ePublishing

Open Schools Journal for Open Science

Vol 5, No 2 (2022)

Open Schools Journal for Open Science

Simulating Predator-Prey System by Cellular Automata

Fabien Teofanis Kunis, Martin Dimitrov, Desislava Markova

doi: 10.12681/osj.31250

Copyright © 2022, Fabien Teofanis Kunis, Martin Dimitrov, Desislava Markova

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0.

To cite this article:

Kunis, F. T., Dimitrov, M., & Markova, D. (2022). Simulating Predator-Prey System by Cellular Automata. *Open Schools Journal for Open Science*, *5*(2). https://doi.org/10.12681/osj.31250

https://epublishing.ekt.gr | **e-Publisher:** EKT | **Downloaded at:** 26/01/2026 08:14:17

Simulating Predator-Prey System by Cellular Automata

Martin Dimitrov[1], Desislava Markova[2], Fabien Kunis[1,3]
[1]Boyan Penev High School, Sofia, Bulgaria
[2]American College, Sofia, Bulgaria
[3]Sofia University St. Kliment Ohridski, Sofia, Bulgaria

Abstract
Our work explores the possibilities of using cellular automata to simulate the predator-prey interaction. A cellular automaton is a discrete model that allows it to be used for various computational purposes. In recent years, we have observed how important it is to know the behaviour of various biological and ecological systems. Knowledge of the behaviour and evolution of different biological and ecological systems will allow us to adapt more quickly to different possible scenarios. Our work aims to simulate the behaviour of a predator-prey system using cellular automata and get realistic data from the simulation.

Keywords
Predator–prey system, cellular automata, computer simulation.

Predator–prey system

Let's look at the populations of two species interacting with each other – one as a predator and the other as prey. Our goal is to describe how the sizes of these populations change over time. One way to do this is to use the Lotka–Volterra model. A pair of differential equations describe this model. Their solutions are the sizes of the populations as functions of time. This model is described in detail and simulated in our previous article (Kunis & Dimitrov, 2020).
In this article, we will take a different approach: we will simulate the predator-prey interactions with cellular automata.

Cellular automata

A cellular automaton is a model which consists of a rectangular grid of cells where each cell can have a finite number of states, for example – alive or dead (Shiffman, 2012). For each cell, a neighbourhood is defined. This can be done in many ways, but usually, the cell's neighbours are its adjacent cells.
In the beginning (at time $t = 0$), each cell is assigned an initial state (usually at random). All the cells and their states at a particular time t are called the *generation* at time t. The next generation (i.e., the one at time $t = 1$) is created by following a specific set of rules. These rules

determine the new state of each cell according to its current state and the states of its neighbour cells.

By repeating this process, we can examine how the characteristics of the generations change over time. For instance, we can trace the number of cells in a particular state or their position.
Let's now define the cellular automaton we will use to simulate the predator-prey interactions.
First, let the dimensions of our grid be 100 by 100, as this will result in reasonable population size of $10\ 000$. From a statistical point of view, that is big enough as it provides a good number of different configurations. This number is also small enough to suit our practical needs.
Second, let the cells have three possible states: predator (for example, a fox), prey (for example, a rabbit), and an empty cell (the environment). We will use the following colours to visualize these states: red for the predator (fox), green for the prey (rabbit), and white for the empty cell.
Third, we need to define what a neighbourhood is. There are several ways to do this; for example, the two most frequently used are the Moore and the von Neumann neighbourhood (Chen, 2009). For our purposes, the Moore model will be better since the eight cells closest to a particular cell will have the most significant impact on it. Note that unlike the cells in the grid's interior, those on the edges have less than eight neighbours.
Finally, we must choose the rules of our model. This is the most challenging part, and there is no right or wrong way of doing it. We may consider which interactions are biologically logical and experiment to see which rules will result in a realistic simulation outcome.

Cellular automaton representation of the predator–prey system

Let's consider the following set of rules for our model.
1. Let there be a fox in a certain cell at time t (i.e., the cell is red).
We will consider hunger as the leading cause of death among the foxes. We assume that there is enough food for the foxes in a fox's neighbourhood if there is at least one rabbit. We will ignore other factors such as age, illness, or competition for a territory to keep our model as simple as possible.
Following this description, we derive these rules:
At time $t + 1$:
1.1) the fox will survive if there are at least as many rabbits as foxes in its neighbourhood (i.e., the cell will remain red)
1.2) the fox will die of hunger if there are fewer rabbits than foxes in its neighbourhood (i.e., the cell will turn white).

2. Let there be a rabbit in a certain cell at time t (i.e., the cell is green).
We will consider the fox–rabbit interactions as the main cause of death among the rabbits and assume that there is always enough food (grass) for the rabbits. We ignore other factors such as health and terrain to keep our model as simple as possible.
Now, let us look at the fox–rabbit interactions. If there are no foxes in the rabbit's neighbourhood, the rabbit will survive. However, if there is at least one fox, there is a probability that the rabbit will die because the fox will eat it. Otherwise, it will manage to escape and survive.

We will assume that being full is the only necessary condition for the fox to reproduce. Therefore, if the fox manages to eat the rabbit, a new fox will be born in the cell where the rabbit was.

