Modeling Immunity in Light of Disaster Management

by Dr. István BUKOVICS

(istvan.bukovics@katved.hu)

A társadalmi csoportosulások is folyamatosan ki vannak téve különféle környezeti hatásoknak. A dolgozatban egy mesterséges immunitás-modellt mutatunk be a korábban kifejlesztett SORS önszervező védelmi rendszerre vonatkozóan[1].

Nem csak az ember, hanem a társadalmi csoportosulások is (az ingatlanoktól a nemzetekig) folyamatosan ki vannak téve különféle környezeti hatások támadásainak, a szó legáltalánosabb értelmében. Ezeket a hatásokat valamilyen mértékben tolerálják, és a legrátermettebb, immunitási képességétől függően, túléli. A dolgozatban egy mesterséges immunitás-modellt mutatunk a korábban kifejlesztett SORS (Angol eredeti mozaikszó!). önszervező védelmi rendszerre vonatkozóan[2]. A SORS megalkotásához standard sejtautomata módszert alkalmaztunk, kombinálva a hibafa módszer logikai (determinisztikus) verziójával (azaz mellőzzünk minden valószínűségi vonatkozást). *In silico* kísérletekkel megmutatjuk, hogy a megfelelő átmeneti szabályok és egyszerű genetikus algoritmusok szükségszerűen valamiféle mesterséges immunitás kialakulásához vezetnek, anélkül, hogy bevezetnénk egyfajta ilyen képességet a sejt modelljébe. Áttekintjük eredetét és tulajdonságait. A "mesterséges immunitásmodell" terminus azt jelenti, hogy nem *leírni* vagy *szimulálni* akarunk valamilyen immunitás-rendszert, hanem inkább *megalkotni* egy olyan normatív rendszert, melynek célja annak felderítése, milyen szabályok illetve feltételek biztosítják egy komplex, mesterséges, adaptív rendszer védekező képességét sikerességét.

Abstract

Not only human body but also social groups (from real estates to nations) are repeatedly attacked by several environmental effects in the most general sense of the word. Attacks are tolerated for awhile and the fittest, depending on its acquired immunity, survives. Here an artificial immune property is demonstrated of an artificial self organizing raiding system (SORS). To construct SORS standard cellular automata techniques are used combined with a logic (or deterministic) version (ie.dispensing with probability notions) of fault tree methodology. It is shown, by performing *in silico* experiments, that suitable transition rules and simple genetic algorithms necessarily entail the emergence of a kind of artificial immunity without explicitly introducing any fitness property into the cells. Its genesis and

[1] SORS-hivatkozás
[2] SORS-hivatkozás

properties are discussed. The term „artificial immunity modeling" means that we do not want to describe or simulate a real immune system but, rather, to construct a normative system is aimed at questioning what rules and other conditions ensure successful defense capability of a complex adaptive artificial system

Introduction

The complex adaptive system, the candidate for possessing (or rather developing) *immunity*, is called **SORS** (Self Organizing Raiding System). It is a cellular automata[3] (cellular space „CellSpace" **CS** for short preferred) consisting of two type of cells called respectively *defender* and *defendee* (defendent) agents (cells). Alternatively, we speak of special cells called „Guards" whose task is to „defend" the other (common) cells. Common cells are interpreted as the *land units* of a *site* (such as eg. a country)
The **CS** is a closed (torus-like) cellular automata with the usual four-nearest-neighbor neighborhood, as on Fig. 4. There are two types of the common cell *states*. A common cell can be either in a „*virtual*" or in a „*real*" state. The *state transition rule* serves two goals. In case of virtual states it ensures the perpetual changes of states resulting in a global state cycle of the **CS**. In case of real states it ensures modeling (or to describe) the *land unit's* (desirable or expedient) behavior under and following an attack and a defense procedure.

Preliminaries

CellSpace

The cell space (**CS**) is characterized[4] by

- a grid of *cells* containing 64 rows and 64 columns,
  each cell having *nStates* states s = 0, 1,…, *nStates* – 1,
  the state of cell **C** at time *t* is denoted by State(**C,** *t*), t integer.
- the *neighborhood* any cell **C** consisting of the four nearest neighbors of **C** being

  N(**C**) = <N1(**C**), N2(**C**), N3(**C**), N4(**C**)>,
  respectively the northern (top),
  the eastern (left),
  the southern (bottom) and
  the western (right) neighbor.

  It is supposed that the neighborhood is independent of time
- the **transition function F(,)** of cell **C** with parameter t is of the form
  F (C, t)= State(**C**, t + 1) = **F**(State(**C**, t), State(**N**(**C**), t))

A cell **C** is generally identified by its place in the grid of **CS** i.e. by the ordered pair

(Row(**C**), Col(**C**))

where Row(**C**), Col(**C**) is the row and to column of the cell **C** on the grid of **CS** respectively. If necessary, we write

(Row(**C,** t), Col(**C,** t))

For the place of cell **C** at time t**.**
Thus we speak of Cell(12, 36) meaning the cell in row = 12 and column = 36.. Accordingly, S(12, 36) = 5 means the state of the cell **Cell**(12, 36) in the *virtual* state = 5 while S(12, 36) = 5! means the state of the cell **Cell**(12, 36) in the *real* state = 5. In the present paper the number of cells nCells is chosen conventionally to be $2^{12} = 4096$

[3] Technical terms related to cellular automaton can be found e.g. in [Wolfram]
[4] This is not the mathematical definition of a cellular automaton. For a detailed formal treatment, see [Riguet]

Guards

In the SORS CellSpace there are two kinds of cells: *common cells* and *guards*. Common cells obey the transition rule (defined by the transition function above).
Guards walk according to the

Guard Walk Algorithm

Guard walk means that a guard at each time step *t* looks around clockwise in its neighborhood (starting at the top neighbor) searching for the „*defendent cell"*. The *defendent cell* **DC(G)** (if exists) of the guard cell **G** is a common cell in maximal real state. If it does not exist then **G** chooses randomly a common cell from its neighborhood. Then, at time $t + 1$ the guard occupies the defendent's place and takes the defendent's state with virtual thread. Otherwise (if there is no defendent cell in its neighborhood) **G** doesn't move.
Formally, a cell **G** is (or rather occupied by) a Guard if its *walk function* is of the form

(RG) $(\mathrm{Row}(\mathbf{G}, t + 1), \mathrm{Col}(\mathbf{G}, t + 1)) = (\mathrm{Row}(\mathbf{G}, t) + \rho), \mathrm{Col}(\mathbf{G}, t) + \sigma))$

where $\rho, \sigma < 1, > 0$, are random variables according to the above „defendent searching" procedure.
The interpretation (or rather the practical realization) of the guard's move, especially across the *border* of the CellSpace, is by no means straightforward. Still we trust it to be feasible.
In the present paper the number of Guards nGuards is chosen conventionally[5] to be $2^8 - 1 = 255$

CellState

The state of a common cell can be changed in two ways: *spontaneous* and *forced.* Spontaneous cell state change occurs according to the *state transition rule*. The next spontaneous state of a common cell is easily calculated by the state transition function.
Forced state change occurs through the attack. The next forced state of a common cell is determined by the state of the risk explicatum assigned to the cell and calculated by the State Calculation Algorithm

[5] Due (computational) technical reason Guard(8) does not exists. (For, in Visual Basic, Chr(8) is the code of Space, while the Chr() function is used to code Guards and the Space for the abscence of guard at a location.)

The cell space

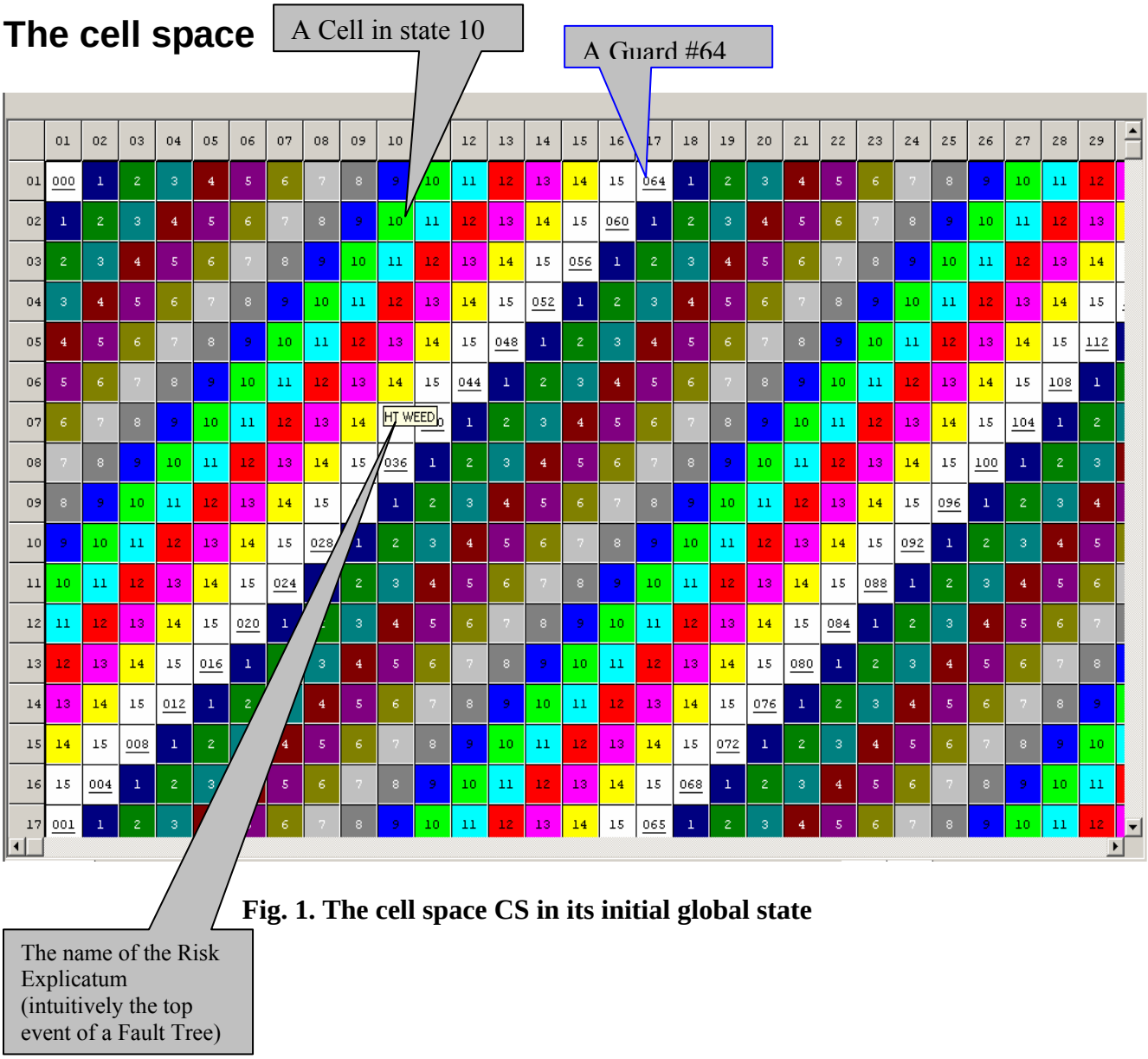


Fig. 1. The cell space CS in its initial global state

The SORS cell space (**CS**) in the present implementation is a systolic closed cellular automaton with an array of 64 x 64 *cells*, each cell having 16 possible states s = 0, 1,…, 15.
To each cell there belongs a „logic" or „deterministic" fault tree as opposed to the adjective „probabilistic". If a traditional fault tree[6] is deprived from its probabilistic features and also from graphical representations using logic gates, one arrives to the formal notion of the Risk Explicatum. Its definition is the following:
By a Risk Explicatum we mean an n-element set of Boolean equations of the following form:

$$E_i = C(E_{i_1},...,E_{i_{m_i}})$$

Here:
Letter **E** means an element – called „event" – of a fixed finite distributive lattice[7] with *m* atoms,
i = 1,…, *n*
m_i = 1,…, *n* with all i_1,…, $i_{mi} > i$ (1)
C is either a conjunction or a disjunction of m_i variables.

