Tools for computer research of cellular automata dynamics

V.Z. Aladjev [a], ***V.K. Boiko*** [b]

a E–mail: aladjev@yandex.ru; International Academy of Noosphere; Tallinn, Estonia
b E–mail: boiko@grsu.by; Grodno State University; Grodno, Belarus

To the present, the problematics of *Cellular automata (CA)* well enough is advanced, being quite independent field of modern mathematical cybernetics, having own terminology and axiomatics at existence of a rather broad field of various appendices. *CA* is a parallel information processing system consisting of intercommunicating identical finite automata. Although the *CA* term will be used throughout the paper as the usual term, it is necessary to keep in mind that the *CA*, iterative networks, etc. are essentially synonyms. We can interpret *CA* as a theoretical basis of artificial parallel information processing systems. From logical standpoint the *CA* is an infinite automaton with specifical internal structure. The *CA* theory can be considered as a structural and dynamical theory of the infinite automata. *CA* models can serve as an excellent basis for modelling of many discrete processes, representing interesting enough independent objects for research too. Now, the undoubted interest to the *CA* problematics has arisen anew and in the given direction many remarkable results have been obtained [1–4].

So, the *CA* axiomatics provides such three fundamental properties as homogeneity, localness and parallelism of functioning. If in a similar computing model we shall associate with an elementary automaton a separate microprocessor then it is possible to unrestrictedly increase sizes of such computing system without any essential increase of temporal and constructive expenses, required for each new expansion of the computing space, and also without any overheads connected to coordination of functioning of arbitrary supplementary quantity of elementary microprocessors. Similar high–parallel computing models admit practical realizations consisting of large enough number of elementary microprocessors which are limited not so much by certain architectural reasons as by a lot of especially economic and technologic reasons defined by a modern level of development of microelectronic technology, however with the great potentialities in the future, first of all, in light of rather intensive works in field of nanotechnology [5].

The above three such features as high homogeneity, high parallelism and locality of interactions are provided by the *CA* axiomatic, while such property important from the physical standpoint as reversibility of dynamics is given by program way. In light of the listed properties even classical *CA* are high–abstract models of the real physical world, which function in a space and time. For this reason, they in many respects better than many others formal architectures can be mapped onto a lot of physical realities in their modern understanding. Moreover the *CA* concept itself is enough well adapted to solution of various problems of modelling in such areas as mathematics, cybernetics, development biology, theoretical physics, computing sciences, discrete synergetics, dynamic systems theory, robotics, etc. Told and numerous examples available for today lead us to the conclusion that *CA* can represent a rather serious interest as a new perspective modelling environment of modelling and research of many discrete processes and phenomena, determined by the above properties; in addition, raising the *CA* problematics onto a new interdisciplinary level and, on the other hand, as an interesting enough independent formal mathematical object of researches.

Meantime, in spite of extremely simple concept of the classical *CA*, they have generally speaking a complex enough dynamics. In very many cases theoretical research of their dynamics collides with essential enough difficulties. For this reason, computer simulation of these structures that in the empirical way allows to research their dynamics is a rather powerful tool. For this reason this question is quite natural for considering within the present paper, considering the fact that *CA* at the formal level represent the dynamical systems of highly parallel substitutions. The detailed enough discussion of the problem of computer simulation of the *CA* can be found in [1-4]. In the same place it is possible to familiarize in details with the *CA* concept and its discussion.

At present, the problem of computer modelling of the *CA* is solved at *2* levels**:** *(**1**) software that modells the dynamics on computing systems of traditional architecture*, *and (**2**) simulation on the hardware architecture that as much as possible corresponds to the CA concept**;** so–called CA–oriented architecture of computing systems*. Thus, computer simulation of *CA* models plays a rather essential part at theoretical researches of their dynamics, in the same time it is even more important at practical realizations of the *CA* models of various processes. At present, a whole series of rather interesting systems of software and hardware for giving help to researchers of different types of the *CA* models has been developed**;** their characteristics can be found in [3-4].

