

# Exercises for Automaton

for special course in 2012

Akira Imada

Brest State Technical University, Belarus

(last modified on)

April 28, 2012

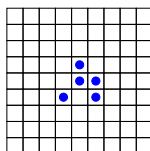
## 1 Conway's Game of Life

**Exercise 1** *Create a troidal gridworld as large as your screen size. Then starting with an initial pattern such as shown bellow, apply the rule of state transition (also shown below) and observe how they explore the gridworld.*

### Rule 1 (Classification only with prior probability)

1. *Any live cell will die if its live neighbours are fewer than two (too lonely to live).*
2. *Any live cell will remain live if its live neighbours are two or three.*
3. *Any live cell will die if its live neighbours are more than three (too crowdy to live).*
4. *Any dead cell will become a live cell only if its live neighbours are exactly three, otherwise remain die.*

Note that the number of live neighbors is based on the cells before the rule was applied. In otherworld, the rule should be applied to all the cells before changing states, and then change the state of all the cells simultaneuousy in the next time flame.

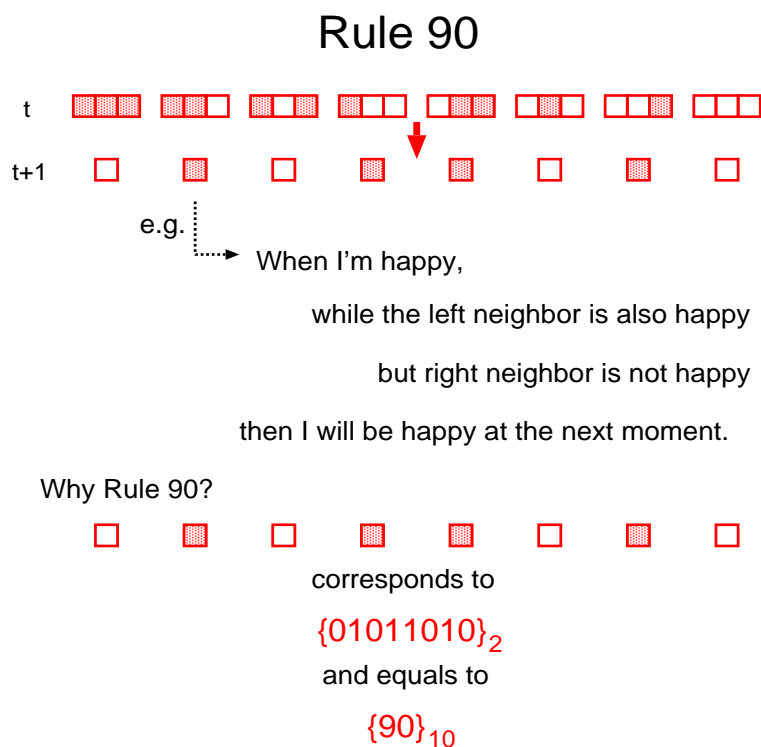


Also try to run with other initial pattern you can see by clicking "initial patterns of the Game of Life," in my home page.

## 2 Cell Automaton (CA)

**Exercise 2** Try to display a one-dimensional cell automaton with 41 cells starting with the one in which only center is 1 and all others 0, by applying the Rule 90 starting with a single black cell. Then display those evolved cells from  $t = 0$  to, say,  $t = 100$ .

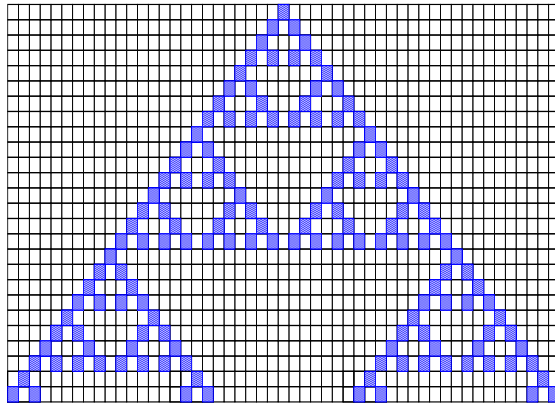
Now recall what is Rule-90. In 1-D cell automaton, state of a cell changes according to the states of two its neighbors - right and left one. The rule of state change can be expressed by 3-digit binary numbers. For example. If the cell is 1 and both neighbor are 1 also, let's the next state of the cell (center) be 0. This is represented as  $111 \Rightarrow 0$ . Next, if the cell is 1 and left neighbor is 1 but the right neighbor is 0 then let's the center state be 1. Thus you may specify the rule by repeating this procedure from 111 to 000. And if, for example, the results of repeating this procedure 8 times, is 01011010 this rule is called *Rule 90* because this binary means 90 in the system of 10. See the Figure below.



