Proceedings of CECNet 2021
A.J. Tallón-Ballesteros (Ed.)

doi:10.3233/FAIA210464

Modeling Ebola Virus Dynamics by Colored Petri Nets

Dmitry A. ZAITSEV[a,1] and Peyman GHAFFARI[b] and Virginia Sanz SANCHEZ[c]

[a] *University of Information Technology and Management in Rzeszów, Poland*
[b] *Department of Mathematics, University of Aveiro, Portugal*
[c] *Faculty of Pharmaceutical Sciences, University of Iceland, Iceland*

Abstract. We develop techniques to translate deterministic cellular automata models into colored Petri nets based on an example of known cellular automata for modeling and mimicking Ebola virus dynamics. Cellular automata for Ebola virus dynamics use parametric specifications of rules that is peculiarity of the present study. The simulation results completely coincide with known results obtained via dedicated simulator. Having uniform language for models specification brings in advantages for models mutual transformations and simulation.

Keywords. Ebola virus, immune response, cellular automata, colored Petri net, simulation

1. Introduction

Ebola is one of the most lethal infectious diseases known to mankind. The Ebola virus causes an acute, often fatal hemorrhagic illness. The incubation time is something between two days and three weeks after becoming infected with the virus [1].

Cellular automata (CA) are widely applied for modeling processes of virus spreading both on micro-level of cells of some organ [2] to macro-level of geo-spatial representation [3].

A Petri net is a well-known formalism for specification of concurrent processes [4]. A family of Petri net classes offers a uniform language for modeling in various domains with possibility of applying both model-checking and simulation techniques of analysis.

In the present paper, we develop a technique for transformation of cellular automaton models for virus spreading into colored Petri net (CPN) models using a known cellular automaton that specifies Ebola virus dynamics [2].

2. Cellular automata and Petri nets

A cellular automaton [4] represents a homogenous structure of elements (cells) having rather simple behavior such that their collective behavior represents a mass parallel

[1] Corresponding Author, Dmitry A. Zaitsev, Information Technology Department, University of Information Technology and Management in Rzeszów, ul. Sucharskiego 2, 35-225 Rzeszow, Poland; E-mail: daze@acm.org.

process capable of modeling processes in many application domains. A state of cell in the next tact of time depends on the cell current state and states of its neighbors. Usually, an infinite square lattice of cells in multidimensional space is considered in theoretical research though for practical applications, a finite part of the lattice is simulated. Even one dimensional elementary (having binary state) cellular automata are Turing-complete [5] that illustrates power of the concept. Traditionally, von Neumann (sparse) and Moore (dense) neighborhoods of a cell are considered though recently a generalized neighborhood, that fills the gap between two mentioned neighborhoods within multidimensional lattices, has been introduced and studied [6], [7].

A Petri net [8] represents a bipartite directed graph with a dynamic process introduced on it. The first part of vertices, depicted as circles or ovals and called places, represents conditions. The second part of vertices, depicted as rectangles or bars and called transitions, represents events. Dynamic elements, called tokens and depicted as dots, are situated within places; they are consumed and produced as the results of firing transitions. Within a conventional Petri net, tokens are elementary; the net state is specified by the place marking represented by a vector which components contain the number of tokens in the corresponding places. A conventional Petri net is applied for modeling concurrent processes within a wide variety of application domains. Comparable simplicity of Petri net concept allows us to apply analytical methods for model checking [6] though its descriptive power is not too high for detailed specification of systems and processes.

A colored Petri net [9], [10] represents a combination of a Petri net graph and a functional programming language ML. Tokens are considered as elements of data types traditionally called color sets. Arcs and transitions are inscribed with ML constructs. Besides, time characteristics are introduced for the processes simulation. Supplied with a range of random distribution functions, CPN Tools [10] allows us to provide statistical analysis of timed and other characteristics. For composition of complex models, hierarchical design is provided via the transition substitution when a transition is substituted by a subnet. CPNs find wide application for the performance evaluation of systems, especially networks, grids, and clouds [11]. For this purpose, classical simulation technique is applied refined with special measuring components of models to compute statistical characteristics on-the-fly directly in the process of simulation [9]. For comparably small models, application of state space technique for the model properties analysis is possible.