[6] For an introduction of traditional fault tree methodology, see e.g. [Henley]
[7] Loosely speaking a Boolean algebra without negation.

$\mathbf{E}_i$ is said to have the logic type „A" („AND") or „V (OR, „Vel")" if it is a conjunction or disjunction respectively.
Events occurring on the right hand sides are called *explicants* of the event of the left hand side.
Events occurring *only* on the right hand sides are called *primitive* events (primevents, prime explicants or just primes for short) and denoted by p.
Events that are not primes are sometimes called *complex* or *composite* (events).

Example[8]:
$n = 39$, $m = 22$ (writing Ei instead of $\mathbf{E}_i$, pi instead of p_i), using „+" and „x" for disjunction and conjunction respectively.

E1 = E2 + E3 + E4	E2 = E5 x E6	E3 = E7 x E8 x E9
E4 = E10 x E11	E6 = E14 x E15	E7 = E12 x E13 x E16
E8 = E17 x E18	E9 = E19 x E20	E10 = E31 + E32
E11 = E33 x E34	E18 = E23 + E24	E20 = E21 + E22
E24 = E25 + E26	E25 = E27 + E28	E26 = E29 + E30
E34 = E35 x E36 x E37	E35 = E38 x E39	

The prime events:

p1 = E5	p2 = E12	p3 = E13	p4 = E14
p5 = E15	p6 = E16	p7 = E17	p8 = E19
p9 = E21	p10 = E22	p11 = E23	p12 = E27
p13 = E28	p14 = E29	p15 = E30	p16 = E31
p17 = E32	p18 = E33	p19 = E36	p20 = E37
p21 = E38	p22 = E39		

As for the representation of the fault tree we prefer the outline view of Microsoft Windows® Word instead of the clumsy an obsolete graphical representation using logic gates. See Fig. …

[8] The example is taken with permission from [Kortenhaus ea]. The names of the event is in the Appendix

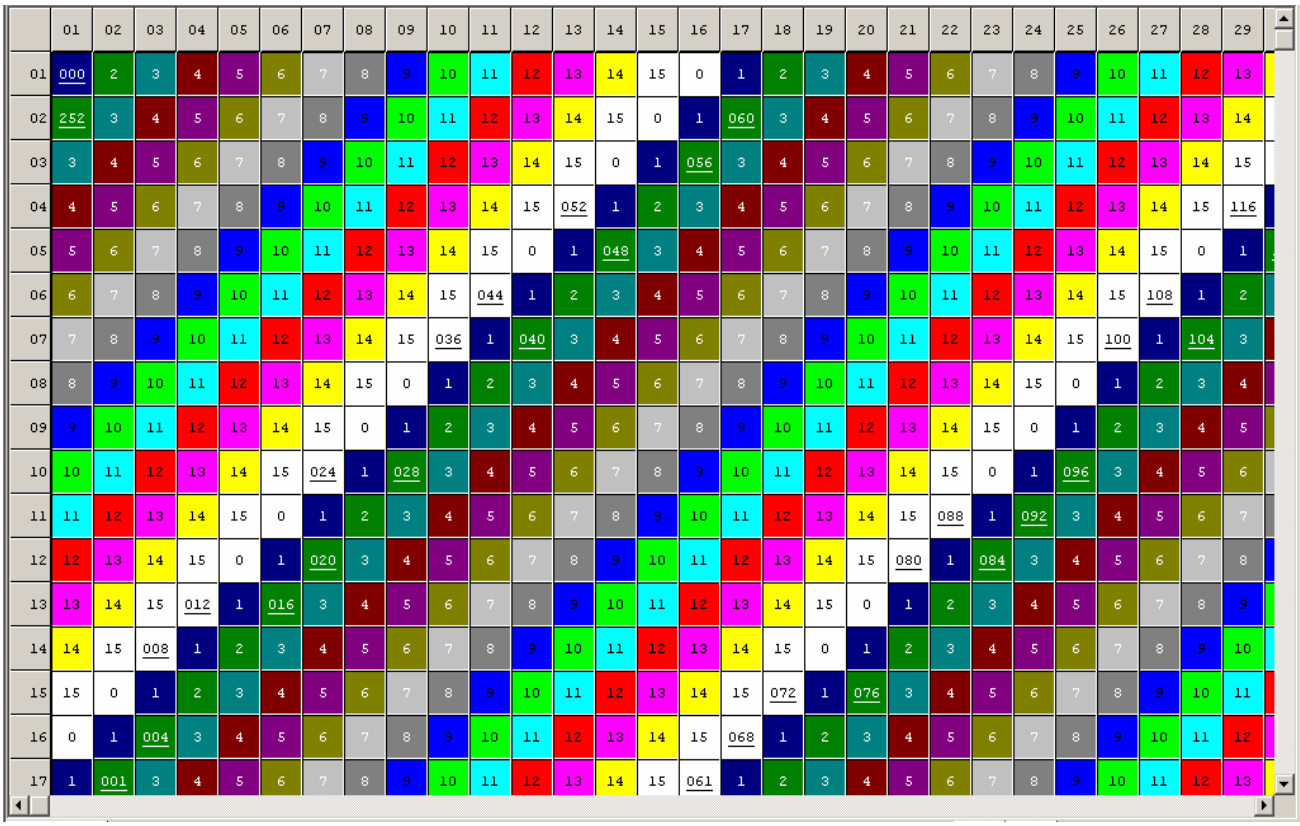


Fig. 2. The Cell Space after the first step: the guards (with underlined numbers) finished with one step in their random walk.

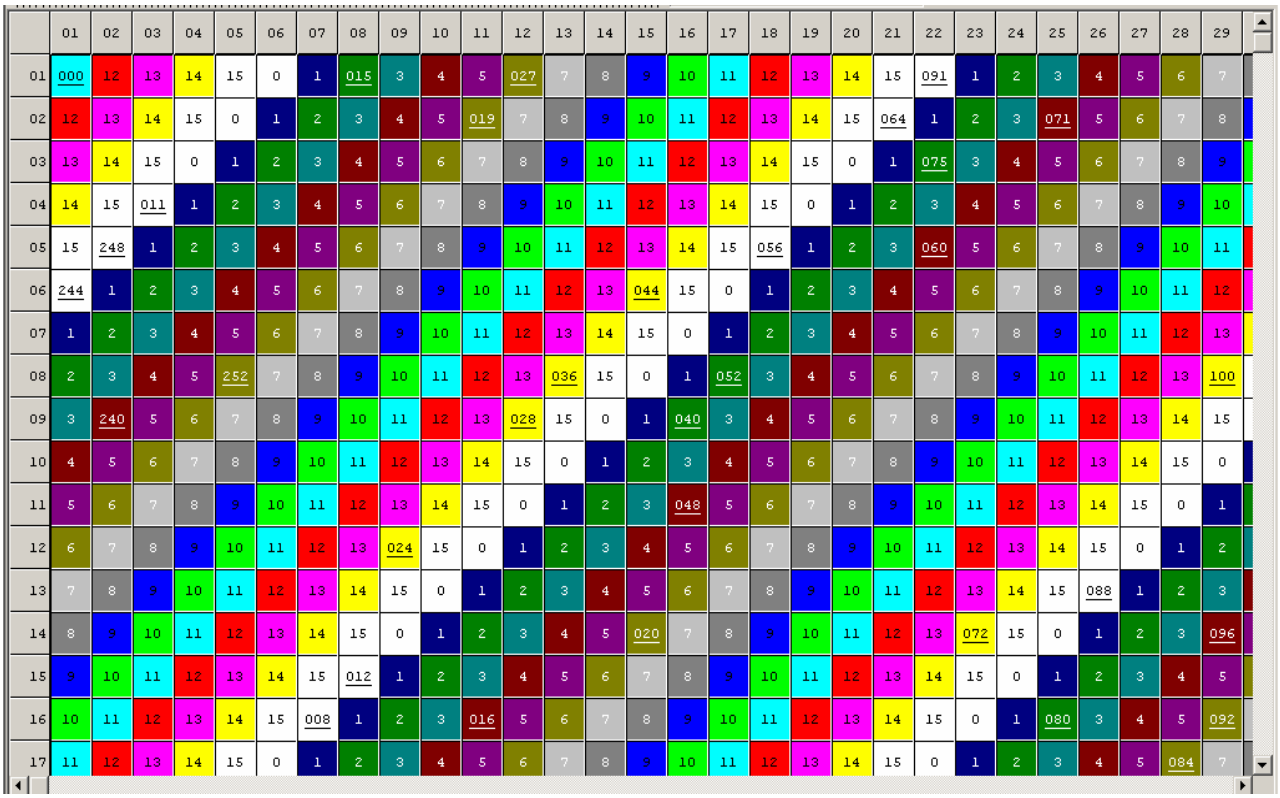


Fig. 3. The Cell Space after a few spontaneous steps: the guards (with underlined numbers) finished with a few steps in a random walk.

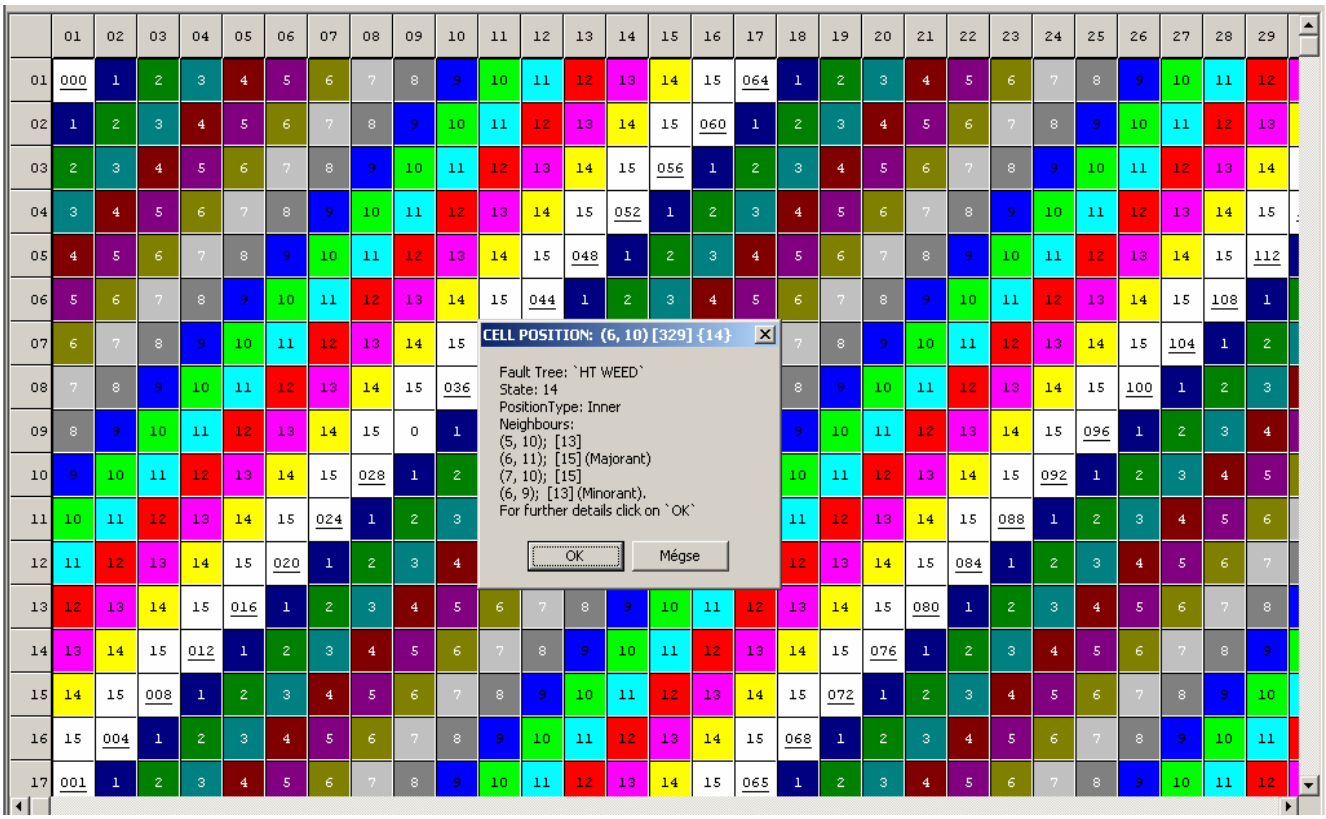


Fig. 4. The neighborhood if an „inner cell” in row 6, column 10. Its index (serial number) is 329, state = 14 („Mégse” is Hungarian for „Cancel”). After pressing „OK”, Fig. 6. displays.