Thus if you start with the 1-D cell array with only the center being 1 and all others 0, you may observe like below from  $t = 1$  to  $t = 3$ . Then try to continue



See one of our results during practice course.



From the work by Khanghee Lim (2012 Spring)

### 3 Finite State Automaton I: Lucky Dog

**Exercise 3** *Design an FSA to make a robot dog navigate a gridworld with, say, 4 States, 4 inputs, 4 actions using a transition table. Assume the gridworld is made up of, say, 200x200 rectangle cells. you put (i) a dangerous pond so that if the dog enters into it, the dog will die; (ii) wall(s) so that if the dog run into it, the dog can not go strait; (iii) and sausages somewhere in one sell so that the dog can enjoy eating it.*

*So, the inputs are (1) empty cell; (2) warter flont: (3) wall; and (4) sausage. Actions are (a) go one step to strait forward; (b) go one step backward; (c) go one step to the right cell; and (d) go one step to the left cell. Dog start at the center.*

*Thus, you create a couple of different such an FSA, and run one by one, and observe.*

For the sake of simplicity, let's the FSA with 2 states, in addition with only one input 0 meaning empty cell. Then an example of the transition table is: You may observe a

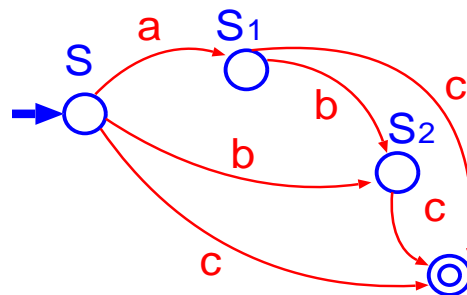
current state	input	action	next state
A	0	a	B
		c	A
		d	A
		b	A
B	0	b	B
		a	B
		c	A
		d	B

dog who wonders aimlessly in a empty gridworld. You change the 3rd and 4th columns randomly and observe the dog, from one run to the next. Then you may add the input one by one, or state C and/or D.

## 4 Language Recognition by FSA

**Exercise 4** *Design an FSA that can recognize strings over  $\{a, b, c, d, e\}$  in alphabetical order starting with  $a$ . Then implement it in your PC, and see if your input of a string from keyboard is correctly recognize.*

Recall we studied an FSA that can recognize strings over  $\{a, b, c\}$  in alphabetical order ending with  $c$ .



**Exercise 5** *Then expand it into the case of 5 letters or more.*

**Exercise 6** *Try also  $a^n b^n$ .*

$S$	$\Rightarrow$	$aA$
$A$	$\Rightarrow$	$b$
$A$	$\Rightarrow$	$aB$
$B$	$\Rightarrow$	$bb$
$B$	$\Rightarrow$	$abB$

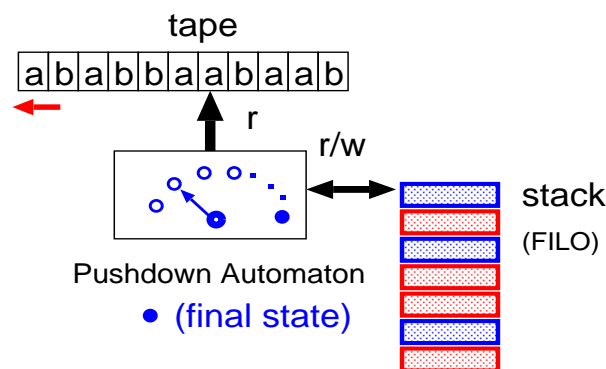
**Exercise 7** *Or,  $a^n b^n c^n$ .*

$S$	$\Rightarrow$	$abc$
$S$	$\Rightarrow$	$aAbc$
$Ab$	$\Rightarrow$	$bA$
$Ac$	$\Rightarrow$	$Bbcc$
$bB$	$\Rightarrow$	$Bb$
$aB$	$\Rightarrow$	$aa$
$aB$	$\Rightarrow$	$aaA$

## 5 Language Recognition by Pushdown Automaton

**Exercise 8** *Design a Pushdown Automaton that can recognize strings over  $\{a, b\}$  with a form of  $a^n b^n$ . In other words, a sequence of as followed by an equal number of bs. Then implement it in your PC, and see if your input of a string from keyboard is correctly recognize.*

First, let's recall what is pushdown automaton by seeing the following schematic diagram.