3. Cellular automata models of Ebola virus dynamics

We use Ebola virus dynamics model [2]. A two dimensional CA with the cell states within the set {0,1,2,3} is considered where 0 represents a healthy state, 1 and 2 represent an infected cell, and 3 represents a depleted (dead) cell. Here we present a model for one of three rules studied in [2] that explains basic principles of the ransformation.

Let us consider the mentioned rules description where asterisk symbol "*" denotes any state and a matrix represents states of a cell (in the center) and its neighbors according to Moore neighborhood [6]. Before a matrix, a rule symbol is written; the rule application results in a new state of the central cell that is written after the equality symbol.

CA Q with rule q specification follows:

$$q\begin{pmatrix} * & * & * \\ * & 0 & * \\ * & * & * \end{pmatrix} = \begin{cases} 1, \text{at least one} * \text{is 1 or 2} \\ 0, \text{otherwise} \end{cases}$$

$$q\begin{pmatrix} * & * & * \\ * & a & * \\ * & * & * \end{pmatrix} = 3, \text{for } 1 \leq a \leq 2$$

$$q\begin{pmatrix} * & * & * \\ * & 3 & * \\ * & * & * \end{pmatrix} = 0.$$

The three above lines specify all possible four states of a cell within the set {0,1,2,3}: first line – 0, second line – 1 or 2, and third line – 3. Note that, in second and third line (for the current cell state 1, 2, or 3), the next state does not depend on neighbors. The first line can be considered as a variant of totalistic rule with check that the sum of all neighbor cell states is equal to or greater than 1.

CA Q specifies fast propagation of decease when, according to the first line of its description, at least one deceased cell within Moore neighborhood causes the current cell got infected. Together with other rules studied in [2], it serves for the subsequent composition of stochastic automata that is beyond the scope of the present paper.

4. CPN model of a cell

We compose colored Petri net model of a cell functioning under rule q shown in Fig. 1. In colored Petri net (CPN) of modeling system CPN Tools [10], [11] places are depicted as ovals; each place has its name, written inside a place, data type of tokens (color set), and initial marking. A token is not elementary in CPN; in the general case, it is an element of abstract data types such as cortege, record, list, or union composed of elementary types, such as integer or real numbers, strings and others. In the present work, we use only color sets: an integer number (INT), and a timed integer number (tint). Basic declarations look rather simple:

```
colset tint = INT timed;
var i,ii,j1,j2,j3,j4,j5,j6,j7,j8:INT;
var c:tint;
```

The reserved word "closet" serves for defining new sets of colors while the reserved word "var" serves for declaration of variables. A color set is specified using the equality sign "=", a new color set tint is defined on basic integer color set INT using modifier "timed". Definition of variables consists of a list of names separated by semicolon ":" symbol with the name of color set.

Within our model, we have only no more than one token inside each place though in the general case the place marking is represented by a multiset composed using double plus sign "++". Recall that, each element belongs to a multiset in definite number of copies (multiplicity), the multiset element is represented as k`e, where k denotes multiplicity and e denotes an element. On the initial marking CPN Tools creates current marking of simulation system which is displayed in green colors and changes as result of transitions firing.

Timed delays are associated either with transitions or with separate arcs. For timed multisets, an element is represented as k`e@t, where t is the token activation time. A transition fires instantaneously, increasing the step counter Step value, a delayed token spends time before its activation in the corresponding output place of transition in passive state not possible for enabling any transition. The timed delays are specified as k`e@+d, where d species the delay. Here we use integer numbers to specify delay though in the general case, a random distribution function can be applied, CPN Tools providing a series of random distribution functions including Gauss, Poisson and others.

