UDC 681.5

N. Smolij, O. Lisovychenko, V. Smolij

SIMULATION TOOLS: FORMAL LANGUAGE FOR CELLULAR AUTOMATONS BEHAVIOR DESCRIPTION

Abstract: Object is a formal language for cellular automatons behavior description. The main tasks are 1) to describe variety of possible events that cell can experience in cellular automaton; 2) to define possible attributes that can cause these events (over/under population leads to death, neighboring cell has state that affects next state of current cell, etc.); 3) to describe language that describes connections between states and events. Program environment for cell automaton and conduct research on existing rules for cellular automatons are created in order to learn about events that can occur with cell and attributes of cell that can lead to these events. In this way information that is required to build terminal vocabulary of language can be found. Then main non-terminal symbol of the language must be discovered and grammar terms it consists of. Thus, grammar structure of formal language will be defined and it becomes possible to build grammar-recognizing automaton. It will provide possibility of recognizing sentences written in our informal language and interpretation of written information will become possible.

Keywords: cellular automaton, simulation, instruction language.

Problem description

Cellular automaton by itself is a discrete model of computation studied in automata theory. As a simplified description of cellular automaton 2-dimentional array filled with different integer numbers can be given. As cells in real life, it evolves with time. Next state of every cell depends on its neighborhood and set of rules that connects variety of cell states with a variety of possible neighbors’ placements. As an example John’s Conway’s “Life Game” can be given. It’s outer-totalistic, irreversible cell automaton. Irreversible means that previous state of array can’t be re-built using current state because different states can evolve into same state. Outer-totalistic means that the state of the cell in the next generation depends on both its own state and states of its neighbors [2].

The main tasks are 1) to describe variety of possible events that cell can experience in cellular automaton; 2) to define possible attributes that can cause these events (over/under population leads to death, neighboring cell has state that affects next state of current cell, etc.); 3) to describe language that describes connections between states and events.

ISSN 1560-8956

The main content and results of the work

Program environment for cell automaton and conduct research on existing rules for cellular automatons are created [1] in order to learn about events that can occur with cell and attributes of cell that can lead to these events. In this way information that is required to build terminal vocabulary of language can be found. Then main non-terminal symbol of the language must be discovered and grammar terms it consists of. Thus, grammar structure of formal language will be defined and it becomes possible to build grammar-recognizing automaton. It will provide possibility of recognizing sentences written in our informal language and interpretation of written information will become possible [3].

Results**.** 1) List of discovered events occurred with cell: birth**; s**urvival (cell didn’t die during evolving of cell automaton)**;** change (cell changed its type)**;** death.

2) List of possible attributes that can cause events are distinguished: number of neighbors; type of cell and place of cell relative to another cells.

3) Main non-terminal symbol is code like B3/S2,3. It consists of smaller parts: pairs of event-condition separated with ‘/’ sign. All this information can be used for build informal language. The proper form of sentence in this language is:

N:<name of cell type>

<event>:<condition>/<event>:<condition>/<event>:<condition>/<event>:<conditio>

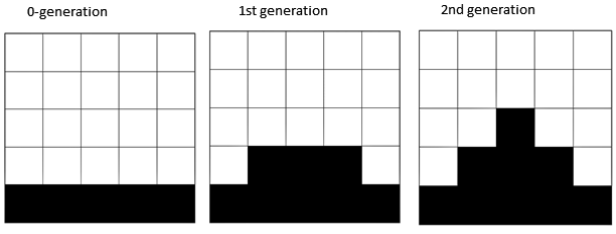
Where <event> may be the one event from the list above and <condition> may relate to one or many of the attributes. It may be a case when we need to make references to many attributes. In this case <condition> non-terminal will be divided into parts:

<amount reference><placement reference><type reference>

As an example, there is a code that describes pyramid building in cell automaton:

N:block/B:3|down|block

In this case the following changes will happen (fig. 1):



a) b) c)

Figure 1. Building a pyramid in a cellular automaton

Possible amount references: 1,4,6,7 – enumeration; [2;8]-non-strict array; (2;8)-strict array.

<array1>U<array2> - logical “or”-array1 with array2

<array1>!<array2> - logical “not”-array1 without array2

Possible placement references:

None-8 cells around current

Up/Left/Right/Down-to “look” at 3 cells in according direction.