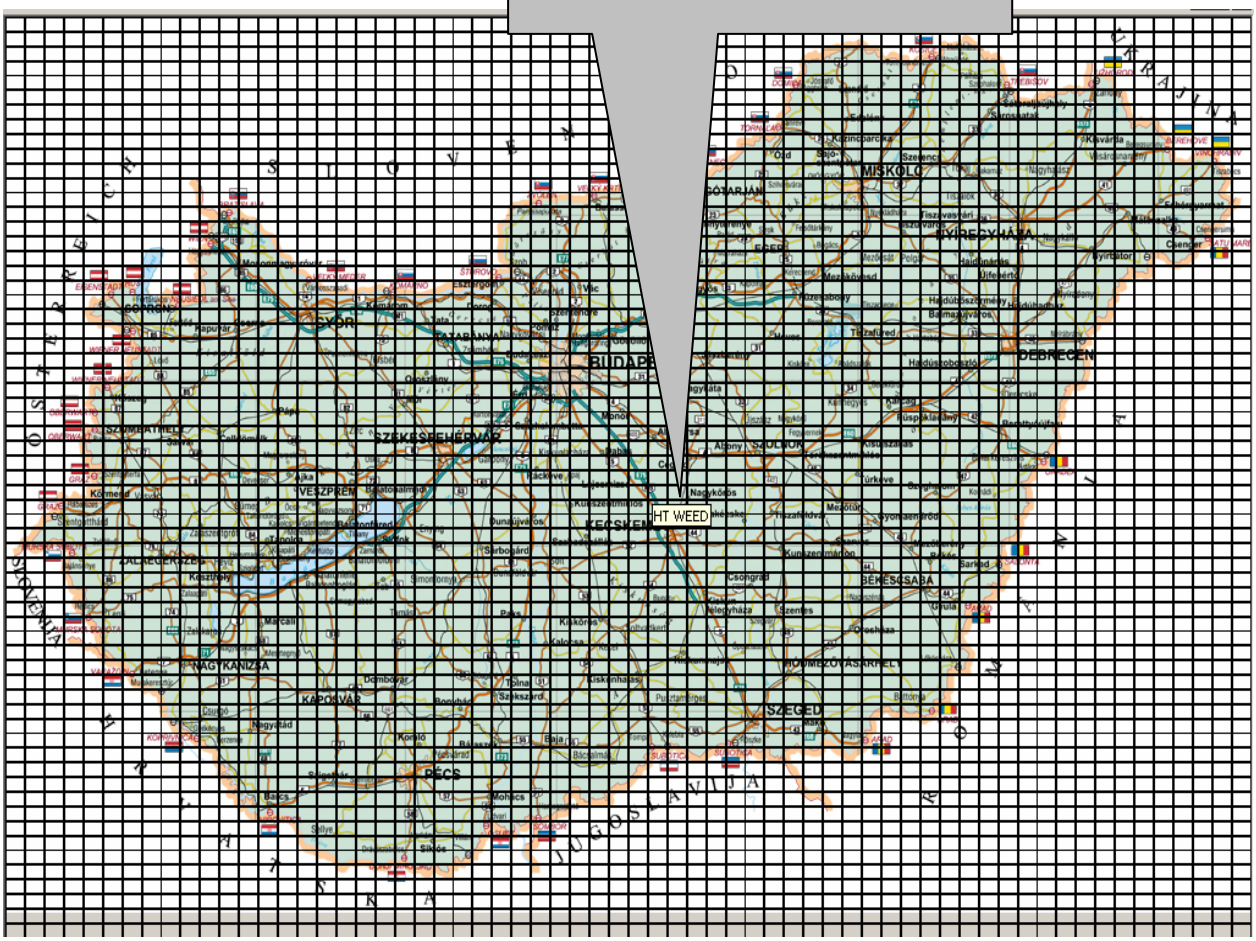


Fig. 5. The interpretation of the cell space CS. A site (here Hungary) is covered by a grid of 64x64 rectangles called „land units”. To each land unit there belongs a fault tree. (Different fault trees to different, but not necessarily vice versa.) ”HT Weed” is for „Herbicide Tolerant Weed. See
http://www.deh.gov.au/settlements/publications/biotechnology/hazard/fault.html ”

Let us consider a „*site*”. A *site* can be a territory, a domain, a field, a spot. To be concrete let it be a country (say, Hungary) that we want to investigate from disaster prevention and management point of views. To be more precise: we want to examine how to avert the attack that threatened the country in the general and abstract sense of the word „attack”[9] Divide the site to rectangular *land units* and suppose that to each land unit there belong a fault tree-like knowledge base equipped with suitable sensing devices (or sensors). This will be called henceforward a Risk Explicatum with the formal definition given later.

States

States of the cells in the Cellpace are uniformly assigned to the cells as s = 0,1,2,…,z where here in the present implementation z = 15.

[9] The concept of „attack” is to be considered here as the explicatum of the everyday word. For the method of explication, see [Carnap]

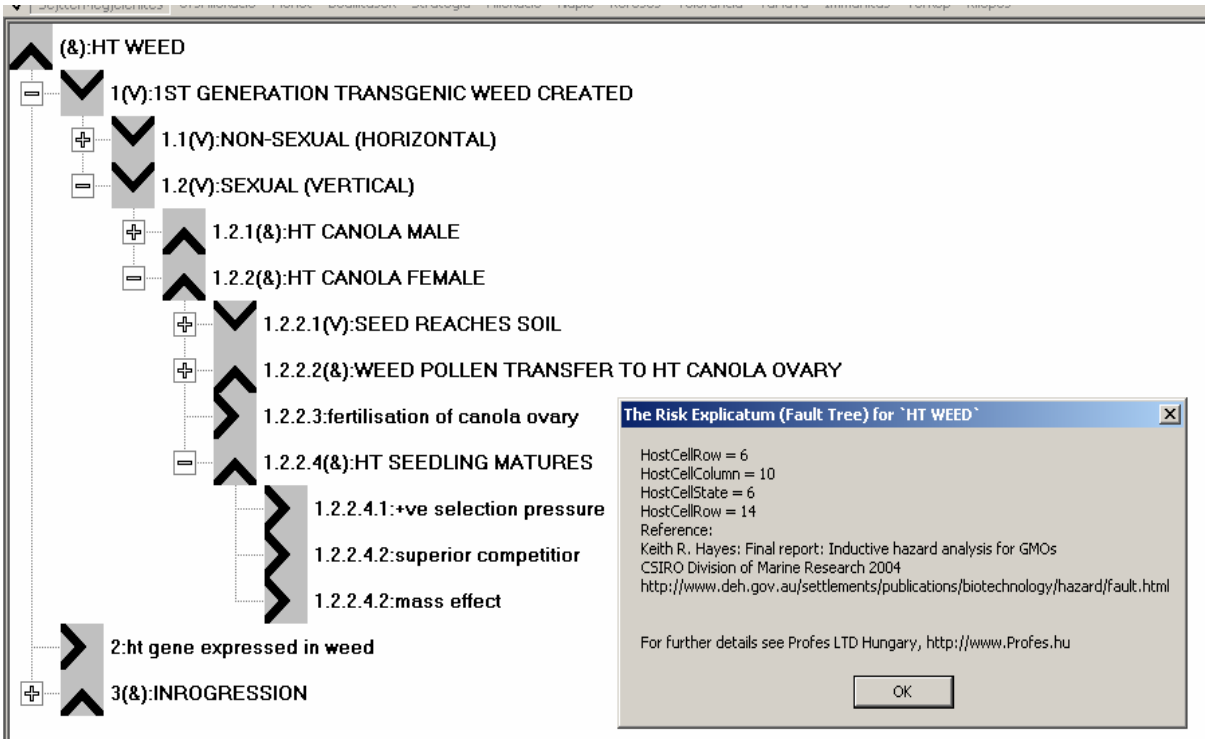


Fig. 6. The fault tree representation of a Risk Explicatum, REX belonging to the cell in question. (& means conjunction, V means disjunction)
(With permission of Profes Ltd. Hungary, http://www.profes.hu**)**

Transitions

The *transition rule* is probably the simplest possible one: „majorant replication"
The *majorant* m of a cell's state s is the state of the first[10] neighbor cell with m = s + 1 if s < z, and m = 0 if s = z (= 15). Thus the next state of a common cell is the majorant's state.

Attacks

An *attack* here means that a (common) cell changes its threat (or type) from „*virtual*" to „*real*"and changes its state (increasing, decreasing or remaining) randomly.
The rules for attack (or rather the normative restrictions) are as follows.
(AR1) Guards and Guards's neighbors are never attacked.
(AR2) Border cells are never attacked.
(AR3) A common cell in state *s* is attacked only if *s* > SL (the *Safety Level*)
At present we speak of 15 Safety Levels, SL = 0, 1, 2, … , 14.

The interpretation of the related notions of „attack"

The interpretation of the *cells* is the *land units* of a *site*. Here the site is a country (actually Hungary).
A cell can be either in a *virtual* or in a *real* state.
The common name of *virtual* and *real* cell state is the *threat* (of a land unit). We sometimes speak loosely of „virtual thread" and „real thread". Also, the term „unthreatened state"and „threatened" is used for the virtual state. Each cell must be always in one of the state 0 ,1,…, 15.
The state (of a cell) s_1 is interpreted as *less dangerous* than s_2 whenever $s_1 < s_2$, and both states have the same threat. Comparison of states with different threads is uninterpreted.

[10] „first" in the sense of the walk around the cell clockwise starting at the top neighbour (north)

Thus we theoretically differentiate between state s = 0 and r = 0 when s is unthreatened (virtual) while r is threatened (real). As for the interpretation, see below.

The interpretation of the *virtual* cell state is the required safety preparedness of the land unit in question. (Possessing fire extinguishers, sprinklers, etc.)
The interpretation of the *real* cell state is the degree of the actual thread of the land unit in question. It is measured by the *Franklin-parameters*: the cost and time necessary (and sufficient) to mitigate the damage and to restore the original unthreatened state of the cell.
The interpretation of the *state transition* changes according to the following cases.

- Case 1: virtual – virtual transition: controlling safety preparedness
- Case 2: virtual – real transition: attack
- Case 3: real – virtual transition: defense
- Case 4: real – real transition: thread spread or land unit destruction.
  In case of the state transition of the form $r_1 \Rightarrow r_2$ where both r_2 and r_1 are real states and $r_2 > r_1$ we speak of „thread spread".
  In case of the state transition of the form $r_1 \Rightarrow r_2$ where both r_1 and r_2 are real states and $r_2 = r_1$ we speak of „stagnant thread".
  In case of the state transition of the form $r_1 \Rightarrow r_2$ where both r_1 and r_2 are real states and $r_2 < r_1$ we speak of destruction (of the land unit). It may occur in the only case when $r_2 = 0$ and $r_1 = 15$.

The impact (effect) of an attack **ATT** wrt a **CS** configuration **CSC** is defined by the 3-tuple

<**CSC**, **ASI, SL**>

Where
CSC – the CellSpace Configuration of the given **CS.**
By definition **CSC** is a bit string of length L = *nCells* x *k*
Where
nCells is the number of cells in **CS.** (Here *nCells* = 4096)
k is the number of bits of *nStates.* (In case of *nState* = 16 then *k* = 4)

The Attack-algorithm

An attack in the SORS-model is carried out by the following algorithm.
1. Initialization.
Set the *AttackDuration. AttackDuration* is a quantity between 0 and 100. It means the number of steps when an attempt is made for changing the threat and state of a common cell.
Set the *Safety Level* SL to SL = 0
2. Set the Ammunition.
Select randomly the maximal number of the cells to be attacked.
It is

Ammunition = nCells x (Duration/ 100*)%*

3. Aim a cell.
Select a random cell index *CellIndex* of a common inner cell **C** that is not a guard's neighbor.
4. Calculate the state of Cell C(*CellIndex*) using the **State Calculation Algorithm**.
5. Increase the safety level.
If SL < 14 then increase it SL =: SL + 1 and go to **2.** else the attack-algorithm is over.