In our models, we use a timed CPN to specify synchronous work of all the cells. The marking change within a lattice is implemented in two tacts. At the first tact, all the neighbors and the current cell itself can read the current “0,0” cell state using two directed arcs without change of the marking. During this tact, a new state of the current cell is computed and stored within additional place “new 0,0”. Then, at the second tact, the current state of all cells is instantaneously replaced by their new states. For this purpose, we have a clock token 1@t moving within a circle created by places clock1 and clock2 and transitions which connect the mentioned places. The clock extracted by variable c inscribed on the arcs from place clock1 activates one of the four alternative transitions having names “0”, “1”, “2”, and “3”, corresponding to the current “0,0” cell state. The corresponding alternative values of the current state are inscribed on the transitions incoming arcs. We use double directed arcs though one of the mentioned transition fire only once because of the clock arc; after firing a transition, the clock token is delayed by “1@+1” to give time for all cell to compute their new state.

The automaton Q specification in the Section 2, contains simple subrules for the cell states equal to 1, 2, and 3: transform 1 or 2 into 3 and transform 3 into 0 without taking into consideration state of the neighboring cells. Each of these transformations is specified by a corresponding transition, putting the new current state value into place “new 0,0”. For the current cell state equal to 0, the automaton Q, according to its first subrule, checks states of its neighbors. There are two alternatives of this check represented by transitions “1 or 2” and “all 0 or 3”. Now let us pay attention to the naming of cells within the cell model. Because the model should suit to each cell of the lattice, we do not use the cell index; instead of it we use neighbor offsets with regard to the current state index. Thus, the current cell is named as “0,0”, the left upper cell is named as “-1,-1” and so on because we use a matrix-like orientation of coordinate axes. To extract values of the neighboring cells, we use a set of integer variables j1, j2, j3, j4, j5, j6, j7, j8 corresponding to the clockwise enumeration of neighbors starting from the upper neighbor. Note that, to avoid a tangled picture with manifold crossed lines, we implement connections of either of transitions “1 or 2” and “all 0 or 3” with neighboring cells as a bus merging all arcs in a single line; the bus encircles the cell shape with endings of arcs coming to the neighbor cells states.