By pointing at exact cell using coordinate system connected with current:

<-1;0>

By two cells using coordinate system connected with current:

<-1;0><1;3>c - rectangular from 11 cells(“c” includes current cell in array).

By concatenation or cutting of rectangles or points using logic operators:

<-1;0><1;3>^/U/!<-1;-1><1;3>^/U/!<p1;p2>^/U/!...

Possible type references can be made with pointing on exact type, enumeration of types and using logic operators “!”,”^”,”U”(not, or, and). Also logical operators ^,U and ! can be placed between conditions in event groups in order to make proper logic formulas.

A few words about cellular automatons should be mentioned:

Type references can’t be used without amount or placement reference. Cellular automatons are described as 0-player game. “Player” can only describe first state of automata can observe it’s evolution. These automatons with specific set of rules are Turing complete machines.

They can be described as a regular cell grid and every cell has a state. It can be zero or non-zero value. Grid can be any size that you want. For every cell there is a defined set of cells that will be analyzed while evolution into the next state.

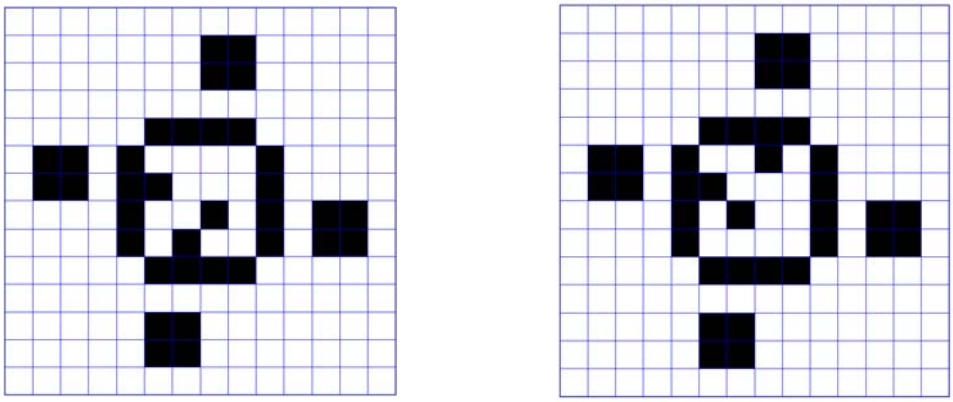
So, to make cell automata work, zero generation of cells and set of rules for evolution is required.

In the fig.2, you can see "clocks" made with "Life Game". Three dots in the centre will circle for eternity.

The following picture has more rules (fig. 3). Different rules: used by biologist to predict population evolution, - used by mathematicians in different algorithm models with different purposes.

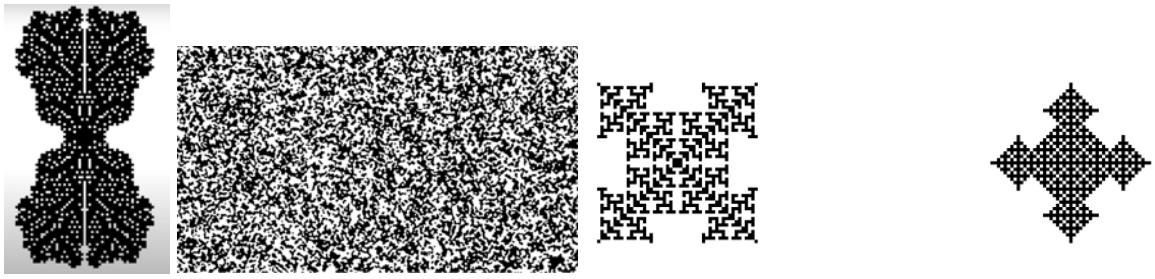
The first picture (fig. 3, a) is a picture of cellular automaton with rules B1/S[1;8]. This set of rules described cells as born with one neighbor and surviving in any condition. Picture of a strange rectangle (fig. 3, b) is cellular automaton with set of rules B. In the next picture there two similar automatons: both use rules B/S. But neighborhood for the first one is 8 cells (fig. 3, c) and for the second only 4 in form of + sign (fig. 3,d).