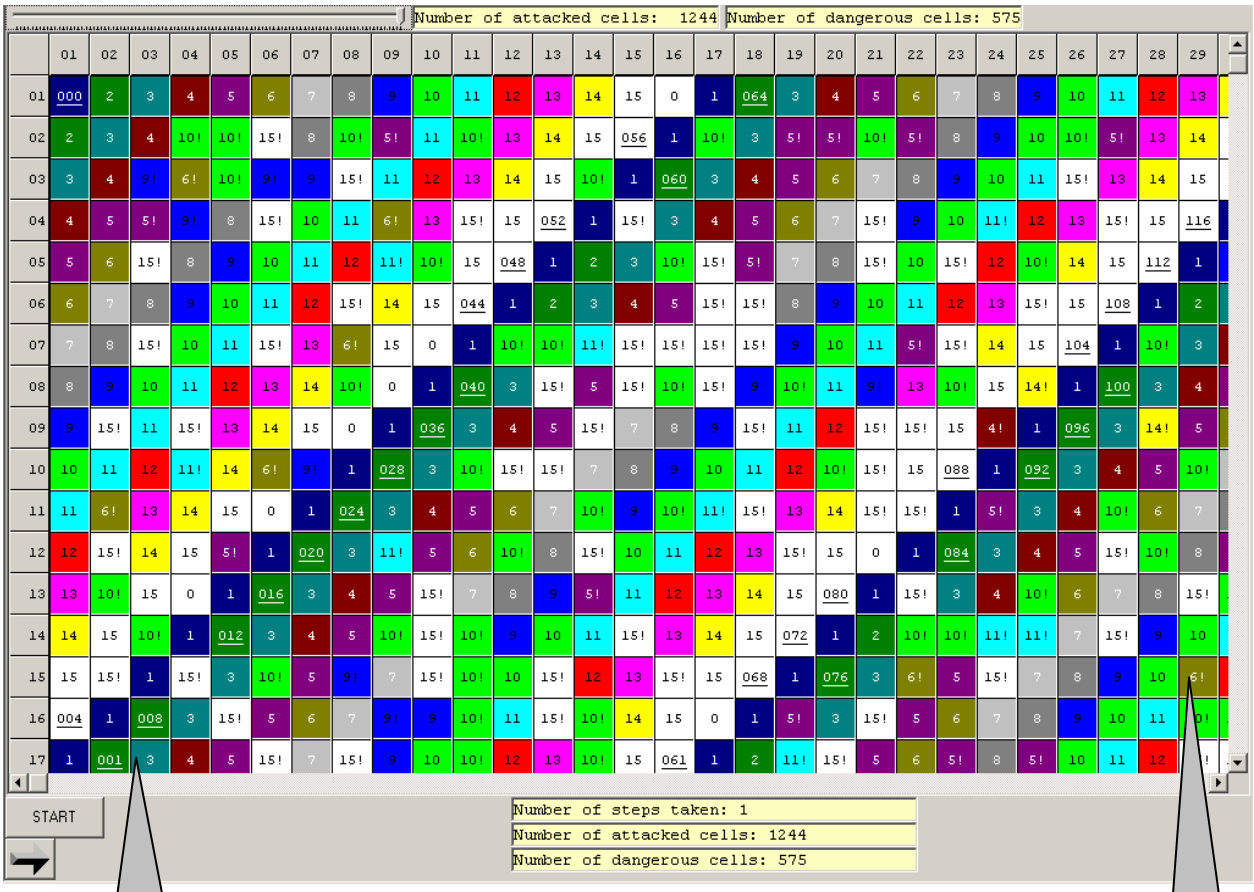


Fig. 7. The Cellspace (modeling the Landscape) after an attack at safety level = 0.
The initial global state (configuration) of the CS has radically changed.

The Forced State Calculation Algorithm

The basic idea of the CellState of a **Cell**(CellIndex) that occurs due to an attack step is as follows. The impact of an attack is detected by the sensors attached to the land unit (cell) and determine the *primestate* (the state, or rather the activity, of the prime events) of the Risk Explicatum **REX** (basically a Fault Tree belonging to the land unit). To each primevent p there belong in advance the four Franklin parameters $\mathbf{FP}_i$, i = 1, 2, 3, 4: the prevention and renovation cost and time need of p: the PrevCost(p) = $\mathbf{FP}_1(p)$, the RenCost(p)) = $\mathbf{FP}_2(p)$, the PrevTime(p) = $\mathbf{FP}_3(p)$, and the RenCost(p) = $\mathbf{FP}_4(p)$ respectively. These Franklin parameters $\mathbf{FP}_i(p)$, i = 1, 2, 3, 4 can be extended[11] to each composite event e even to the top event f the **REX.**

Now to derive the forced CellState s from the PrimeState defines the following variables:

MaxPrevCost = $\mathbf{FP}_1(f)$,

MaxPrevTime= $\mathbf{FP}_2(f)$,

SumPrevCost = The sum of all $\mathbf{FP}_1(p)$ with active p

SumPrevTime = The sum of all $\mathbf{FP}_1(p)$ with active p

Divide into four parts the intervals [0, MaxPrevCost] and [0, MaxPrevTime] to get four PrevCostIntervals and four PrevTimeIntervals as

PrevCostInterval (0) = [0, 0.25 x MaxPrevCost)

[11] For the description and code of the algorithm for calculating the top event's Franklin parameters from that of the prime events, contact the author.

PrevCostInterval (1) = [0.25, 0.5 x MaxPrevCost)
PrevCostInterval (2) = [0.5, 0.75 x MaxPrevCost)
PrevCostInterval (3) = [0.75, MaxPrevCost)
And
PrevTimeIntervals and four PrevTimeIntervals as
PrevTimeInterval (0) = [0, 0.25 x MaxPrevTime)
PrevTimeInterval (1) = [0.25, 0.5 x MaxPrevTime)
PrevTimeInterval (2) = [0.5, 0.75 x MaxPrevTime)
PrevTimeInterval (3) = [0.75, MaxPrevTime)

Define 16 „ForcedStateBox(i)" for i = 0, 1, …,15 by the direct products of the above intervals. The ForcedCellState is defined according that what box contains the SumPrevCost and the Sum PrevTime in the sense of Forced State Calculation Algorithm below.
Now the forced CellState determined by the PrimeState is defined as the result *TheCellState* of the

Forced State Calculation Algorithm: (written in Visual Basic 6[12])

```
Select Case True
   Case 0 <= SumPrevCost And SumPrevCost < MaxPrevCost * (1 / 4) And _
      0 <= SumPrevTime And SumPrevTime < MaxPrevTime * (1 / 4)
      TheCellSte = 0
   Case 0 <= SumPrevCost And SumPrevCost < MaxPrevCost * (1 / 4) And _
      MaxPrevTime * (1 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (2 / 4)
      TheCellSte = 1
   Case 0 <= SumPrevCost And SumPrevCost < MaxPrevCost * (1 / 4) And _
      MaxPrevTime * (2 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (3 / 4)
      TheCellSte = 2
   Case 0 <= SumPrevCost And SumPrevCost < MaxPrevCost * (1 / 4) And _
       MaxPrevTime * (3 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (4 / 4)
       TheCellSte = 3
   Case MaxPrevCost * (1 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (2 / 4)
And _
      0 <= SumPrevTime And SumPrevTime < MaxPrevTime * (1 / 4)
      TheCellSte = 4
   Case MaxPrevCost * (1 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (2 / 4)
And _
      MaxPrevTime * (1 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (2 / 4)
      TheCellSte = 5
   Case MaxPrevCost * (1 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (2 / 4)
And _
      MaxPrevTime * (2 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (3 / 4)
      TheCellSte = 6
   Case MaxPrevCost * (1 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (2 / 4)
And _
       MaxPrevTime * (3 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (4 / 4)
       TheCellSte = 7
   Case MaxPrevCost * (2 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (3 / 4)
And _
      0 <= SumPrevTime And SumPrevTime < MaxPrevTime * (1 / 4)
      TheCellSte = 8
   Case MaxPrevCost * (2 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (3 / 4)
And _
      MaxPrevTime * (1 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (2 / 4)
```

[12] With permission of Profes LTD. www.profes.hu

```
      TheCellSte = 9
   Case MaxPrevCost * (2 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (3 / 4)
And _
      MaxPrevTime * (2 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (3 / 4)
      TheCellSte = 10
   Case MaxPrevCost * (2 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (3 / 4)
And _
       MaxPrevTime * (3 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (4 / 4)
       TheCellSte = 11
   Case MaxPrevCost * (3 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost And _
      0 <= SumPrevTime And SumPrevTime < MaxPrevTime * (1 / 4)
      TheCellSte = 12
   Case MaxPrevCost * (3 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost And _
      MaxPrevTime * (1 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (2 / 4)
      TheCellSte = 13
   Case MaxPrevCost * (3 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost And _
      MaxPrevTime * (2 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (3 / 4)
      TheCellSte = 14
   Case MaxPrevCost * (3 / 4) <= SumPrevCost And SumPrevCost < MaxPrevCost * (4 / 4)
And _
       MaxPrevTime * (3 / 4) <= SumPrevTime And SumPrevTime < MaxPrevTime * (4 / 4)
       TheCellSte = 15
End Select
```

Defense

Defense here means that a (common) cell changes its *threat* (or type) from„*real*" to „*virtual*" and changes its threat according to the following defense rules (DR1-DR2)

(DR1) If a common cell **C** in a real state *r* has a guard neighbor, then the next state s of cell **C** is s = r, but the threat of **C** becomes *virtual.*

(DR2) The cell becomes a guard. (The guard „occupies the cell")