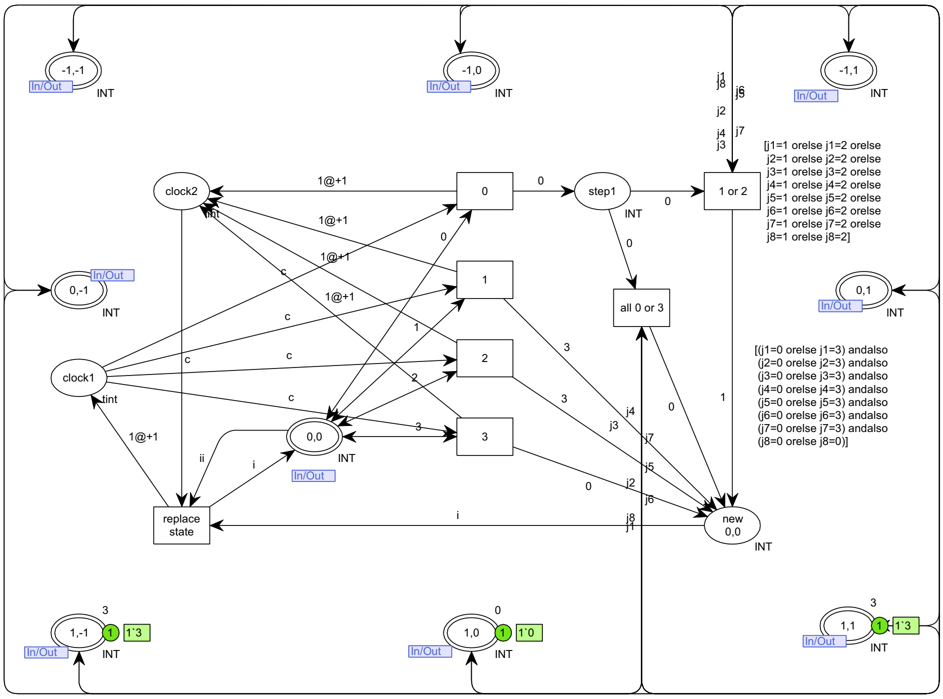

-1,-1
In/Out
INT
-1,0
-1,1
0,-1
0,1
0,0
1,-1
1,0
1,1
1`3
1`0
clock1
clock2
step1
new 0,0
replace state
0
1
2
3
1 or 2
all 0 or 3
1@+1
tint
c
i
ii
j1 j2 j3 j4 j5 j6 j7 j8
[j1=1 orelse j1=2 orelse j2=1 orelse j2=2 orelse j3=1 orelse j3=2 orelse j4=1 orelse j4=2 orelse j5=1 orelse j5=2 orelse j6=1 orelse j6=2 orelse j7=1 orelse j7=2 orelse j8=1 orelse j8=2]
[(j1=0 orelse j1=3) andalso (j2=0 orelse j2=3) andalso (j3=0 orelse j3=3) andalso (j4=0 orelse j4=3) andalso (j5=0 orelse j5=3) andalso (j6=0 orelse j6=3) andalso (j7=0 orelse j7=3) andalso (j8=0 orelse j8=0)]

Figure 1. Model of cell with rule *q*.

To check the state of neighbors, we employ a guard function of a transition. The guard function represents a logic expression that combines check of characteristics for tokens extracted by various incoming arcs of a transition. The logic expression is written in square brackets using the following notations: “orelse” denotes disjunction and “andalso” denotes conjunction. Thus we interprete the first variant of the first subrule for automaton Q, implemented by transition “1 or 2” as

$$(j_1 == 1)\vee(j_1 == 2)\vee(j_2 == 1)\vee(j_2 == 2)\vee(j_3 == 1)\vee(j_3 == 2)\vee(j_4 == 1)\vee(j_4 == 2)\vee(j_5 == 1)\vee(j_5 == 2)\vee(j_6 == 1)\vee(j_6 == 2)\vee(j_7 == 1)\vee(j_7 == 2)\vee(j_8 == 1)\vee(j_8 == 2).$$

We interpret the second variant of the first subrule for automaton Q as the above expression negation with regard to the state value range, implemented by transition “all 0 or 3” as

$$((j_1 == 0)\vee(j_1 == 3)) \wedge ((j_2 == 0)\vee(j_2 == 3)) \wedge (j_3 == 0)\vee(j_3 == 3) \wedge (j_4 == 0)\vee(j_4 == 3) \wedge (j_5 == 0)\vee(j_5 == 3) \wedge (j_6 == 0)\vee(j_6 == 3) \wedge (j_7 == 0)\vee(j_7 == 3) \wedge (j_8 == 0)\vee(j_8 == 3).$$

For the further composition of a lattice using the designed cell model, places corresponding to the current and neighboring cell states are labeled by the input/output

(In/Out) port tag drawn in blue color. They represent parameters of the model and will be studied in detail when composing a lattice.

5. Composing CPN model of a lattice

To compose a lattice of CA Q, we create a separate place to store the corresponding cell state, we use cell indices as names, for instance “0,0”, “0,1” etc. With each cell we associate a cell model represented by a separate transition, to which name prefix “c” is added, to obtain the main page of CA Q model, shown in Fig. 2. The arcs reflect connections with neighboring cells; for Moore neighborhood applied in the present model connections look rather dense. Note that, because the current cell state is not actually localized within the cell model, we have actually 9 connections for each cell represented by bidirected arcs. Neighbors only read the current cell state values while the current cell model changes it.