Also, we will consider that four rabbits in the rabbit's neighbourhood are too many. Therefore, the rabbit will die. This rule is introduced so that the population of rabbits can be limited. It is unrealistic that a population can increase unlimitedly.
This description translates to the following rules:
At time $t + 1$:
2.1) if there are more than or equal to four rabbits in the rabbit's neighbourhood, the rabbit will die (i.e., the cell will turn white)
2.2) if there are no foxes in the rabbit's neighbourhood, it will survive (i.e., the cell will remain green)
2.3) if there is at least one fox in the rabbit's neighbourhood, there is a probability p that the rabbit will be eaten by a fox and a new fox will be born in its place (i.e., the cell will turn red)
2.4) there is a probability $1 - p$ that the rabbit will escape and survive (i.e., the cell will remain green).
During the simulation of the model, we will consider different values for p.

3. Let a certain cell be empty at time t (i.e., the cell is white).
We consider the following set of rules:
At time $t + 1$:
3.1) a rabbit will be born if there are more rabbits than foxes in the cell's neighbourhood (the cell will turn green)
3.2) a fox will be born if there are more foxes than rabbits in the cell's neighbourhood and if there is at least one rabbit (the cell will turn red)
3.3) the cell will remain empty if there are just as many rabbits as foxes or if there are no rabbits (the cell will remain white).

Simulation of the predator–prey system

We will simulate the predator-prey system using a program that implements cellular automata. The program will track the sizes of the two populations over time and display the system's state at regular intervals.
The program code is written in the programming language Java, and it's shown in the pictures below (*Figure 1* and *Figure 2*).

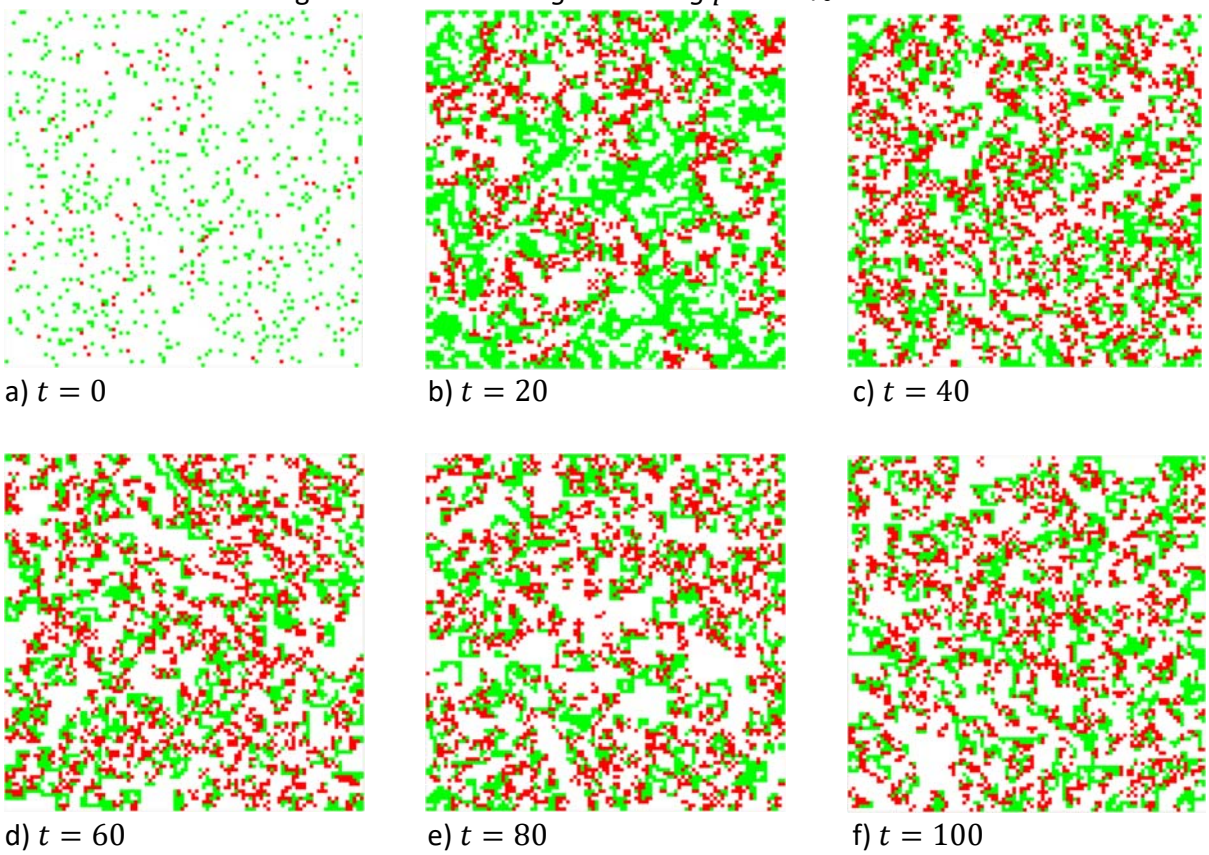
```java
// Start simulation
for (int gen = 1; gen <= genCount; gen++) {
    int [][] nextGrid = new int[n][n];

    // Calculate next generation
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {

            int preyNeighbours = countPreyNeighbours(i,j);
            int predatorNeighbours = countPredatorNeighbours(i,j);

            // Rules for empty cell
            if (grid[i][j] == 0) {

                if (preyNeighbours > 0) {
                    if (predatorNeighbours > preyNeighbours) {
                        nextGrid[i][j] = 2;
                        predatorStats[gen]++;
                    }
                    else if (preyNeighbours > predatorNeighbours) {
                        nextGrid[i][j] = 1;
                        preyStats[gen]++;
                    }
                }
                else {
                    nextGrid[i][j] = 0;
                }
            }
```

Figure 1. Rules for empty cells.

```java
// Rules for prey
else if (grid[i][j] == 1) {
    int Rand = rand.nextInt(100);

    if(preyNeighbours > 3){
        nextGrid[i][j] = 0;
    }
    else{
        if(predatorNeighbours > 0 && Rand < 100){
            nextGrid[i][j] = 2;
            predatorStats[gen]++;
        }
        else{
            nextGrid[i][j] = 1;
            preyStats[gen]++;
        }
    }


// Rules for predator
else if (grid[i][j] == 2) {

    if(preyNeighbours < predatorNeighbours + 1){
        nextGrid[i][j] = 0;
    }
    else{
        nextGrid[i][j] = 2;
        predatorStats[gen]++;
    }

}
```

Figure 2. Rules for prey and predator.