In our works many programs in various program systems for different computer platforms had been represented. In particular, tools of the ***Mathematica*** system support algebraic substitutions rules that allow to easily model the local transition functions of the classical ***1***–dimension *CA*. In this context many interesting programs for simulation of the *CA* models in the ***Mathematica*** had been created. On the basis of computer simulation a whole series of rather interesting theoretical results on the theory of classical *CA*–models and their applications in the fields such as computer sciences, developmental biology, mathematics, etc. had been received. By way of illustration a number of the procedures providing the computer research of certain aspects of dynamics of the classical ***1***–dimension *CA* in the ***Mathematica*** system is represented below.

In researches of the *CA* models an essential enough role is played by a research of dynamics of configurations-predecessors that is caused by a research of a problem of the nonconstructability connected with important problem of reversibility in the *CA* models. The means, programmed by us are focused on a computer research of important aspects *CA* dynamics because of complexity of their theoretical research. These tools allow to obtain not only estimated characteristics of the studied dynamics, but in many cases allow to obtain serious hints on further ways of research. At the same time, it must be kept in mind that the procedures given below in turn may contain non-standard, relatively ***Mathematica,*** software, but which is documented and are in the ***MatToolBox*** package with freeware license [5]. These procedures in many respects allow rather essentially to simplify programming, to optimize and make more compact program codes.

So, the procedure call ***Predecessors*[*L, Co, n*]** on the basis of a ***L*** list which determines the local transition function of a ***1***–dimension classical *CA* with ***n*** size of its neighbourhood template and initial ***Co*** configuration – *a continuous finite block of states of elementary automata* – returns the list of *configurations–predecessors* for the block ***Co*** configuration. At that, parallel substitutions *x1x2x3...xn* **–>** *x*1* that define the local transition function of the classical ***1–CA*** in the ***L*** list are represented by strings of the format *"x1x2x3...xnx*1"*. In particular, the procedure can identify existence for an arbitrary classical ***1–CA*** of the nonconstructability of *NCF*–type with printing the appropriate message. The source code of the procedure is represented below.

```
Predecessors[Ltf_ /; ListQ[Ltf], Co_ /; StringQ[Co], n_ /; IntegerQ[n]] :=
   Module[{L, a, b, c, h = {}, i, j, k, d = StringLength[Co]},
      a = Gather[Ltf, StringTake[#1, –1] === StringTake[#2, –1] &];
    For[k = 1, k <= Length[a], k++,
      L[StringTake[First[a[[k]]], –1]] = Map2[StringDrop, Map[ToString1, a[[k]]], {–1}]];
      b = L[StringTake[Co, 1]];
     For[k = 2, k <= d, k++, c = L[StringTake[Co, {k, k}]];
      For[i = 1, i <= Length[b], i++,
        For[j = 1, j <= Length[c], j++,
      If[SuffPref[b[[i]], StringTake[c[[j]], n – 1], 2],
      h = Append[h, b[[i]] <> StringTake[c[[j]], –1]], Null]]]; b = h; h = {}];
    If[Length[b] != (n – 1)^Length[a],
      Print["Structure possesses the nonconstructability of NCF–type"], Null]; b]
```

While the ***PredecessorsL*** and ***PredecessorsR*** procedures act as certain extensions of the above procedure, allowing to obtain interesting enough special results of the *CA* dynamics [3,5]. As a whole, the above procedures allow experimentally to investigate a whole series of aspects of the reversibility problem in classical ***1–CA***.

The procedure call ***RevBlockConfig*[*C, Ltf*]** returns *True* if a list of block configurations which are predecessors of a finite block configuration ***C***, relative to a local transition function ***Ltf*** given by the list of rules *(the parallel substitutions)* is other than the empty list, and *False* otherwise. While the call ***RevBlockConfig*[*C, Ltf, h*]** with optional ***3rd*** argument ***h*** *– an indefinite symbol* **–** through it returns the list of all predecessors of the ***C*** configuration. The next fragment represents source code of the ***RevBlockConfig*** procedure with the typical examples of its application.

```
In[4449]:= RevBlockConfig[C_ /; StringQ[C], Ltf_ /; ListQ[Ltf] &&
   AllTrue[Map[RuleQ[#] &, Ltf], TrueQ], h___] :=
   Module[{a = CollectRules[Ltf], b = Characters[C], c = {}, d, p,
     n = StringLength[Ltf[[1]][[1]]] – 1, g},
     d = Flatten[Map[ListToRules, a], 2]; c = Flatten[AppendTo[c, Replace[b[[1]], d]]];
  Do[c = RepSubStrings1[c, If[Set[g, Replace[b[[k]], d]] === b[[k]], g = 74; Break[], g], n],
     {k, 2, Length[b]}];
  If[g === 74, Return[False], Null]; If[{h} != {} && ! HowAct[h], h = c, Null];
  If[Max[Map[StringLength, c]] == StringLength[C] + n, True, False]]
In[4450]:= {RevBlockConfig["0110111101010", {"00" –> "0", "01" –> "1", "10" –> "1",
"11" –> "0"}, v75], v75}
Out[4450]= {True, {"00100101001100", "11011010110011"}}
```