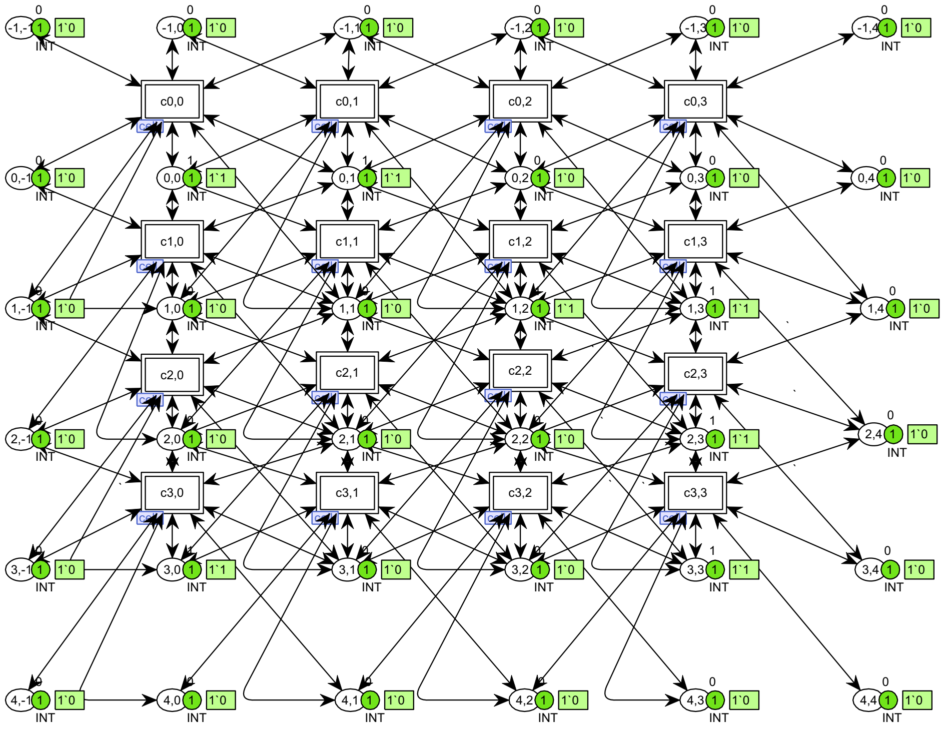


Figure 2. Model of CA *Q* (with cell model in Fig. 1), finite 4x4 lattice with constant border conditions.

As a mathematical model, CAs are thought of as infinite lattices and this definition is applied for deriving analytical results with regard to CA properties. For computer simulations, we always use a finite part of lattice obtained according to given size of CA, usually square for a two dimensional automaton. Thus, a question regarding the borders of finite lattice arise because if we simply cut a square finite lattice, cells on the borders will use indices of nonexistent cells to implement their rules. The simplest solution, implemented in our model of lattice (Fig. 2), consists in adding places storing states of neighbors for one more layer of cells on each border; it corresponds to constant border conditions, though individual for each border cell. Thus in Fig. 2, actually cell state of 5x5 lattice is stored while cell state of 4x4 lattice is modified according to the rule q. Note that we can implement other border conditions, for instance, connect opposite edges to obtain a torus [12] or use additional subnets which implement varying border conditions.

Now let us consider in detail how hierarchical models are composed within CPN Tools. The basic operation is substitution of a transition by a subnet. In Fig. 2, each transition, which implements modification of the corresponding cell state, is tagged by the transition substitution tag "cell". It means that the subnet shown in Fig. 1 works inside each transition with specific data according to the cell neighborhood state. To use parameters while substituting a transition, socket-port mapping is provided by CPN Tools [11]. Places of a submodel, which map on the corresponding places of a model, are called ports and indicated by port tags, for instance In/Out tags in Fig. 1. Ports are mapped to places of the model, which are called sockets, using special tools of Hierarchy palette of CPN Tools. It is supposed that transition substitution and port-socket mapping are implemented manually.

When the CA Q model is composed, we can simulate CA dynamics for various input state of lattice and border conditions. For this purpose CPN Tools offers step-by-step mode of simulation for model debugging. Then simulation of specified number of steps can be launched. It is convenient to use time break-points to check the lattice state when all the cells implemented change of state at the current timed tact.

6. Analysis of simulation results

Specifying here Ebola virus dynamics model in CPN language, we apply mainly simulation to study the model behavior and properties. Besides, though it is feasible for rather small models without using high performance resources, we can build complete state space of model, obtain automatic report on it containing conclusions of such basic properties as boundedness, liveness, and fairness [7], [11]. There is an additional possibility to compose requests to state space for finding specific properties which can be interesting, for instance, when modeling viruses. For 1x1 CA Q shown in Fig. 3, CPN Tools built state space drawn in Fig. 4.