We obtain the following results shown in *Figure 3 using* $p = 80\%$.



a) $t = 0$ b) $t = 20$ c) $t = 40$

d) $t = 60$ e) $t = 80$ f) $t = 100$

Figure 3. The states of the predator–prey system displayed at times $t = 0, 20, 40, 60, 80, 100$ *where* $p = 80\%$.

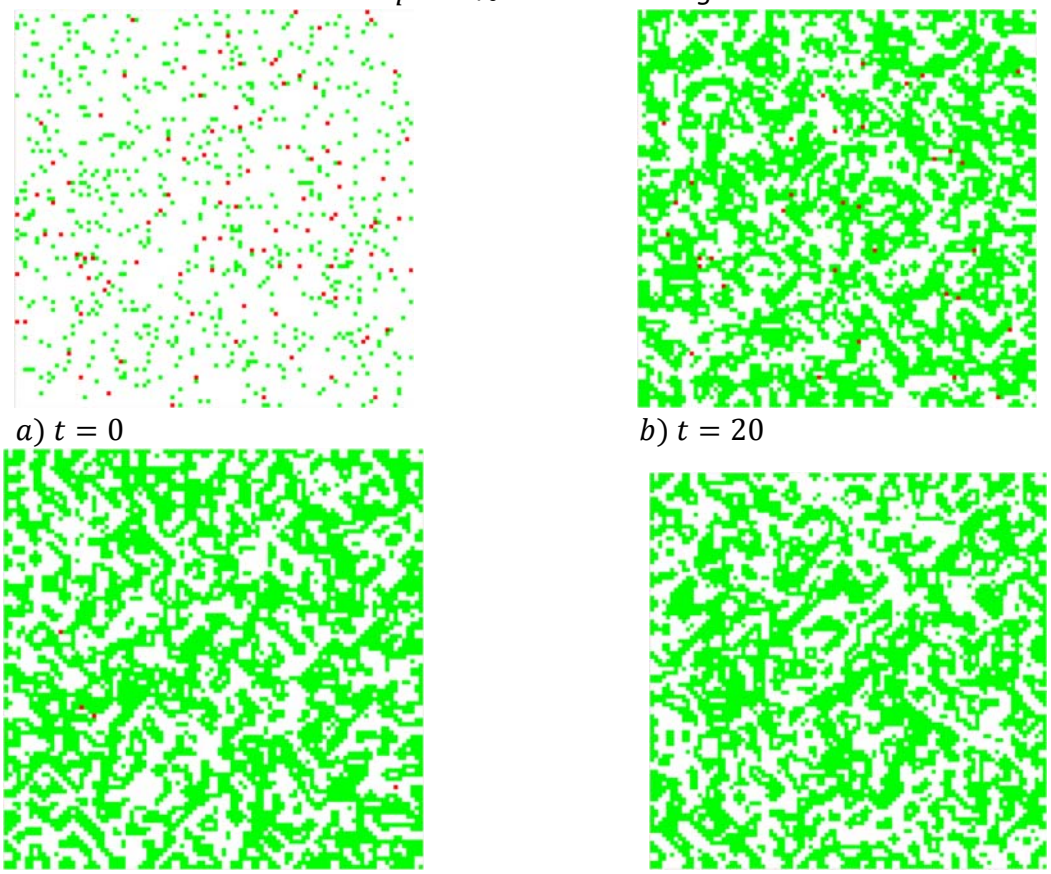
At time $t = 20$, the foxes have reproduced themselves and occupy the regions close to the initial location of the foxes, and the rabbits occupy all the other regions. At time $t = 40$, we observe a uniform diffuse distribution over the whole grid of the rabbits' and the foxes' populations. The exact location of the animals varies, but the type of distribution remains the same over time – the system reaches a stable state.

Note that even though the sizes of the two populations change over time, that is not evident in the pictures. That is because the difference between the maximum and the minimum size of the populations is just 2% of the number of cells in the grid. The change is too small to be viewed with a naked eye.

Next, we investigate the system when varying the parameter p.

1. Case: $p = 0\%$

The results from the simulation when $p = 0\%$ are shown in *Figure 4*.