The research of configurations dynamics, i.e. the sequences of the configurations generated by the ***1–CA*** from initial configurations represents special interest, and here very important part is assigned to computer simulation. In this direction a number of procedures oriented on research of various aspects of configurations dynamics of has been created [4,5].

In particular, the procedure call ***CFsequences*[*Co, A, L, n*]** prints the sequence of configurations generated by a ***1–CA*** with alphabet of states ***A***=*{0,1, ..., p} (p = 1..9)*, local transition function ***L*** from a finite ***Co*** configuration, given in string format, during ***n*** steps of the automaton. At that, a function of the kind F[x, y, **...**, t] := x*****, and the list of substitutions of the kind "xy **...** t" **–>** "x*" {x,x*,y,z,**...**,t}∈***A*** can act as the third argument ***L.*** The procedure processes basic mistakes arisen at encoding an initial configuration ***Co,*** an alphabet ***A*** and/or a local transition function ***L*** with returning $*Failed* and printing of strings with the appropriate messages. The source code of the procedure with an example of its application are represented below.

```
In[3345]:= CFsequences[Co_ /; StringQ[Co] && Co != "", A_ /; ListQ[A] &&
   MemberQ[Map[Range[0, #] &, Range[9]], A], L_ /; ListQ[L] &&
   AllTrue[Map[RuleQ[#] &, L], TrueQ] || FunctionQ[L], n_ /; IntegerQ[n] && n >= 0]:=
   Module[{a = StringTrim2[C, "0", 3], b, c, t = {}, t1 = {}, t2 = {}, t3 = {}, f, p = n},
     If[! MemberQ3[Map[ToString, A], Characters[C]],
     Print["Initial configuration <"<> C <> "> is incorrect"]; $Failed,
     If[FunctionQ[Ltf], b = Arity[L], Map[{{AppendTo[t, StringQ[#[[1]]]],
        AppendTo[t1, StringLength[#[[1]]]]}, {AppendTo[t2, StringQ[#[[2]]]],
  AppendTo[t3, StringLength[#[[2]]]]}} &, L]; b=Map[DeleteDuplicates[#] &, {t,t1,t2,t3}];
  If[! (MemberQ3[{True}, {b[[1]], b[[3]]}] && Map[Length, {b[[2]], b[[4]]}] == {1, 1} &&
  Length[t1] == Length[A]^(b=b[[2]][[1]])), Print["Local transition function is incorrect"];
Return[$Failed], f=Map[ToExpression[Characters[#[[1]]]] –>ToExpression[#[[2]]] &, L]];
```

```
   c = StringMultiple2["0", b]; Print[a]; While[p > 0, p—; a = c <> a <> c;
   a = Partition[ToExpression[Characters[a]], b, 1];
   a = If[FunctionQ[L], Map[L @@ # &, a], ReplaceAll[a, f]];
   a = StringJoin[Map[ToString, a]]; Print[StringTrim2[a, "0", 3]]];]]
In[3346]:= CFsequences["100111100001", {0, 1}, {"00" –> "0", "01" –> "1", "10" –> "1",
"11" –> "0"}, 5]
"100111100001"
"1101000100011"
"10111001100101"
"111001010101111"
"1001011111110001"
```

A number of modifications of the ***CFsequences*** procedure has been programmed, including the procedures oriented on the dialogue mode.

