Autonomous Intelligent Systems,
Institute for Computer Science VI, University of Bonn

Dr. N. Goerke
Friedrich-Ebert-Allee 144, 53113 Bonn, Tel: (0228) 73-4167
E-Mail: goerke@ais.uni-bonn.de
www.ais.uni-bonn.de

Exercises for Artificial Life (MA-INF 4201), SS15

Exercises sheet 3, till: Mon 4.5.2015

27.4.2015

	Name	16	17	18	19	20	21	22	23	$\sum$

Assignment 16 (1 Point)

Characterise the rule of Conway's Game of Life with respect to the following rule properties: silent state, totalistic, symmetric, peripheral, legal.

Assignment 17 (1 Point)

What has been proven by the development of Gosper's Glider Gun?

Assignment 18 (2 Points)

What class, or kind of behaviour (Wolfram's classification) is the *r-pentomino* showing? Support your answer with scientific arguments.

Assignment 19 (3 Points)

Imagine an exam question asking to depict a 5-cell configuration for Conway's Game of Life CA, that has been produced by Gosper's Glider gun.
How many correct configurations exist?
Please support your answer by an explanation.

Assignment 20 (3 Points)

The idea has come up to use Cellular Automata to work on non-regular grid structures and meshes. Develop and describe a way to operate a Cellular Automaton to work with the following three topologies:
an undirected graph, a tree, a binary tree.

Please describe how the topology is influencing the structure and the definition of the rules, the neighborhood, the number of possible rules. Propose a way to calculate a numerical value for describing the rule (following the concept of Wolfram Numbers).

Assignment 21 (1 Point)

Find an application for cellular automata (that has not been mentioned in the lecture).
Give a scientific acceptable reference for the application.

Assignment 22 (2 Points)

What is the smallest possible configuration for Conway's Game of Life (i.e., using the fewest number of living cells) that will completely die out in a single update?
To make this question nontrivial, at least one cell must die from overcrowing (from having four or more neighbors).
(*Adopted from Problems 1.1, C. Adami, Introduction to Artificial Life, Springer 1999*).

Assignment 23 (2 Points)

Draw pictures, that visualize the methods of *Fixed Boundary* and *Adiabatic Boundary, mirror* for d=1, $k > 2$, r=2, and r=3, Cellular Automata.

Programming Assignment: B (10 Points, due date **Mon 11.5.2015**)

Implement a two-dimensional (d=2) (Cellular) Automaton on a 101×82 sized, rectangular grid. Use a torus toplogy to manage the boundary of the grid. If you use console output (ASCII-art) take characters that are aligned with the content they represent and that are easy to distinguish from each other.

You can choose if you would like to implement Langton's Ant, Conway's Game of Life or the probabilistic Cellular Automaton called "Forest Fire".

Langton's Ant:
Make the user choose the initial configuration of the grid from the following list of 5 possibilities: all white, all black, checker board, horizontal stripes, random setting and make the user chose the starting position and orientation for the ant.
Write the total number of cells living for each time step into a file (one ASCII value per line).

Conway's Game of Life:
Make the user choose the initial pattern to start with, from the following list of 5 possibilities: blinker, glider, r-pentomino, Gosper's Glider Gun, and a pattern (class 3 or 4) of your own choice.
Write the total number of cells living for each time step into a file (one ASCII value per line).

Forest Fire Model:
Make the user set the probabilistic parameters p, f, q. Start with an *empty* field (all cells are in state *Ashes*).
Write the number of cells for each state (Ashes, Tree, Fire) for each time step into a file (three values per line, ASCII format, separated by blanks).

Implement your program using C, C++, Java or Python:
Make sure your program can be compiled and started from a console, or a terminal; using an IDE is fine for developing, but make sure the final program can be operated without the IDE.
Send an E-Mail to your tutor containing the **documented** source code and **a description how to compile and run your program** from the console (no IDE).