$a)\ t = 0$ $b)\ t = 20$

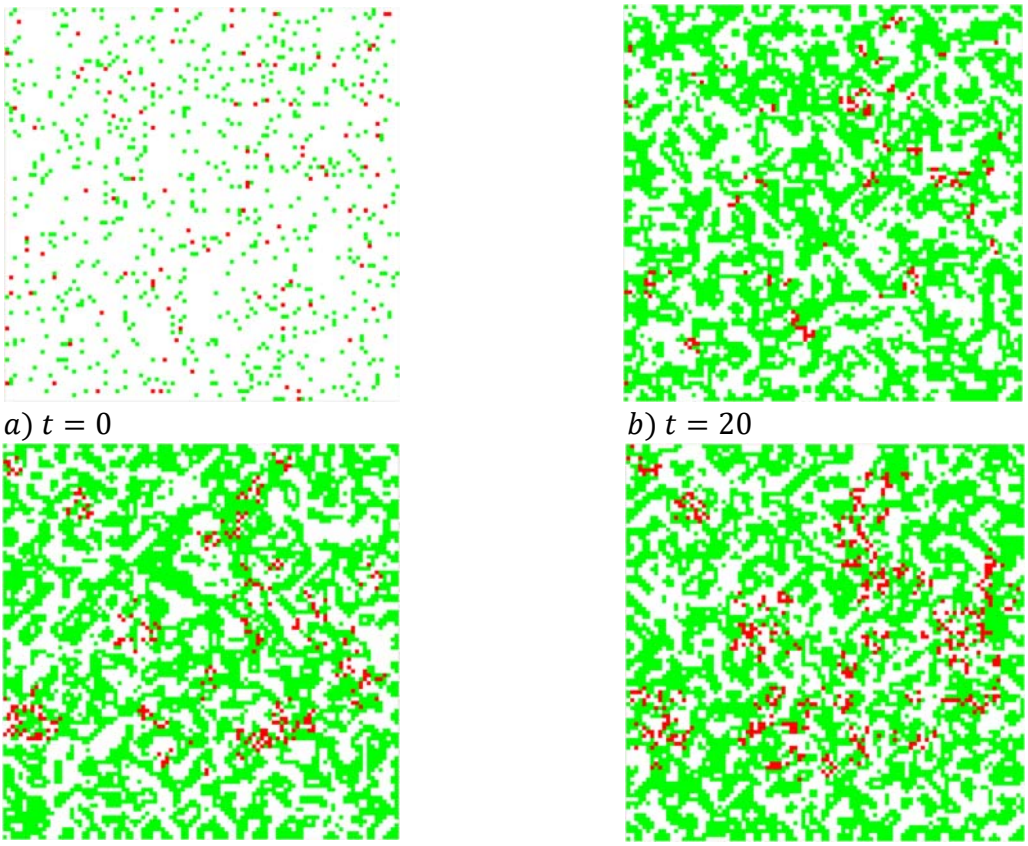
$c)\ t = 100$ $d)\ t = 150$

Figure 4. The states of the predator–prey system displayed at times $t = 0, 20, 100, 150$ *where* $p = 0\%$.

When $p = 0\%$, the foxes cannot reproduce by *Rule 2.3* and in the general case, there aren't enough foxes close to each other in the beginning (initially, they are distributed randomly), so

Rule 3.2 also doesn't apply. In this case, the foxes' population decreases over time until it disappears.
By time $t = 20$, about half of the initial foxes have died. There are only four foxes at time $t = 100$. And at time $t = 150$, there are no foxes left.
There is usually enough food for the foxes, so they survive by *Rule 1.1*. However, as the location of the rabbits varies, there is a probability that at some time, all the neighbours of a certain fox will be empty cells (for example, the fox in the bottom right corner at time $t = 100$). In this case, the fox dies (*Rule 1.2*).
After time $t = 150$, the system reaches a stable state where rabbits are distributed over the whole grid, and the change in their position is determined by Rule 2.1 and Rule 3.1.

2. Case: $p = 10\%$
The results from the simulation when $p = 10\%$ are shown in *Figure 5*.



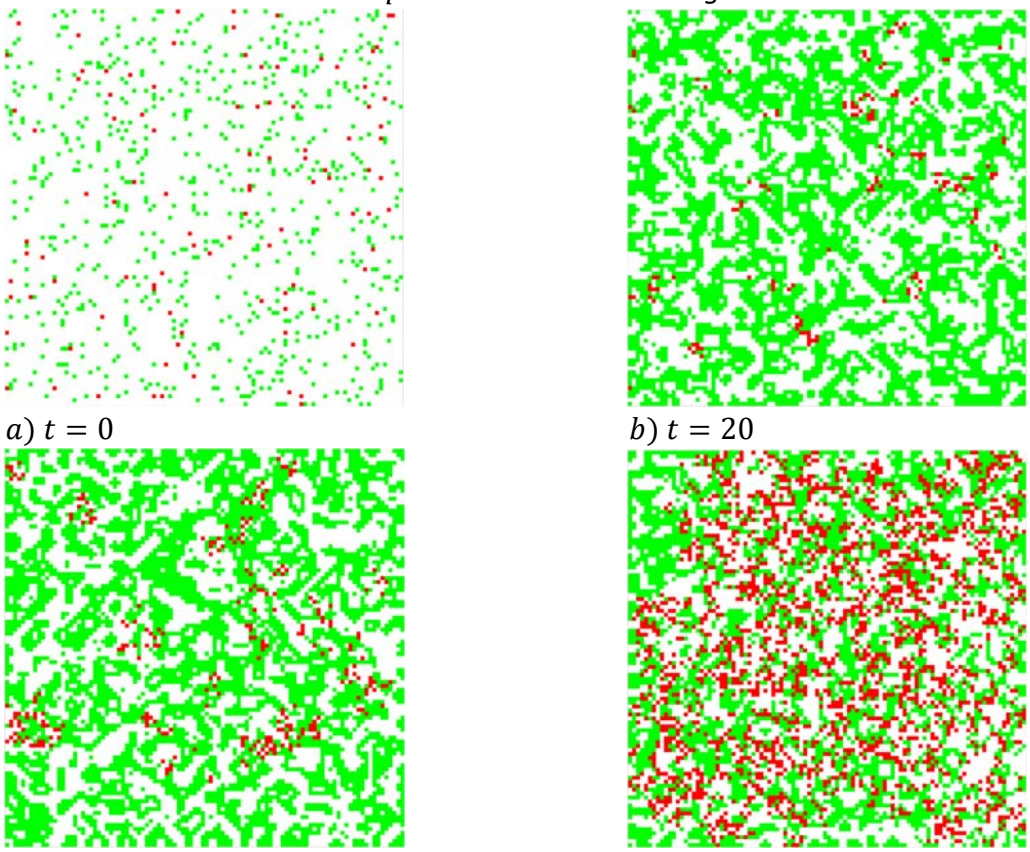
$a)\ t = 0$ $b)\ t = 20$ $c)\ t = 100$ $d)\ t = 150$

Figure 5. The states of the predator–prey system displayed at times $t = 0, 20, 100, 150$ *where* $p = 10\%$.