So, the technology presented above on the basis of modifications of the ***CFsequences*** procedure allows to obtain a number of rather interesting properties of numeric sequences, generated by ***1***–dimensional binary *CA* models. In particular, the computer analysis by means of the above tools allows to formulate rather interesting assumptions, namely**:** ***1****-dimensional binary CA model with the local transition function Ltf*[*x_, y_, w_*] **:=** Mod[*x*+*y*+*w, 2*] *from a configuration Co = "1…1"* (StringLength[*Co*] = *2k; k = 1,2,3, …*) *generates the numerical sequence that not contains prime numbers; whereas from a configuration different from Co the above binary CA model generates the numerical sequence that contains only finite number of prime numbers; more precisely, since some step that depends on initial Co configuration such binary CA model doesn't generate prime numbers. Additionally, the* ***1****–dimensional binary CA with local transition function Ltf*[*x_, y_*] **:=** Mod[*x*+*y, 2*] *from configurations of the form Co = "10…01"* (StringLength[*Co*] **>=***14) generates numerical sequences that not contain prime numbers. While the* ***1****–dimensional binary CA model with local transition function Ltf*[*0,0,0*] = *0, Ltf*[*0,0,1*] = *1, and Ltf*[*x_,y_,w_*]**:=** Mod[*x*+*y*+*w*+*1,2*] *otherwise, generates the infinite sequence of prime numbers from initial Co="11" configuration.*

In addition to the above empirical results, the procedure call ***CFPrimeDensity*[*Co, A, f, n*]** prints the sequence of ***2***–element lists whose the first element defines the number of step of ***1****–CA***,** the second element defines the density of primes on this interval**;** the arguments of this procedure fully complies with formal arguments of the ***CFsequences*** procedure. In the same time, there are also a number of the other results interesting enough in this direction [1–4].

Meanwhile, the problem of self-reproducing finite configurations is one of the main directions of researches in *CA* models. In this direction a number of both theoretical, and experimental results is obtained. So, the next procedure represents a rather certain interest of experimental character. The procedure call ***SelfReprod*[*c, n, p, j*]**, programmed in the ***Mathematica*** returns the number of iterations of a linear global transition function with neighbourhood index ***X*** = {*0, 1, ..., n* – *1*} and alphabet ***A*** = {*0, 1, ..., p* – *1*} *(p – an arbitrary integer)* that was required to generate ***j*** copies of an initial ***c*** configuration. Furthermore, in case of a rather long run of the procedure, it can be interrupted, by monitoring through the list {***d, t***} the reality of obtaining the required number of copies of the ***c*** configuration**,** where ***d*** – number of the iterations and ***t*** – the quantity of initial ***c*** configuration. Whereas ***SelfReprod1*** and ***SubConf*** procedures acts as certain extensions of the above procedure. The procedures ***HS*** and ***HSD*** serve for study of configurations dynamics of the classical ***1****–CA* and ***1****–CA* with delays accordingly [1–5,7].

In the light of research of subconfigurations attainability of in the chains of the configurations generated by the *CA* models the ***CFattainability*** procedure is a rather useful. The procedure call ***CFattainability*[*x, y, A, f, n*]** prints the two–element list whose first element defines the number of step of ***1****–CA* with alphabet ***A*** and local transition function ***f*,** the second element defines the number of ***y*** subconfigurations containing in a configuration generated by the *CA* model from a ***x*** configuration**,** and ***n*** argument determines the steps interval of generating, when the inquiry on continuation or termination of the procedure operating is done *(key Enter – continuation, "No" – termination).* The call in response to *"no"* returns nothing, terminating the procedure, whereas in response to *"other"* a new configuration is requested as a sought ***y*** configuration. At that, value in the answer is coded in string format. The following fragment represents the source code of the ***CFattainability*** procedure with examples of its application.

```
In[3678]:= CFattainability[x_ /; StringQ[x] && x != "" || IntegerQ[x] && x != 0,
  y_ /; StringQ[y] && y != "" || IntegerQ[y], A_ /; ListQ[A] &&
  MemberQ[Map[Range[0, #] &, Range[9]], A], f_ /; ListQ[f] &&
  AllTrue[Map[RuleQ[#] &, f], TrueQ] || FunctionQ[f], n_ /; IntegerQ[n] && n > 1] :=
   Module[{a, b, d, d1, c, tf, p = 0, h = ToString[y]},
     If[FunctionQ[f], b = Arity[f], b = StringLength[f[[1]][[1]]];
     tf = Map[ToExpression[Characters[#[[1]]]] –> ToExpression[#[[2]]] &, f]];
      c = StringMultiple2["0", b]; Label[New]; a = StringTrim2[ToString[x], "0", 3];
While[p < Infinity, p++; a = c <> a <> c; a = Partition[ToExpression[Characters[a]], b, 1];
     a = If[FunctionQ[f], Map[f @@ # &, a], ReplaceAll[a, tf]];
     a = StringJoin[Map[ToString, a]]; d = StringTrim2[a, "0", 3];
    If[Set[d, StringCount[d, h]] >= 1, Print[{p, d}]; Break[],
     If[Mod[p, n] == 0, Print[p]; d1 = Input["Continue?"];
       If[SameQ[d1, Null], Continue[], If[d1 === "no", Break[], If[d1 === "other",
    d1 = Input["A new configuration in string format"];
     p = 0; h = d1; Goto[New], Break[]]], Null]]]]];]
In[3679]:= f[x_, y_] := Mod[x + y, 2]; CFattainability[1, 11000011, {0, 1}, f, 100]
{11, 1}
In[3680]:= g[x_, y_, z_] := Mod[x + y + z, 2]; CFattainability[1101, 1100011, {0, 1}, g, 30]
30
Continue? – Enter
60
Continue? – Enter
90
Continue? – "other"
"A new confiduration in string format"
"11011101000011010000110111010000000000000000000000000000000000000110111010000
1101000011011101"
30
Continue? – Enter
60
Continue? – Enter
{76, 1}
```