In Fig. 3, we renamed vertices because state space palette of tools has more strict requirements to syntax: we replaced zero by "O" and one by "I" and sign minis by "m" to have names starting from a letter and containing no special characters. Thus, we have a cell with state 1 that has 7 neighbors with state 0 and one neighbor with state 1. Built state space of CA Q 1x1 (Fig. 3) represented in Fig. 4 contains 9 states; after an initial part of first two transitions, the models falls in a loop of states from 3 to 9. Initially drawn state space does not reflect detailed descriptions of states and transitions; firing a transition is represented by an arc connecting state, a state is specified by its number and the number of incoming and outgoing arcs. Fig. 4 contains attributes of all transitions expanded, as well as expanded state description for state 7. Transition specification indicates the transition name and values of tokens which were taken at transition firing (in curly brackets). Compound names of vertices are written including a prefix with the subnet name followed by, separated by quotation symbol, the node name. State specification lists marking of all transitions. The obtained report concludes that the net is bounded, its initial marking is a home marking, dead markings are absent though some instances of transitions are dead, for example "cell'all0or3", and classifies transitions into impartial, fair, just and unfair, no unfair transitions found.

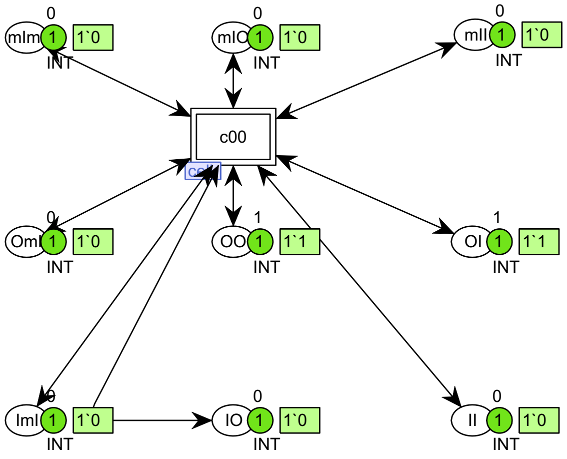
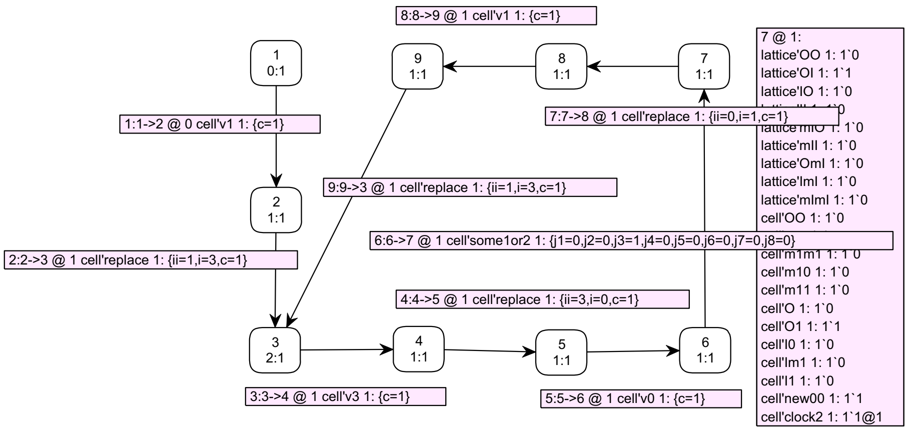


Figure 3. CA Q 1x1.



1 0:1
2 1:1
3 2:1
4 1:1
5 1:1
6 1:1
7 1:1
8 1:1
9 1:1
1:1->2 @ 0 cell'v1 1: {c=1}
2:2->3 @ 1 cell'replace 1: {ii=1,i=3,c=1}
3:3->4 @ 1 cell'v3 1: {c=1}
4:4->5 @ 1 cell'replace 1: {ii=3,i=0,c=1}
5:5->6 @ 1 cell'v0 1: {c=1}
6:6->7 @ 1 cell'some1or2 1: {j1=0,j2=0,j3=1,j4=0,j5=0,j6=0,j7=0,j8=0}
7:7->8 @ 1 cell'replace 1: {ii=0,i=1,c=1}
8:8->9 @ 1 cell'v1 1: {c=1}
9:9->3 @ 1 cell'replace 1: {ii=1,i=3,c=1}
7 @ 1:
lattice'OO 1: 1`0
lattice'OI 1: 1`1
lattice'IO 1: 1`0
1`0
1`0
lattice'mII 1: 1`0
lattice'OmI 1: 1`0
lattice'ImI 1: 1`0
lattice'mImI 1: 1`0
cell'OO 1: 1`0
cell'm10 1: 1`0
cell'm11 1: 1`0
cell'O 1: 1`0
cell'O1 1: 1`1
cell'I0 1: 1`0
cell'Im1 1: 1`0
cell'I1 1: 1`0
cell'new00 1: 1`1
cell'clock2 1: 1`1@1

Figure 4. State space of CA Q 1x1 (Fig. 3).