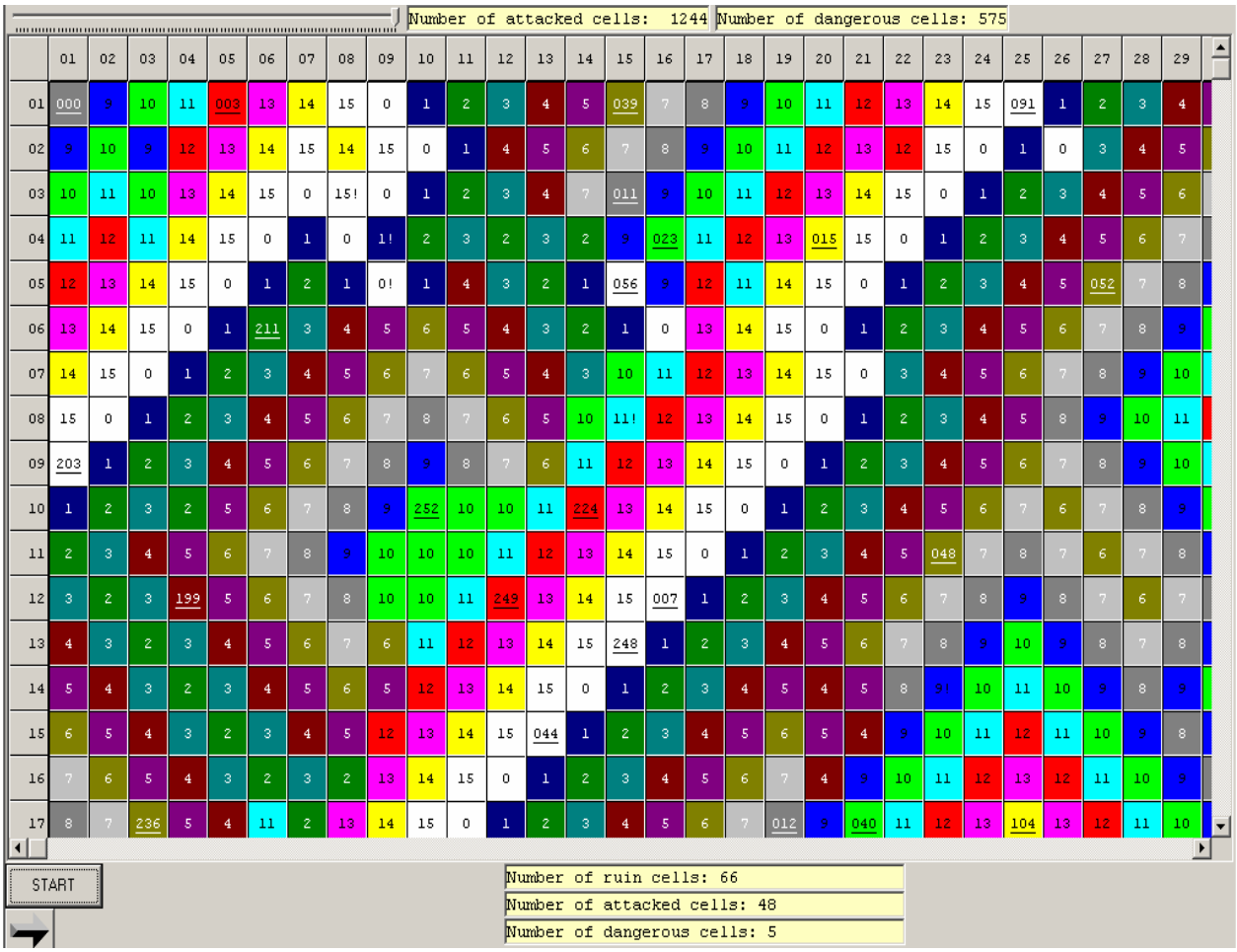


Fig. 8. After the 40-th defense step (nDS = 40)

Experiment: Lull – Attack – Defense

An (*in silico*) *experiment* with the system modeled (or rather normatively described) by the SORS project generally includes three global *epochs*:

- *The Lull*. It is a time interval with each cell being in virtual state, no state is missing, and the structure (the state configuration) of the CellSpace is more or less disordered, guards walk at random.
  During Lull, - as the experience shows - the states of set of the **CS'**s common cells form a cycle with length *nStates* = 16.
- *The Attack.* Randomly selected common cells in virtual state of random population change their threat to *real* and change their state depending on the cell's risk explicatum independently of the state transition rules using the Forced State Calculation Algorithm.
- *The Defense.* If a common cell in a real state has a guard neighbor, then the *threat* of the cell becomes *virtual,* the state remains unchanged and the guard occupies the cell. According to the experience the defense always ends with a success in more or less defense steps. See Fig. 9.-11.

The three epochs forms – by definition – an *X-run*.
An *experiment* is – by definition – the series of consecutive *X-runs* ending with the *last run* (*L-run*). The number of runs within an expert is denoted by *nRuns*
Let *X-run* be the nRun-th (nRun > 1) member of an experiment denoted by *X-run*(*nRuns*)
The *relative frequency* **RF**(*X-run*) of an *X-run* is - by definition –

$$\mathbf{RF}(X\text{-}run) = \mathrm{nDefs}(\mathrm{nRuns}) / \mathrm{nRun}$$

where
nDefs = the number of defense steps during the X-Run
Let *X1-run and X2-run two consecutive X-runs*:

$$X1\text{-}run = X\text{-}run(nRuns\text{-}1)$$
$$X2\text{-}run = X\text{-}run(nRuns)$$

The *last run* or the *stochastic limit* of an experiment is the X-run if the difference between its relative frequency and of its predecessor is relatively small. It is stipulated here the difference to be = 1%.
It is said that the experiment *ends* if *stochastic convergence* eventuates.

Performing and Report of an experiment

See Figs. 7 -11. As for the related algorithms see www.profes.hu

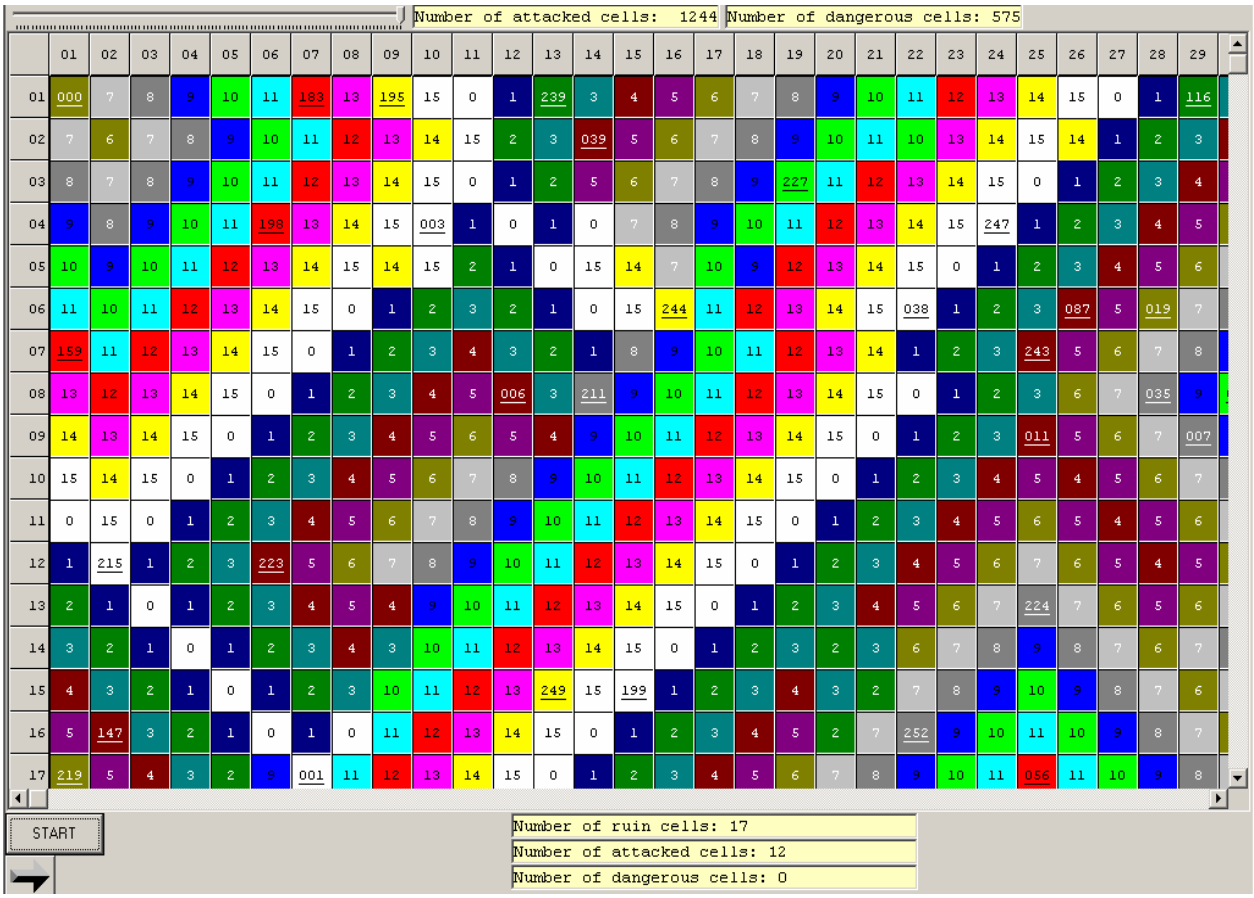


Fig. 9. After the 70-th defense step (nDS = 70)

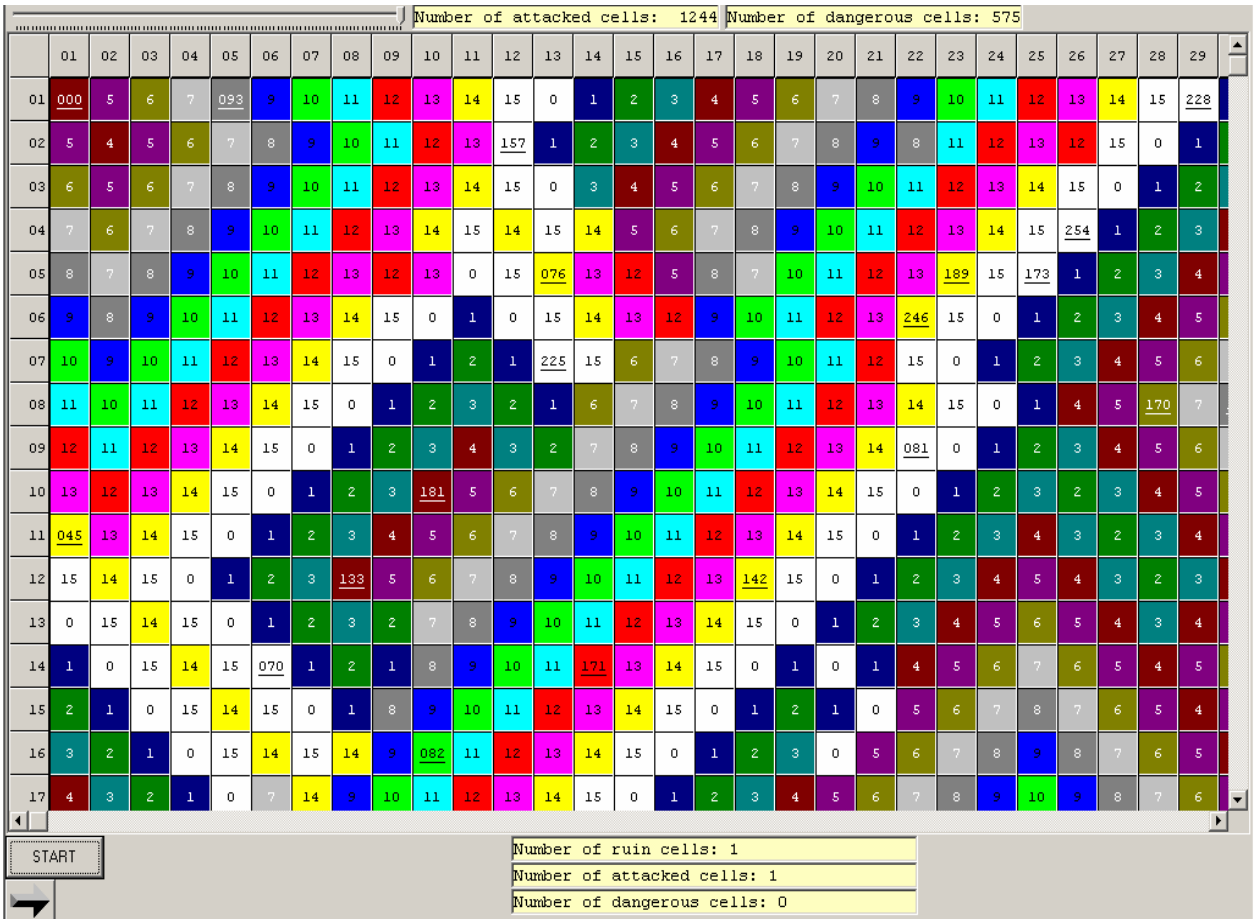


Fig. 10. After the 180-th defense step (nDS = 180)