When $p = 10\%$, the foxes can reproduce by both *Rule 2.3* and *Rule 3.2*. In this case, the foxes' population increases over time until the foxes are distributed over the whole grid.

The foxes begin to reproduce themselves and form clusters of foxes around their initial locations. By time $t = 150$, many of the groups have merged, and after enough time, they will cover the whole grid, and the system will reach a stable state, where the rabbits and foxes occupy the entire grid.

3. Case: $p = 20\%$
The results from the simulation when $p = 30\%$ are shown in *Figure 6*.



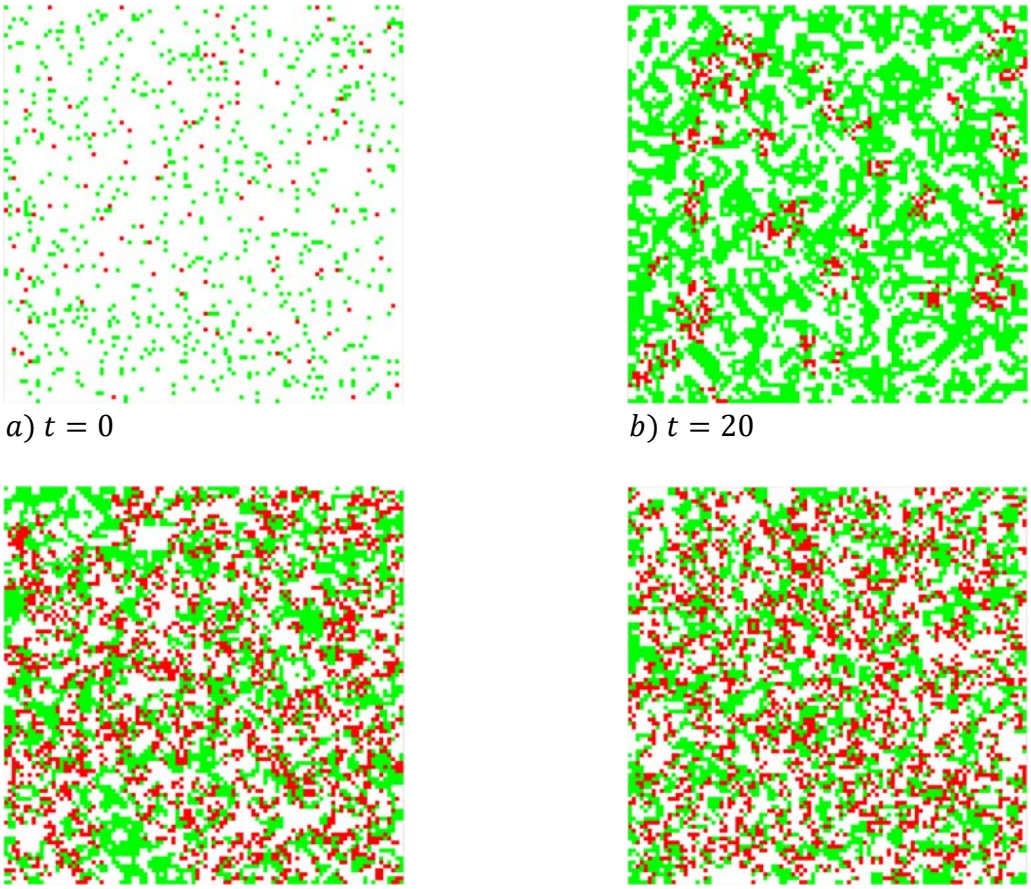
$a)\ t = 0$ $b)\ t = 20$

$c)\ t = 100$ $d)\ t = 150$

Figure 6. The states of the predator–prey system displayed at times $t = 0, 20, 100, 150$ *where* $p = 20\%$.

Increasing p by 10% the rate of expansion of the clusters increases and by $t = 150$ the system has almost reached the stable state.