For convenience of tracking of quantity of the steps generated by means of the *CA* model the numbers of steps, multiple to value of the ***n*** argument are printed. This information is used for decision–making concerning the further choice of a way of operating with the procedure.

Among research problems of dynamics of sequences of the configurations, generated by the *CA* models, the research of diversity of subconfigurations composing the generated configurations plays a rather essential part. In the same time, this problem in theoretical plan is a rather complex therefore the computer research is used enough widely. The ***SubCFdiversity*** procedure allows to research the diversity for the ***1***–dimensional *CA* models. The call ***SubCFdiversity*[*Co, A, f, n*]** prints the sequence of ***3***-element lists whose the first element defines the number of step of ***1-CA*** model, the second element defines the quantity of various subconfigurations which are contained in a configuration generated by means of the *CA* with ***A*** alphabet and local transition function ***f*** from initial ***Co*** configuration on this step whereas the ***3rd*** element defines the quantity of various non–overlaping subconfigurations. At last, the fourth ***n*** argument determines quantity of steps of *CA* model on which research of the specified phenomenon is made. At that, the first argument admits both the string, and numeric format**,** ***A*** = *{0, 1, …, p} (p = 1..9)* whereas a list of rules, or function can be as third argument. The fragment below represents source code of the procedure ***SubCFdiversity*** along with examples of its application.

```
In[3474]:= SubCFdiversity[Co_ /; StringQ[Co] && Co != "" || IntegerQ[Co] && Co != 0,
   A_ /; ListQ[A] && MemberQ[Map[Range[0, #] &, Range[9]], A], f_ /; ListQ[f] &&
   AllTrue[Map[RuleQ[#] &, f], TrueQ] || FunctionQ[f], n_ /; IntegerQ[n] && n >= 0] :=
   Module[{a = StringTrim2[ToString[Co], "0", 3], b, d, d1, d2, c, t1 = {}, t2 = {},
      t3 = {}, t4 = {}, tf, p = n}, If[! MemberQ3[Map[ToString, A], Characters[a]] ||
  ! MemberQ3[A, IntegerDigits[ToExpression[Co]]], Print["Initial configuration <" <>
  ToString[Co] <> "> is incorrect"]; $Failed, If[FunctionQ[f], b = Arity[f],
  Map[{{AppendTo[t1, StringQ[#[[1]]]], AppendTo[t2, StringLength[#[[1]]]]},
 {AppendTo[t3, StringQ[#[[2]]]], AppendTo[t4, StringLength[#[[2]]]]}} &, f];
     b = Map[DeleteDuplicates[#] &, {t1, t2, t3, t4}];
If[! (MemberQ3[{True}, {b[[1]], b[[3]]}] && Map[Length, {b[[2]], b[[4]]}] == {1, 1} &&
Length[t2] == Length[A]^(b = b[[2]][[1]])), Print["Local transition function is incorrect"];
Return[$Failed], tf=Map[ToExpression[Characters[#[[1]]]]–>ToExpression[#[[2]]] &, f]]];
     c = StringMultiple2["0", b]; Print[Co]; While[p > 0, p—; a = c <> a <> c;
     a = Partition[ToExpression[Characters[a]], b, 1];
     a = If[FunctionQ[f], Map[f @@ # &, a], ReplaceAll[a, tf]];
     a = StringJoin[Map[ToString, a]]; d = StringTrim2[a, "0", 3];
d1=Length[DeleteDuplicates[Map[StringJoin, Flatten[Map[Partition[Characters[d],#,1] &,
Range[2, StringLength[d]]], 1]]]];
d2=Length[DeleteDuplicates[Map[StringJoin, Flatten[Map[Partition[Characters[d],#,#] &,
Range[2, StringLength[d]]], 1]]]]; Print[{n – p, d1, d2}]];]]
In[3475]:= SubCFdiversity[11, {0, 1}, {"000" –> "0", "001" –> "1", "010" –> "1",
"011" –> "0", "100" –> "1", "101" –> "0", "110" –> "0", "111" –> "1"}, 5]
11
{1, 6, 4}
======
{3, 22, 11}
{4, 30, 13}
{5, 38, 17}
```