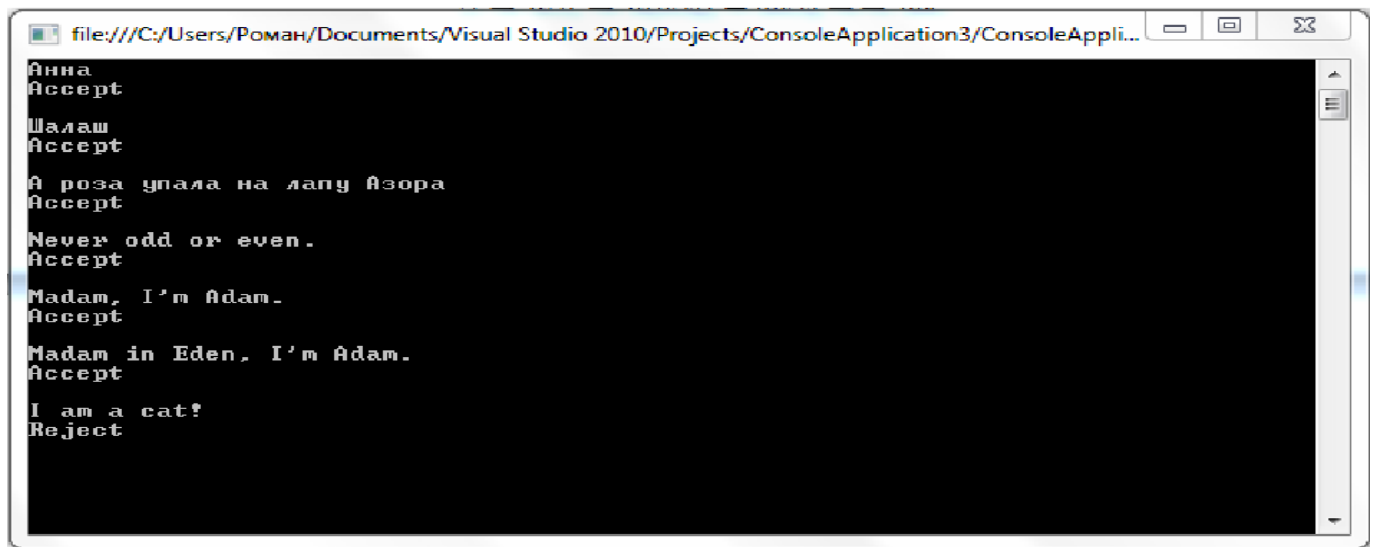


Thus, the input is given from the tape. For example  $aaaabbbb$  from left to right. Assume we have a number of small disks, like a jeton for the Minsk Metro. Then a possible algorithm is:

### Algorithm 1 (Pushdown Automaton)

- 1 Place the tape head on the leftmost input symbol.
- 2 WHILE symbol read = a:
  - (i) Advance tape head one cell to the right.
  - (ii) Place one disc on the top of the stack
- 3 WHILE symbol read = b and the stack still contains discs:
  - (i) Advance tape head one cell to the right.
  - (ii) Remove disc from the top of stack.
- 4 IF input has been all scanned and the stack is empty THEN accept.  
ELSE reject.

**Exercise 9 (Palindrome)** *Design an Automaton that can recognize palindrome.*

A screenshot of a console application window titled "file:///C:/Users/Роман/Documents/Visual Studio 2010/Projects/ConsoleApplication3/ConsoleAppli...". The window has a black background with white text. The text shows the results of a palindrome recognizer for various inputs. The inputs and their corresponding results are: "Анна" (Accept), "Шалаш" (Accept), "А роза упала на лапу Азора" (Accept), "Never odd or even." (Accept), "Madam, I'm Adam." (Accept), "Madam in Eden, I'm Adam." (Accept), and "I am a cat?" (Reject).

```
Анна
Accept
Шалаш
Accept
А роза упала на лапу Азора
Accept
Never odd or even.
Accept
Madam, I'm Adam.
Accept
Madam in Eden, I'm Adam.
Accept
I am a cat?
Reject
```

From the work by Roman Lisenkov, Aleksander Mokin, Ilya Mokin, and Vasily Brutsky  
(2012 Spring).