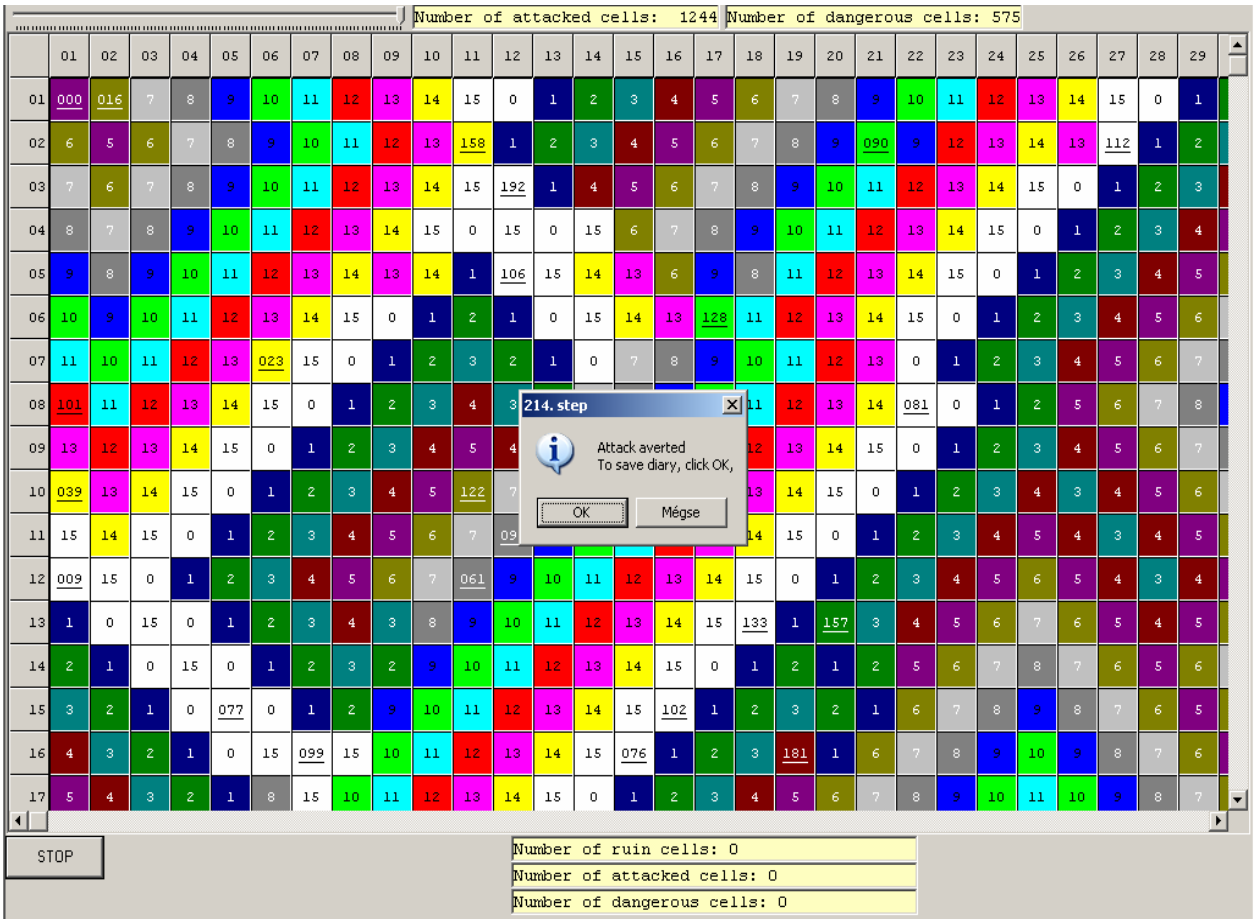


Fig. 11. Success after the 214-th defense step. (nDS =214)
In spite of the radical change in the initial configuration of the CS
Self defense was finally successful.

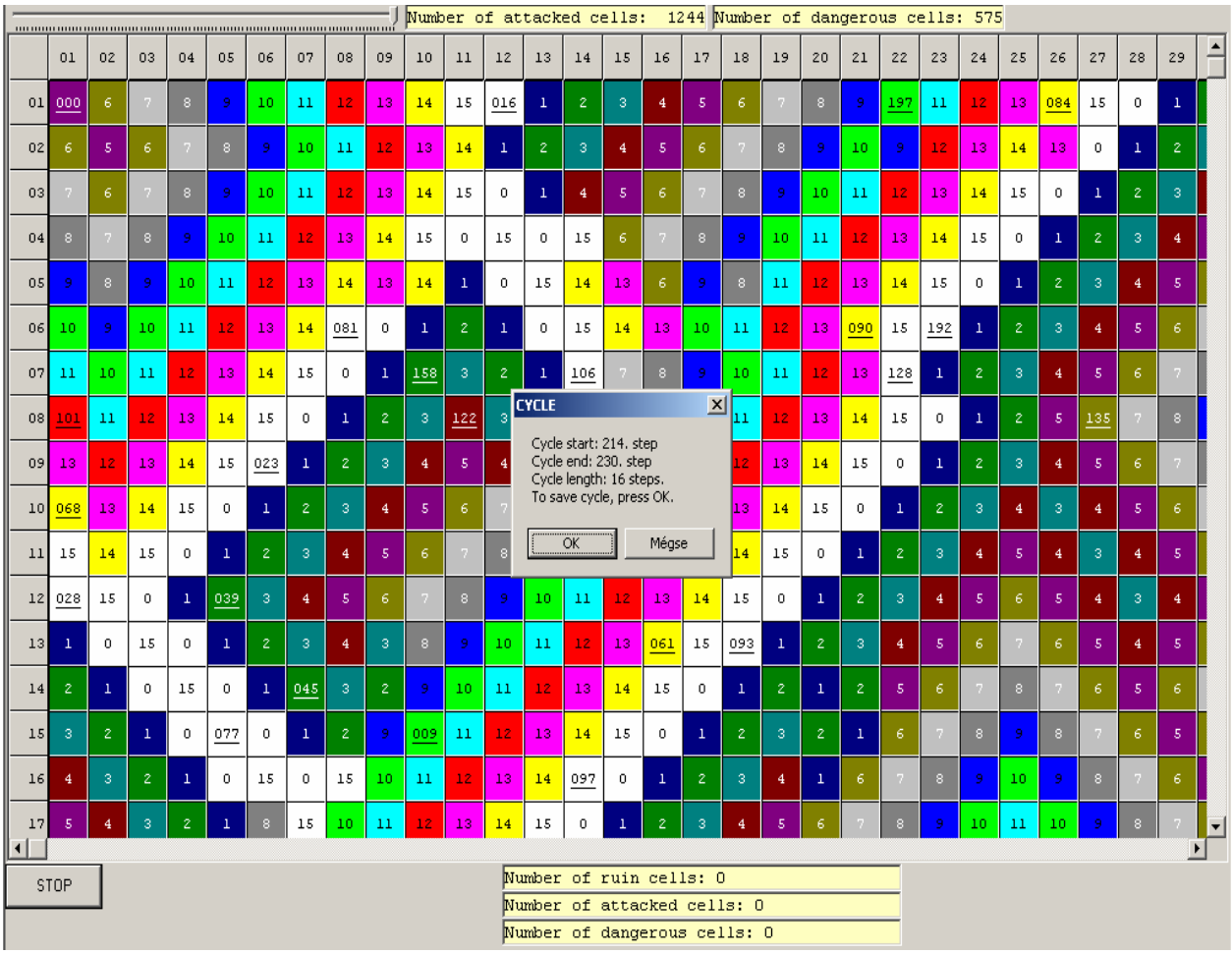


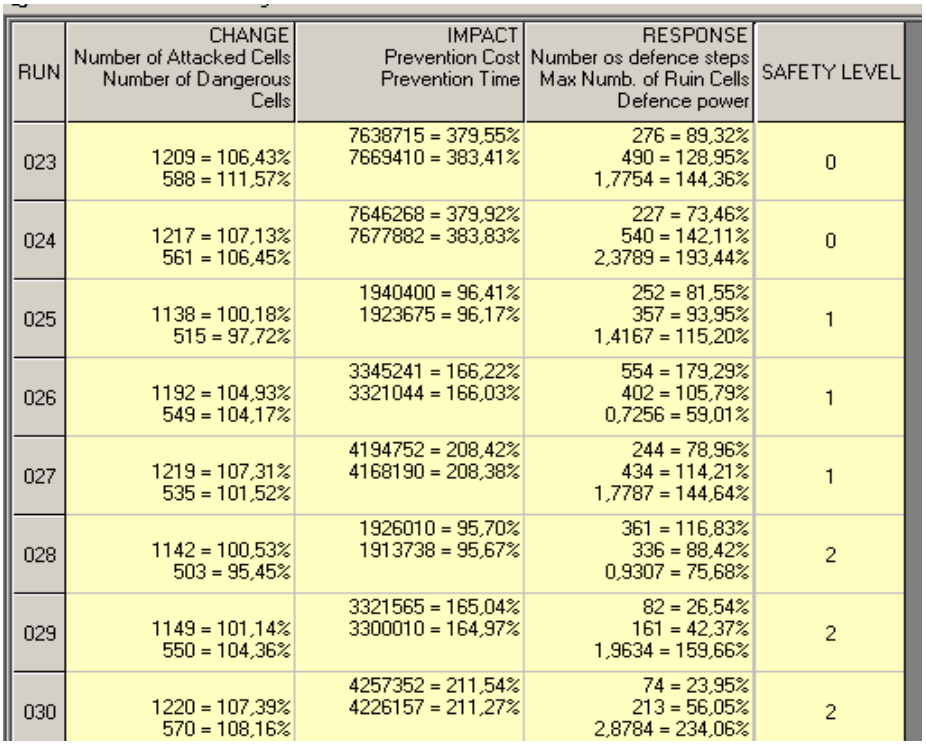
Fig. 12. Cycle after success at 214-th defense step

Power and Safety levels

A run of an experiment can be performed at several Safety Level SL. See Fig. 13.

RUN	CHANGE Number of Attacked Cells Number of Dangerous Cells	IMPACT Prevention Cost Prevention Time	RESPONSE Number os defence steps Max Numb. of Ruin Cells Defence power	SAFETY LEVEL
001	1136 = 100,00% 527 = 100,00%	2012596 = 100,00% 2000323 = 100,00%	309 = 100,00% 380 = 100,00% 1,2298 = 100,00%	0
002	1198 = 105,46% 533 = 101,14%	3376375 = 167,76% 3357698 = 167,86%	106 = 34,30% 184 = 48,42% 1,7358 = 141,15%	0
003	1183 = 104,14% 530 = 100,57%	4183546 = 207,87% 4165800 = 208,26%	112 = 36,25% 208 = 54,74% 1,8571 = 151,02%	0

Fig. 13. The first 3 runs of an attack experiment at safety level = 0



RUN	CHANGE Number of Attacked Cells Number of Dangerous Cells	IMPACT Prevention Cost Prevention Time	RESPONSE Number os defence steps Max Numb. of Ruin Cells Defence power	SAFETY LEVEL
023	1209 = 106,43% 588 = 111,57%	7638715 = 379,55% 7669410 = 383,41%	276 = 89,32% 490 = 128,95% 1,7754 = 144,36%	0
024	1217 = 107,13% 561 = 106,45%	7646268 = 379,92% 7677882 = 383,83%	227 = 73,46% 540 = 142,11% 2,3789 = 193,44%	0
025	1138 = 100,18% 515 = 97,72%	1940400 = 96,41% 1923675 = 96,17%	252 = 81,55% 357 = 93,95% 1,4167 = 115,20%	1
026	1192 = 104,93% 549 = 104,17%	3345241 = 166,22% 3321044 = 166,03%	554 = 179,29% 402 = 105,79% 0,7256 = 59,01%	1
027	1219 = 107,31% 535 = 101,52%	4194752 = 208,42% 4168190 = 208,38%	244 = 78,96% 434 = 114,21% 1,7787 = 144,64%	1
028	1142 = 100,53% 503 = 95,45%	1926010 = 95,70% 1913738 = 95,67%	361 = 116,83% 336 = 88,42% 0,9307 = 75,68%	2
029	1149 = 101,14% 550 = 104,36%	3321565 = 165,04% 3300010 = 164,97%	82 = 26,54% 161 = 42,37% 1,9634 = 159,66%	2
030	1220 = 107,39% 570 = 108,16%	4257352 = 211,54% 4226157 = 211,27%	74 = 23,95% 213 = 56,05% 2,8784 = 234,06%	2