The ***SelfReproduction*** procedure serves for study of self–reproducibility problem in classical ***1–****CA* models [1-5]. The first three arguments {***Co,A,f***} of the procedure are fully equivalent to the above ***CFsequenses*** procedure**,** whereas ***n*** argument defines the demanded number of copies of a ***Co*** configuration in configurations that are generated by the *CA* model, and ***m*** argument defines an interval of the generating when the inquiry on continuation or termination of operating with the procedure is done *(key "Enter" – continuation, No – exit).* The call ***SelfReproduction*[*Co, A, f, n, m*]** returns the ***2***–element list whose the first element determines the number of the *CA* step on which the demanded number of ***Co*** copies has been obtained, while the second element defines the really obtained number of ***Co*** copies. The procedure call in response to ***"****No****"*** returns nothing, terminating the procedure.

The tools represented here and in [5,7] and intended for computer research of dynamic properties of the ***1***–dimensional *CA* models, are of interest not only especially for specific applications of the given type but many of them can be successfully used as auxiliary tools at programming in the ***Mathematica*** system of other problems of the computer research of the ***1***–dimensional *CA* models. The complex of procedures and functions developed by us which are focused both on the computer research of *CA* models, and on expansion of the ***Mathematica*** software are located in the ***MathToolBox*** package which contains more than ***1140*** tools with freeware license. This package can be freely downloaded from web–site [5]. The package is represented in the form of the archive included five files of formats {*cdf, m, mx, nb, txt*} which, excepting *mx*–format, can be used on all known computing platforms.

Meanwhile, the procedures presented in the article are intended for the computer research *CA* of models focused on ***1***–dimensional models. Experience of use of similar means for a case of two-dimensional *CA* models has revealed expediency of use for these purposes of the ***Maple*** system, but not the ***Mathematica*** system. The main reason for it consists that performance of the nested cyclic structures in the ***Maple*** is essential more fast than in the ***Mathematica.*** For these purposes it is the most expedient to use the parallel systems of information processing focused on *CA*–like computing architectures [1-4].

At last, we will make one essential remark concerning of place of the *CA* problems in scientific structure. By a certain contraposition to the standpoint on the *CA*–problematics that is declared by the book [6] our vision of this question is being represented as follows. Our experience of researches in the *CA*–problematics both on theoretic, and applied level speaks entirely another:

(1) *CA*–models represent one of special classes of infinite abstract automata with the specifical internal organization which provides extremely high–parallel level of the information processing and calculations; these models form a specific class of discrete dynamic systems that function in especially parallel way on base of a principle of local short–range interaction;

(2) *CA*–models can serve as a quite satisfactory model of high–parallel calculations just as the Turing machines *(Markov normal algorithms, Post machines, productions systems, etc.)* serve as the formal models of sequential calculations; from this standpoint the *CA*–models it is possible to consider and as algebraical systems of processing of finite or/and infinite words, defined in finite alphabets, on basis of a finite set of rules of parallel substitutions; in particular, a *CA*–model can be interpreted as a some system of parallel programming where the rules of parallel substitutions act as a parallel language of the lowest level;