Thus, the CA Q 1x1 model with the specified initial state, coming from CA state 1, falls into a loop of CA states 3->0->1->3 stored in place "cell'OO". With our model, we observed other variants of cyclic behavior of CA studied in [2] for 3x3 and 7x7 lattices that indirectly acknowledges correctness of our constructs.

For statistical analysis of composed models having big size of the cell matrix, simulation on prolonged intervals of time is applied that also acknowledges conclusions of [2] with regard to parameters choice to provide recovery after the decease spreading. Combining state space and simulation techniques allows us to study the constructed models comprehensively that is an advantage of applying a colored Petri net and modeling system CPN Tools.

7. Conclusions

In this paper we presenter technique to transform so called SEIR models used in mathematical Epidemiology into colored Petri net models. Applying CPN models for specifying both behavior patterns of individual cells and virus dynamics within real-life scale organs results in a uniform concept for model-checking and simulation.

Acknowledgement

We would like to thank the EU COST Association for bringing the authors together through funding events and STSM-grants within the framework of IMAAC (COST Action CA 16227).

References

[1] Rachah, Amira; Torres, Delfim F. M. Dynamics and Optimal Control of Ebola Transmission, Mathematics in Computer Science: Vol. 10, pp. 331-342 (2016).
[2] Emily Burkhead, Jane Hawkins, A cellular automata model of Ebola virus dynamics, Physica A: Statistical Mechanics and its Applications, Volume 438, 2015, Pages 424-435.
[3] Gerardo Ortigoza, Fred Brauer, Iris Neri Modelling and simulating Chikungunya spread with anunstructured triangular cellular automata, Infectious Disease Modelling 5 (2020) 197-220.
[4] Jarkko Kari, Theory of cellular automata: A survey, Theoretical Computer Science, 334(1–3), 2005, 3-33.
[5] Matthew Cook, Universality in Elementary Cellular Automata, Complex Systems, 15(1), 2004, 1–40.
[6] Zaitsev D.A. A generalized neighborhood for cellular automata, Theoretical Computer Science, 666 (2017), 21-35.
[7] D. A. Zaitsev, T. R. Shmeleva and P. Ghaffari, "Modeling Multidimensional Communication Lattices with Moore Neighborhood by Infinite Petri Nets," 2021 International Conference on Information and Digital Technologies (IDT), 2021, pp. 171-181, doi: 10.1109/IDT52577.2021.9497552
[8] Reisig, W., Understanding Petri nets. Modeling techniques, analysis methods, case studies, Springer, 2013.
[9] Zaitsev D.A. and Shmeleva T.R. Modeling With Colored Petri Nets: Specification, Verification, and Performance Evaluation of Systems (pp. 378-404) Chapter 14 in T. Shmelova, N. Rizun, D. Kucherov and K. Dergachov (Ed.) Automated Systems in the Aviation and Aerospace Industries. IGI-Global: USA, 2019.
[10] Jensen, Kurt, Kristensen, Lars M. Coloured Petri Nets. Modelling and Validation of Concurrent Systems, Springer, 2009.
[11] Zaitsev D.A. Clans of Petri Nets: Verification of protocols and performance evaluation of networks, LAP LAMBERT Academic Publishing, 2013, 292 p.
[12] Dmitry A. Zaitsev, Tatiana R. Shmeleva & Birgit Proll, Spatial specification of hypertorus interconnect by infinite and reenterable coloured Petri nets, International Journal of Parallel, Emergent and Distributed Systems, 2021, DOI: 10.1080/17445760.2021.1952580