Fig. 14. Runs of an attack experiment at safety level = 0, 1, 2

RUN	CHANGE Number of Attacked Cells Number of Dangerous Cells	IMPACT Prevention Cost Prevention Time	RESPONSE Number os defence steps Max Numb. of Ruin Cells Defence power	SAFETY LEVEL
099	557 = 49,03% 500 = 94,88%	1944393 = 96,61% 1928748 = 96,42%	92 = 29,77% 242 = 63,68% 2,6304 = 213,90%	13
100	591 = 52,02% 533 = 101,14%	3261390 = 162,05% 3229832 = 161,47%	118 = 38,19% 234 = 61,58% 1,9831 = 161,25%	13
101	495 = 43,57% 495 = 93,93%	1952106 = 96,99% 1935432 = 96,76%	70 = 22,65% 214 = 56,32% 3,0571 = 248,59%	14
102	519 = 45,69% 519 = 98,48%	3188275 = 158,42% 3153252 = 157,64%	92 = 29,77% 221 = 58,16% 2,4022 = 195,33%	14
103	511 = 44,98% 511 = 96,96%	4041860 = 200,83% 4006306 = 200,28%	101 = 32,69% 241 = 63,42% 2,3861 = 194,03%	14
104	526 = 46,30% 526 = 99,81%	4789094 = 237,96% 4760773 = 238,00%	84 = 27,18% 123 = 32,37% 1,4643 = 119,07%	14
105	529 = 46,57% 529 = 100,38%	5265785 = 261,64% 5241856 = 262,05%	66 = 21,36% 138 = 36,32% 2,0909 = 170,02%	14
106	537 = 47,27% 537 = 101,90%	5646968 = 280,58% 5627860 = 281,35%	92 = 29,77% 151 = 39,74% 1,6413 = 133,46%	14
107	512 = 45,07% 512 = 97,15%	5921252 = 294,21% 5915298 = 295,72%	81 = 26,21% 120 = 31,58% 1,4815 = 120,47%	14

Fig. 15. Runs 099 – 111 of an attack experiment at safety level = 13, 14

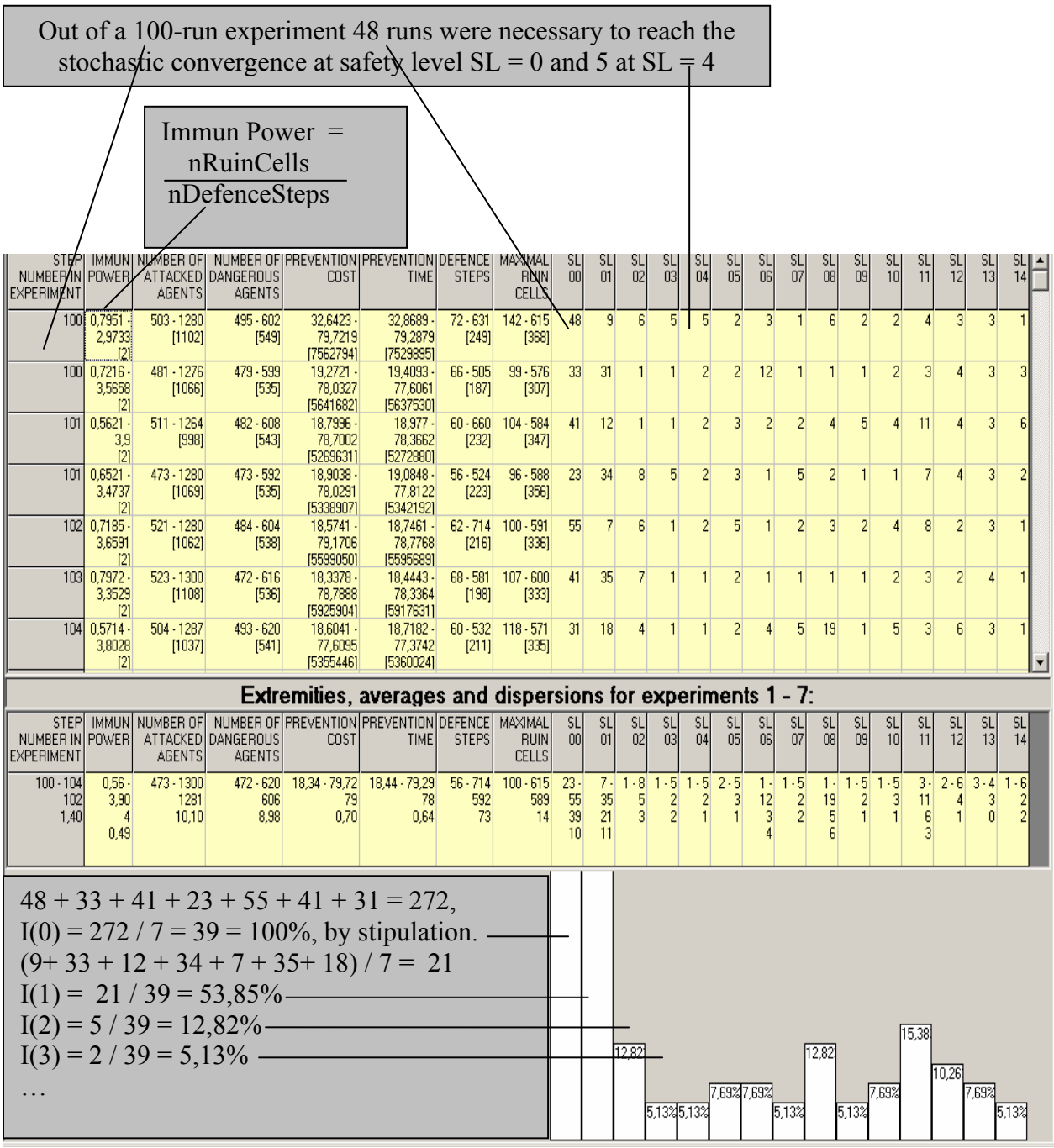


Fig. 16. Empirical Vulnerability in *r-view*. Results of the 1 -7 experiments at each safety level = 0, 1,…,14

STEP NUMBER IN EXPERIMENT	IMMUN POWER	NUMBER OF ATTACKED AGENTS	NUMBER OF DANGEROUS AGENTS	PREVENTION COST	PREVENTION TIME	DEFENCE STEPS	MAXIMAL RUIN CELLS	SL 00	SL 01	SL 02	SL 03	SL 04	SL 05	SL 06	SL 07	SL 08	SL 09	SL 10	SL 11	SL 12	SL 13	SL 14
120	0,5907 - 3,5077 [2]	482 - 1305 [1088]	482 - 588 [541]	18,6829 - 79,2211 [5939847]	18,8112 - 78,9364 [5934364]	55 - 611 [207]	107 - 622 [327]	54	1	24	1	7	1	2	1	1	12	7	2	2	2	3
120	0,4456 - 3,7681 [2]	509 - 1281 [1089]	480 - 591 [538]	19,0449 - 78,8619 [5752331]	19,2621 - 78,5706 [5748096]	55 - 662 [223]	99 - 572 [339]	50	28	5	1	5	3	5	3	1	1	6	2	6	3	1
121	0,7197 - 3,2857 [2]	520 - 1279 [1060]	485 - 605 [540]	19,0715 - 79,1429 [5867368]	19,1157 - 78,8627 [5864253]	52 - 493 [218]	81 - 584 [343]	58	20	2	1	3	1	7	4	1	2	2	12	4	1	3
122	0,553 - 3,4203 [2]	506 - 1292 [1029]	463 - 616 [539]	18,466 - 79,426 [5677641]	18,6062 - 79,0094 [5671392]	59 - 689 [220]	60 - 568 [322]	61	2	2	4	5	4	8	3	1	5	4	12	6	2	3
95	0,6358 - 4,5902 [2]	524 - 1289 [1087]	481 - 585 [541]	18,5535 - 78,4637 [5585737]	18,7186 - 78,1509 [5585048]	61 - 604 [203]	128 - 577 [330]	39	1	23	3	1	1	1	7	7	1	3	2	2	1	3
96	0,6182 - 3,3836 [2]	509 - 1282 [1086]	490 - 608 [537]	19,4151 - 79,1978 [5863543]	19,5501 - 78,7528 [5855799]	60 - 665 [194]	84 - 579 [319]	62	2	6	1	1	2	1	3	1	3	3	3	4	1	3
97	0,6584 - 3,6857 [2]	533 - 1294 [1086]	477 - 594 [541]	18,4677 - 79,2151 [6111303]	18,6674 - 78,8581 [6098159]	69 - 489 [195]	106 - 587 [324]	66	1	1	1	1	1	1	1	1	10	2	1	5	3	2

Extremities, averages and dispersions for experiments 1 - 60:

STEP NUMBER IN EXPERIMENT	IMMUN POWER	NUMBER OF ATTACKED AGENTS	NUMBER OF DANGEROUS AGENTS	PREVENTION COST	PREVENTION TIME	DEFENCE STEPS	MAXIMAL RUIN CELLS	SL 00	SL 01	SL 02	SL 03	SL 04	SL 05	SL 06	SL 07	SL 08	SL 09	SL 10	SL 11	SL 12	SL 13	SL 14
95 - 122 111 6,18	0,24 - 4,59 4 0,52	0 - 1335 1288 13,62	0 - 633 605 11,27	17,57 - 79,82 79 0,78	17,72 - 79,37 78 0,69	17 - 890 607 105	100 - 631 587 17	2 - 84 49 18	1 - 79 14 15	1 - 52 8 10	1 - 36 5 7	1 - 19 4 4	1 - 16 3 3	1 - 18 3 3	1 - 22 4 5	1 - 19 2 3	1 - 12 3 3	1 - 9 3 2	1 - 14 4 3	1 - 8 3 2	1 - 6 3 1	1 - 11 3 2

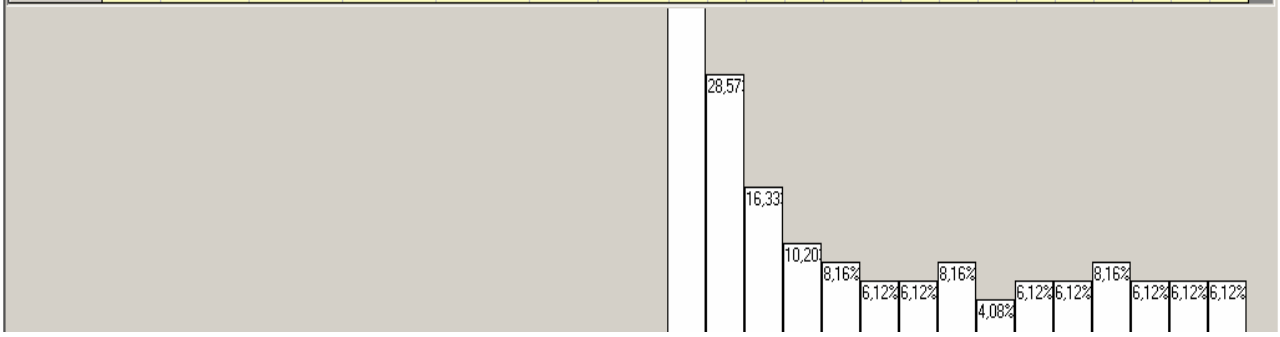


Fig. 17. Empirical Vulnerability in *r-view*. Results of the 1 -60 experiments at each safety level = 0, 1,…,14

Extremities, averages and dispersions for experiments 1 - 105:

STEP NUMBER IN EXPERIMENT	IMMUN POWER	NUMBER OF ATTACKED AGENTS	NUMBER OF DANGEROUS AGENTS	PREVENTION COST	PREVENTION TIME	DEFENCE STEPS	MAXIMAL RUIN CELLS	SL 00	SL 01	SL 02	SL 03	SL 04	SL 05	SL 06	SL 07	SL 08	SL 09	SL 10	SL 11	SL 12	SL 13	SL 14
100 - 116 110 3,83	0,00 - 4,00 4 0,48	473 - 1335 1283 14,20	458 - 633 603 10,44	18,00 - 80,00 78 1,30	18,00 - 80,00 78 1,22	17 - 1128 615 111	44 - 631 589 17	2 - 84 43 18	1 - 79 15 13	1 - 48 9 9	1 - 35 7 7	1 - 27 5 6	1 - 16 4 3	1 - 25 4 4	1 - 26 4 5	1 - 13 3 2	1 - 12 3 2	1 - 14 3 3	1 - 14 3 3	1 - 13 2 2	1 - 6 2 1	1 - 12 2 2

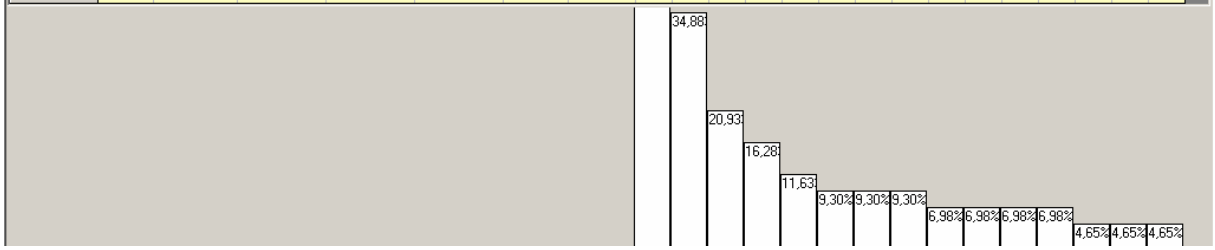


Fig. 18. Empirical Vulnerability in *r-view*. Results of the 1 -105 experiments at each safety level = 0, 1,…,14. From this number of experiments the shape of the *Vulnerability* (SL, X) function changes with decreasing frequency.