4. Case: $p = 30\%$
The results from the simulation when $p = 30\%$ are shown in *Figure 7*.



$a)\ t = 0$ $b)\ t = 20$

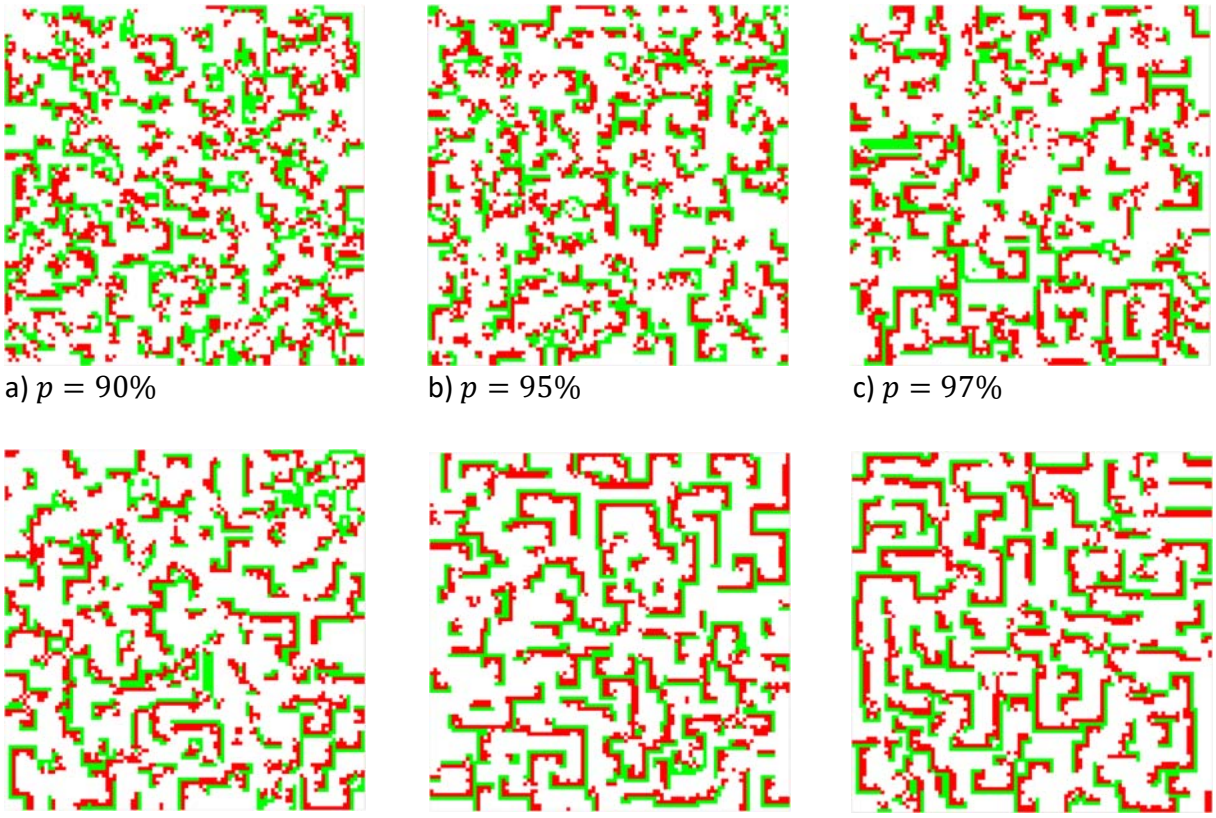
$c)\ t = 100$ $d)\ t = 150$

Figure 7. The states of the predator–prey system displayed at times $t = 0, 20, 100, 150$ *where* $p = 30\%$.

In this case, the system has almost reached the stable state by $t = 100$.
We observe that as we increase p, the time the system reaches a stable position decreases.

5. Case: $p = 90\%;\ 95\%;\ 97\%;\ 99\%;\ 99{,}9\%;\ 99{,}99\%$

The results from the simulation when $p = 90\%;\ 95\%;\ 97\%;\ 99\%; 99{,}9\%;$ $99{,}99\%$ are shown in *Figure 8*.



a) $p = 90\%$ b) $p = 95\%$ c) $p = 97\%$

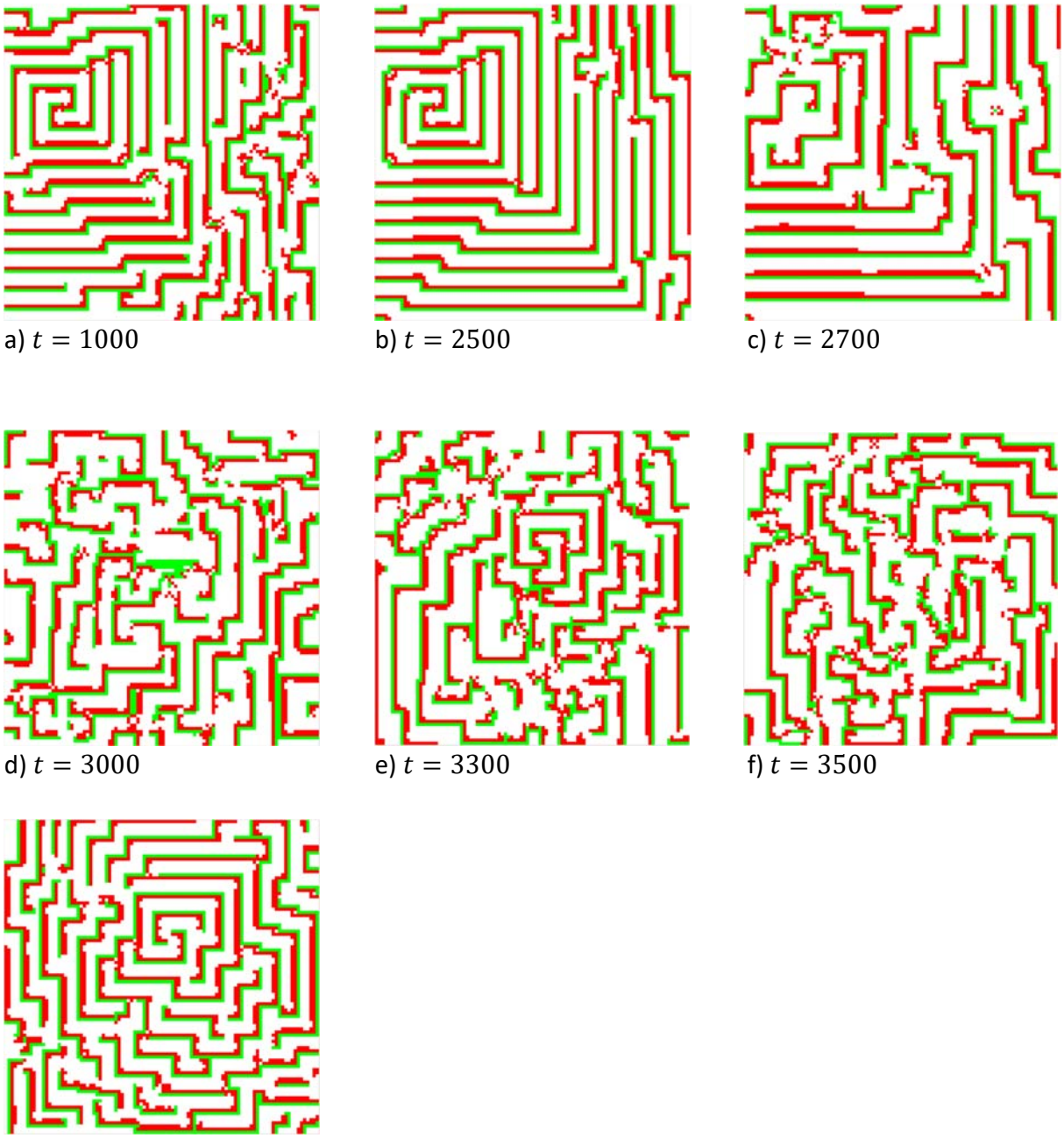
d) $p = 99\%$ e) $p = 99{,}9\%$ f) $p = 99{,}99\%$