(3) principle of local interaction of elementary automata composing a *CA*–model which in result defines their global dynamics allows to use the *CA* and as a fine environment of modelling of a rather broad range of processes, phenomena and objects; furthermore, such phenomenon as the reversibility permitted by the *CA* does their by very interesting tools for physical modelling, and for creation of very perspective computing structures basing on the nanotechnologies;

(4) at last, the *CA*–models represent an interesting enough independent mathematic object whose essence consists in high–parallel processing of words in finite or infinite alphabets.

At that, it is possible to associate the *CA*–oriented approach with certain model analogue of the differential equations in partial derivatives describing those or another processes with that the difference, that if the differential equations describe a process at the average, then in a *CA*–model defined in appropriate way, a certain researched process is really embedded and dynamics of the *CA*–model enough evidently represents the qualitative behaviour of researched process. Thus, it is necessary to define for elementary automata of the model the necessary properties and rules of their local interaction by appropriate way. The *CA*–approach can be used and for research of the processes described by complex differential equations which have not of analytical solution, and for processes, that it is not possible to describe by such equations. Furthermore, the *CA* models represent a rather perspective modelling environment for research of those phenomena, objects, processes, phenomena for that there are no known classical means or they are difficult enough.

As we already noted, as against many other modern fields of science, the theoretical component of *CA*-problematics is no so appreciably crossed with its second applied component, therefore, it is possible to consider *CA*–problematics as two independent enough directions: ***(1)*** research of the *CA* as mathematical objects and ***(2)*** use of the *CA* for simulating; at that, the second direction is characterized also by the wider spectrum. The level of development of the second direction is appreciably being defined by possibilities of the modern computing systems since *CA*–models, as a rule, are being designed on base of the immense number of elementary automata and, as a rule, with complex enough rules of local interaction among themselves. The indubitable interest to them amplifies also a possibility of realization of high–parallel computing *CA*–models on the basis of modern successes of microelectronics and prospects of the information processing at the molecular level *(methods of nanotechnology);* while the itself *CA*–concept provides creation of both conceptual and practical models of spatially–distributed dynamic systems of which namely physical systems are the most interesting and perspective. Namely, from the given standpoint the *CA*–models of various type represent a special interest, above all, from the applied standpoint at research of a lot of processes, phenomena, objects in different fields and, first of all, in physics, computer science and development biology. As a whole if classical *CA*–models represent first of all the formal mathematical systems researched in the appropriate context, then their numerous generalizations represent a perspective modelling environment of various processes and objects.

References

1. Аладьев В.З., Бойко В.К., Ровба Е.А. Классические клеточные автоматы: Теория и приложения.– Беларусь: Гродно: Изд–во Гродненского госуниверситета, 2008, 485 p. ISBN 978–985–551–020–7, ISBN 978–9985–9508–4–5.

2. Aladjev V.Z. Classical Cellular Automata. Homogeneous Structures.– USA, CA: Palo Alto, Fultus Corporation, 2010, 480 p., ISBN 159–682–222–8

3. Aladjev V.Z., Grinn D.S., Vaganov V.A. Classical Homogeneous Structures: Mathematical Theory and Applications.– Kherson: Oldi–Plus Press, 2014, ISBN 9789662890358, 520 p.

4. Aladjev V.Z. Classical Cellular Automata: Mathematical Theory and Applications.– Germany: Saarbrücken: Scholar`s Press, 2014, ISBN 9783639713459, 517 p.

5. Aladjev V.Z. A package of procedures and functions for system ***Mathematica.–*** Tallinn, 2017; https://yadi.sk/d/6qbBq5Fn3LHkLa

6. Wolfram S. A New Kind of Science.– N.Y.: Wolfram Media, 2002, ISBN 978–1579550080

7. Aladjev V.Z., Shishakov M.L. Software Etudes in the ***Mathematica*** system. Tallinn Research Group**.-** USA: CreateSpace Press, An Amazon.com Company, 2017, ISBN-13: 978-1979037273, ISBN–10: 1979037272, 614 p.