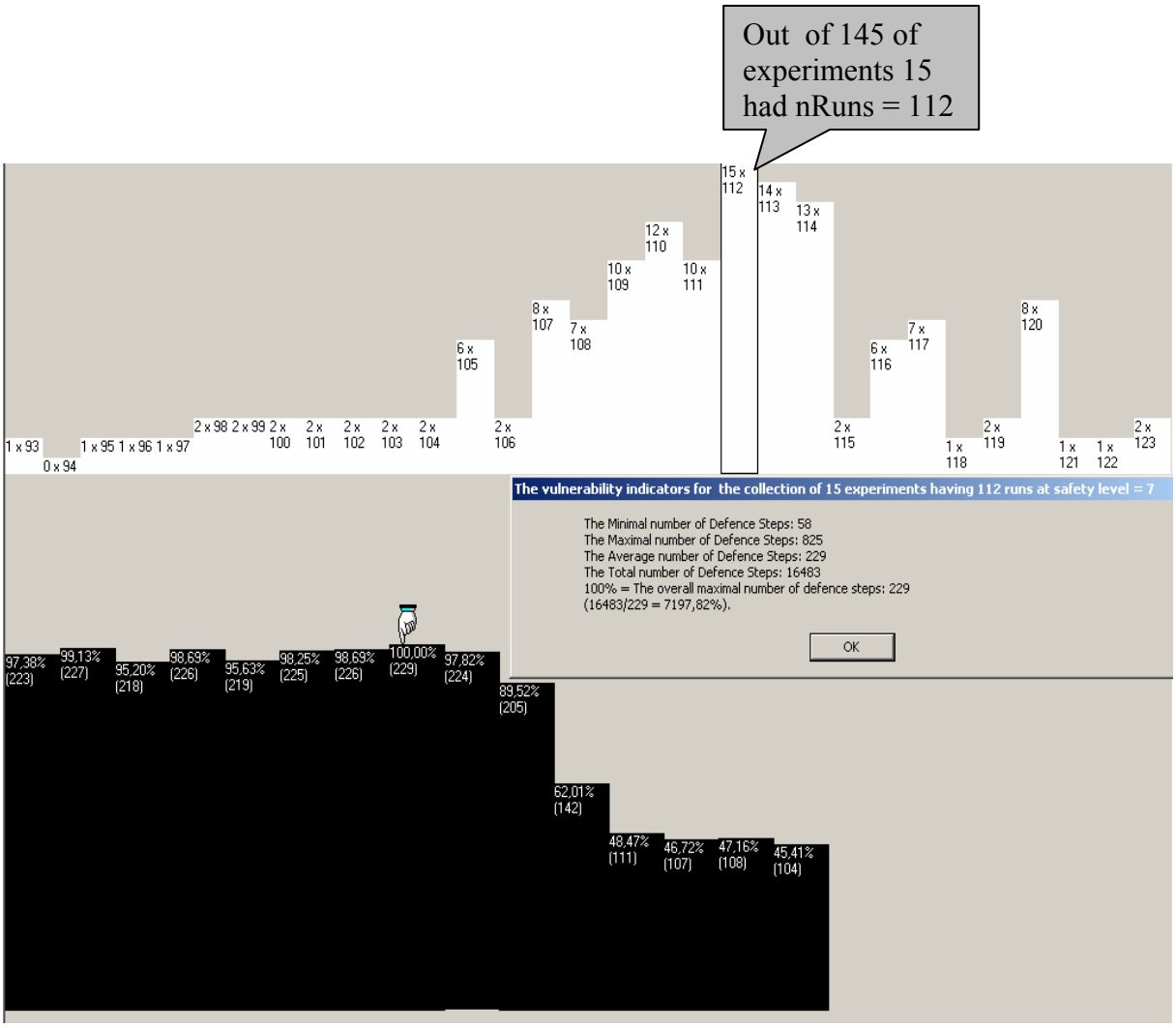


Fig. 19. Empirical Vulnerability in *s-view*. Results of the 1 -145 experiments at each safety level = 0, 1,…,14. The shape of the *Vulnerability* (SL, X) function changes with irregular frequency as a function of nX, the number of experiments..

Immunity and vulnerability

The concept of immunity[13] (of a SORS-like system) stems from the intuitive concept of *vulnerability.* If a system is wounded it looses or weakens its ability to recover or to cope with attacks. *Immunity* is, in a sense, the opposite of vulnerability. The „easier" to recover from a wounded state the better or higher the immunity of the system.

The precise characterization of the recovery process strongly depends on the definition of the „easiness" or rather the „difficulty" of the recovery. It seems to be fruitful[14] to define difficulty as the number of the global steps necessary (and sufficient) to reach a global system state without any cells being in a dangerous state i.e. to reach the **CS_Lull** following an

[13] Immunity is meant here in the intuitive form. Nothing to do with its juristic, medical or other explicative connotations.

[14] By stating that this concept is *fruitful* by no means meant that it is *faithful* also to the intuitive notion of vulnerability. This question is deeply related to the problematics of explication. See [Carnap]. The question of *fruitfulness* vs *faithfullnes* of a concept, in relation to the question of rigor, see [Kreisel]

attack. Intuitively immunity is somewhat similar to *tolerance*[15]. The main difference between tolerance and immunity is that immunity is tolerance *as a function of safety level*.
Along the explication we must take into consideration the following.
The very notion of *attack* is *par excellence* inherently *stochastic*. That is the main reason that one is forced to investigate attack-prone systems from immunity point of view through *in silico* experiments. It follows that the results of the experimental study necessarily refers to experiments. Experiments, however, as such, are always incidental. So, to get theoretical results of considerable generality and validity, one must dispose of all references to experiments. This leads to the concepts of theoretical *immunity* in both *r-view* and s-*view* respectively.
Once a quantitative notion of Vulnerability, measured in %, is agreed upon, relating to an experiment performed at a Safety Level SL, we stipulate *empirical vulnerability* and *empirical*

Immunity (SL, X) = 100% - *Vulnerability* (SL, X).

It seems there are two more or less natural ways to explicate empirical vulnerability. They will be named respectively *r-view* (short for „run-view) and *s-view* (short for „step-view).
Accordingly we will speak of *r-Immunity* and *s-Immunity* in both *r-view* and s-*view* sense.
As a basic concept to prepare for the *r-view* let us introduce nRuns(SL, X) as the number of *runs* wrt an experiment X, performed at Safety Level SL, necessary (and sufficient) to reach the *Stochastic Convergence*.
As a basic concept to prepare for the *s-view* let us introduce nDS (SL, X) the number of global defense *steps* wrt an experiment X, performed at Safety Level SL, necessary (and sufficient) to reach the **CS-Lull** i.e. the **CS** configuration with no *real* (thread of) cellstate. See Figs. 7. – 11.
As an overview some information are presented concerning the results of the algorithms related to immunity or vulnerability in both *r-views* and *s-views*. See Figs. 16 – 19.

Summary

An artificial immunity concept was studied that emerged from an artificial self organizing network capable of defending itself called SORS (Self Organizing Raiding System). It is a systolic closed four-neighbor cellular automaton with two types of cells: defender an defendee in other words guards and common cells. Each cell has 16 state (0..15) and is equipped with a logic fault tree. Two types of cellstate interpreted as the threat of the cell is defined: virtual and real. An attack concept against the SORS is modeled. There are two kind of state transition. Spontaneous and forced. Spontaneous transition occurs according to the common cells. Forces state transition occurs due to the attack. The transition function for the common cells is the „majorant replication" where the majorant of a cell is the first clockwise-found neighbor with the highest state modulo 15, if any.
The transition function for the guard cells is the one-side-step random walk with some restriction: guards don't step on each other, etc. Guards occupy the suitable common neighbor cell and then obey the transition rule for common cells.
The forced cell state cussed by an attack is calculated by an algorithm using the Boolean algebraic properties of the fault tree attached to the cell in question.
Two immunity concept is defined called empirical and theoretical respectively.
The empirical immunity concept is based on the intuitive notion of vulnerability and studied experimentally *in silico*.
Formally *Immunity* (SL, X) = 100% - *Vulnerability* (SL, X)
where X is an experiment performed at a safety level SL (0..14)
An experiment consists of three epochs called respectively Lull, Attack and Defense. Each experiment is conducted at a Safety Level until stochastic convergence wrt the relative

[15] See [Bukovics, Tolerance]

frequency of the defense steps following an attack. A common cell in state *s* is attacked only if *s* > SL (the *Safety Level*).
Each experiment X is characterized by the *nRuns* number of runs and by the *nSteps* the number of defense steps at a Safety Level necessary and sufficient to reach stochastic convergence.
Accordingly two kind of immunity concept is defined. These are called respectively of *r-view* and *s-view*. The theoretical immunity concept is derived from the empirical by standard data processing techniques.
The result received from about a 150 *in silico* experiments is that although the theoretical *s-view* Immunity is more natural and stands closer i.e. more *faithful* to the intuition than the of the *r-view*, the latter, however seems to be more promising, more regular, in one word, more *fruitful*.
Immunity is a generalization of the recently introduced similar concept for SORS-like systems viz tolerance. Tolerance can be considered as the immunity at Safety Level = 0. In other words tolerance is the „zero-immunity". In this framework – of course – „zero tolerance" strictly speaking makes no sense (at least in the police sense) but it may mean „no immunity".

Future Work

In a forthcoming continuation of the present paper we want to provide a detailed report of our work. It is intended to include the empirical and theoretical immunity algorithms for both cases of *r-view* and *s-view*. Related (VB6 source) codes are available from Profes lTD, Hungary, www.profes.hu .

References

[Bukovics, tolerance]: István Bukovics:
The Study of The Tolerability of the Ecosphere in Cellular Automation Models
CEPOL-eDoc ID 6492 www.edoc.cepol.net/6492 also available from the author istvan.bukovics@katved.hu

[Carnap]: Carnap, R.: Physikalische Begriffsbildung. Karlsruhe, Braun, 1926

[Kreisel]: G. Kreisel: Informal Rigor and Completeness Proofs. In: Jaakko Hintikka (ed.) The Philosophy of Mathematics. Oxford University Press. Oxford, UK. 1969, p. 78 – 94.

[Riguet]: Riguet, Jacques: Automates cellulaires a board et automates Codd-ICRA. Comptes rendus de l'academie les sciences de Paris Série A t.282 19 janvier 1976 p. 166 -170 et 26 janvier p. 239-242.
(Hungarian translation available from istvan.bukovics@katved.hu)

[Wolfram]: Wolfram, S. (1994). Cellular Automata and Complexity. Reading, MA, Addison-Wesley.