Figure 8. The states of the predator–prey system for $p = 90\%;\ 95\%;\ 97\%;\ 99\%; 99{,}9\%;$ $99{,}99\%$ *displayed at* $t = 100$.

When $p \geq 90\%$, we observe the formation of spiral-like patterns that become more and more visible as we increase p. The formation of such spirals in a host-parasite system is also observed and further investigated in *Spiral formation in cellular automata of predator-prey systems* by Menno Rubingh (Rubingh, 2001).

5. Case: $p = 99{,}99\%$

The results from the simulation when $p = 99{,}99\%$ are shown in *Figure 9*.



a) $t = 1000$ b) $t = 2500$ c) $t = 2700$

d) $t = 3000$ e) $t = 3300$ f) $t = 3500$

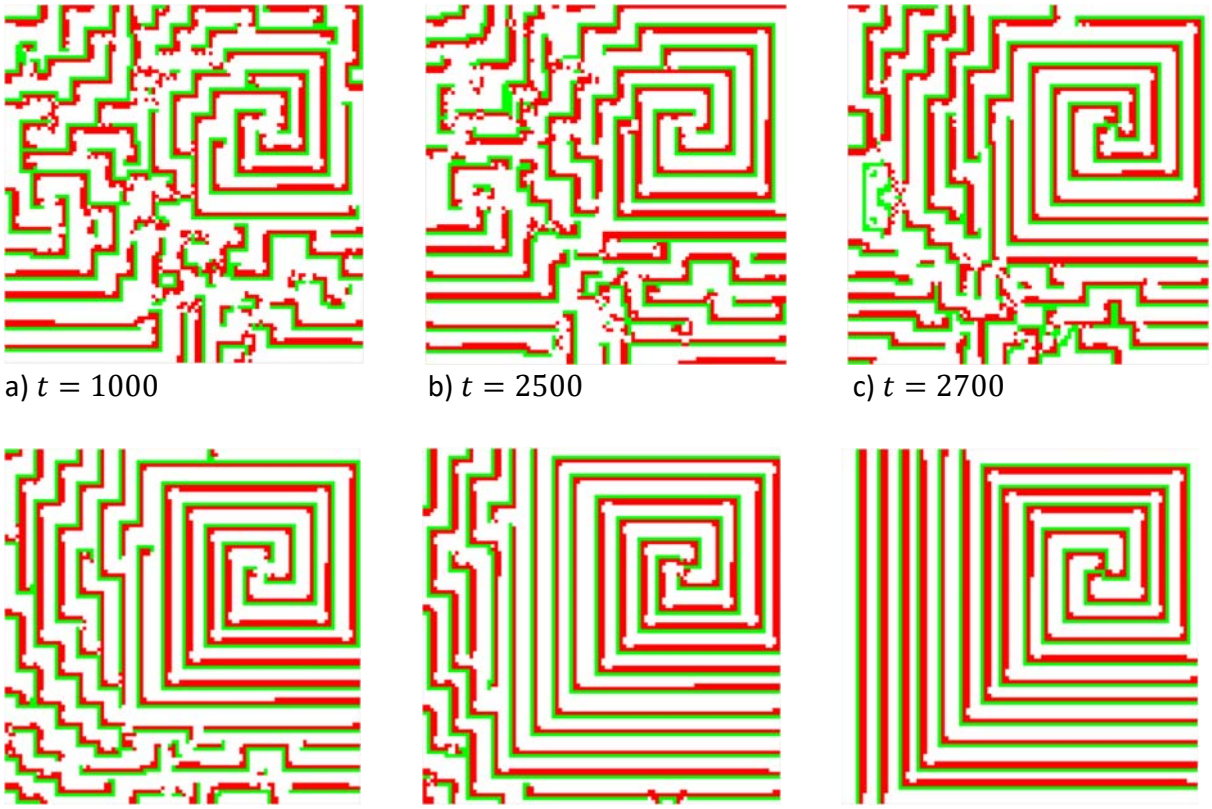
g) $t = 4000$

Figure 9. The states of the predator–prey system displayed at times $t = 1000, 2500, 2700, 3000, 3300, 3500, 4000$ *where* $p = 99{,}99\%$.

