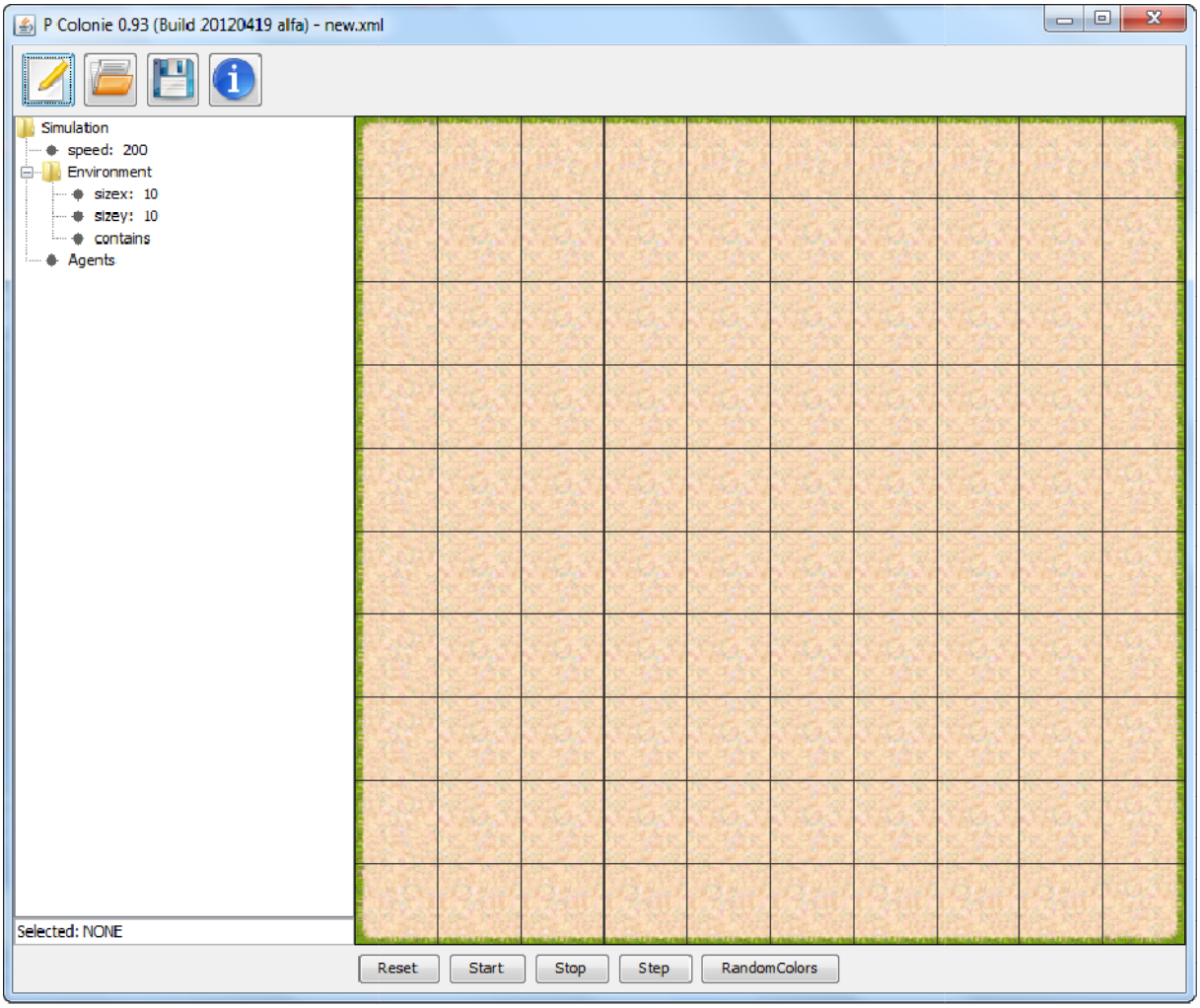
```
<Rule alfa="a" beta="a" op="RW"/>
```

3. Communication rule:

```
<Rule alfa="M" op="CH" beta="e"/>
```

Gui

The following figure shows GUI of the main window of the application.



The GUI of the application is relatively simple and it is divided into the following sections:

1. buttons to create new simulation, load saved simulation and save current simulation
2. buttons to control simulation process – reset, start continuous simulatio, stop simulation process, do one step of process and randomize colors of objels in the environment
3. left panel – show and edit current 2D P colony
4. main part with environment.

During process (computation) log is generated as excel file with configurations of the 2D P colony reached during computation.

Future work

We plan to extend the simulator to use statistical tools and dynamic environment in the future.

References:

[1] Cienciala, L. & Ciencialová, L. & Perdek, M.: *2D P colonies*. In Csuhaj-Varjú et al. (eds.). CMC 2012, Springer, LNCS 7762, 2013, pp. 161-172.

[2] Ciencialová, L., Cienciala, L., Perdek, M.: *2D P colonies used in hydrology simulations*. The proceedings 14 th GeoConference on Informatics, Geoinformatics and Remote Sensing, Sofia, Bulgaria, 2014, pp 3 - 10. ISBN 978-619-7105-10-0, ISSN 1314-2704, DOI: 10.5593/sgem2014B21.

[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.