ISSN 1560-8956



a) 0-generation is made by user b) 1st generation is made according to automata rules

Figure 2. "Clocks" made with "Life Game"



a) b) c) d)

Figure 3. Cellular automaton with different sets of rules

Cell events can be affected by:

- Over/under population.
- It may happen in particular area (8 or 4 cells around).
- The types of cells can have an impact on the next evolution step.

So, every cell may experience 4 events: birth, surviving, type changing and death.

The last two are required for advanced rule building for cell automatons that can work with more than one type of cell.

Building condition descriptions for each of these events may relate to the following attributes:

Neighborhood.

Amount of neighbors.

And the type of these neighbors.

Creation of flexible rules:

- Using of logic operators such as “or”, “and”, “not” will make creation of complicated conditions possible.
- In according to definition of condition some events are required: birth; surviving; changing; death.

Maybe using complicated rules will be required and for this using of logic operators such as or and not should be allowed.

In this case writing such rules as:

Birth if surrounded with 3 cells or one cell of type parent becomes possible.

But a regular structure of rules is required to make it possible for computers to recognize information.

For this we need to define the main non-terminal symbol (Sentence in regular language) and dependent terminals(Noun/Verb phrase, Noun, Verb, word, etc.)

Also alphabet, that will include: logic operators, letters, symbols and letters is required.

Detailed information is given in fig. 4.

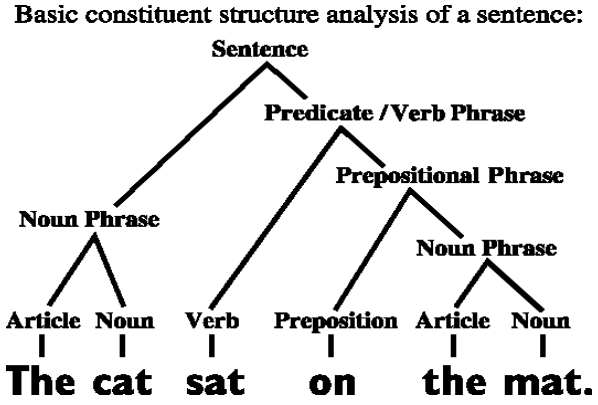


Figure 4. Creating formal language

As a result of all above next structure can be build and used to describe a great variety of possible rules for cellular automatons (tabl. 1).

Table 1. Sets of possible rules for cellular automatons

Possible structure of rules:	**Possible structure of condition:**
N:<name> B:<condition 1>^<condition 2>.. S:<condition 1>^<condition 2>.. C:<condition 1>^<condition 2>..-><name> D:<condition 1>^<condition 2>..	<Type reference 1^Type reference 2.. |area reference 1 ^area reference 2.. |neighbors reference 1^neighbourrs reference 2..>

At the left side of the slide you can see possible structure of the rule. There is traditional parts for birth and surviving conditions. Also there are field for describing specific death or changing conditions in case if you need them.

And on the right side there is structure of conditions, where you can specify amount of neighbors, area where they will be watched for and type of this neighbors.

ISSN 1560-8956

Conclusion

To sum up, problems of creation of artificial languages and integration of these languages into software such as cell automatons are difficult and require further researches. But result of such researches may lead to unpredictable results from discovering of new simulation tools to creation of new algorithms that can be used in programming. As a result of this research creation of one artificial language for describing cell behavior in cell automatons can be presented.

REFERENCES

1. Wolfram S. *(*n.d.). WOLFRAM ATLAS: CELLILAR AUTOMATA. Retrieved from http://atlas.wolfram.com/01/01/.

2. Shkilnyak S. Expressiveness in algebraic systems. Arithmetic predicates, sets, functions. In Academic Council of the Interregional Academy of Management staff (Ed.), FORMAL MODELS OF ALGORITHMS AND ALGORITHMICALLY CALCULATED FUNCTIONS (4th ed., pp.63). Kiev., Ukraine: Publishing house “Personnel”.- 2009.

3. Martynenko B.K. Formal grammar definitions. In E. A. Hirsch (Ed.), Languages and broadcasts (2nd ed., pp.14). St. Petersburg, Russia: ST. PETERSBURG UNIVERSITY PUBLISHINH HOUSE. - 2013.