When $p = 99{,}99\%$, the system tends to behave almost perfectly when forming spiral-like patterns. However, that $0{,}01\%$ probability induces randomness and makes the system extremely sensitive to even minor fluctuations in its state.

At time $t = 1000$, we observe a spiral forming in the top left corner of the grid. This spiral grows; by time $t = 2500$, it has almost covered the whole grid. However, due to the $0{,}01\%$ probability, at some time after that, *Rule 2.4* is applied in the region of the spiral instead of *Rule 2.3*, which leads to turbulence in the stable state. At time $t = 2700$, the geometry of the spiral is disturbed, and at time $t = 3000$, there is almost no such pattern.
Then, the system continues to strive to form spirals, and at time $t = 3300$, there is a new spiral starting in the top right quarter of the grid. However, this spiral is similarly disrupted, and at $t = 3500$, it is gone. Again, at time $t = 4000$, a new spiral is formed.
This cycle of formation and disruption of the spirals continues to infinity.

6. Case: $p = 100\%$
The results from the simulation when $p = 100\%$ are shown in *Figure 10*.



a) $t = 1000$ b) $t = 2500$ c) $t = 2700$

d) $t = 3000$ e) $t = 3300$ f) $t = 3500$

Figure 10. The states of the predator–prey system displayed at times $t = 1000, 2500, 2700, 3000, 3300, 3500$ *where* $p = 100\%$.

When $p = 100\%$, *Rule 2.4* is never applied instead of *Rule 2.3*, and the system's behaviour depends on fixed rules. Thus, the only induced randomness in the system is the initialization of the state at $t = 0$.

The spiral formed at time $t = 1000$ is never disrupted; by time $t = 3500$, it has covered the whole grid. Then, the system enters a stable spiral state and remains in it over time.

Analysing this stable state, we observe the following pattern of *outward motion* of the spiral:
- the white line next to the green one becomes green in the following generation by *Rule 3.1*
- the green line becomes red in the following generation by *Rule 2.3*
- the red line next to the green one remains red in the following generation by *Rule 1.1*
- the second red line (if there is one) becomes white in the following generation by *Rule 1.2*.

Results

This program fills two files with data – Prey.dat and Predator.dat. Each one of them has two columns – one with the values of t and one with the size of the population at this point. We use the app Gnuplot to plot graphs that show the sizes of the two populations over time shown in *Figure 11*. Note that because there is too much fluctuation in the sizes, at time t, we display the mean value of the population size over the next five generations ($t, t+1, t+2, t+3$, and $t+4$).

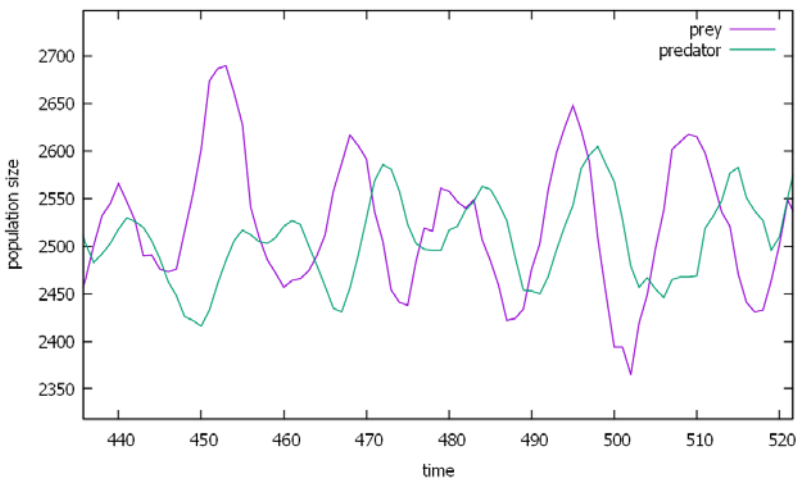


Figure 11. Graph of predator and prey populations over time.

Let us analyse the graph in *Figure 11*.
At time $t = 450$, there is a small number of foxes compared to the number of rabbits. Consequently, a significant portion of the rabbits will survive (*Rule 2.2*), and many new rabbits will be born (*Rule 3.1*), so we observe a nearly exponential growth of the rabbits' population over the next few generations. Because there is enough food for the foxes, almost all of them will survive (*Rule 1.1*), and there will be enough newborn foxes (*Rule 2.3*), so the foxes' population also grows exponentially.

At $time\ t = 454$, the rabbits' population size peaks because it is limited by *Rule 2.1*. Also, the population starts to decrease exponentially because of the large number of foxes (*Rule 2.3*).
At $t = 455$, the foxes' population also peaks and starts to decrease exponentially because of the insufficient number of rabbits (*Rule 1.2*).
At $t = 460$, the rabbits' population hits a low and starts increasing again because of the small number of foxes (*Rule 2.2* and *Rule 3.1*). The foxes' population continues to decrease because the number of rabbits remains insufficient.
At $t = 466$, the foxes' population reaches a low and starts to increase because of the recovery of the rabbits' population.
Then, the state of the predator-prey system returns to the one in the beginning ($t = 450$), and the process begins again. We observe a repeating cycle of increasing and decreasing the two populations.

Discussion

Cellular automata represent a very simplified model of the complex interactions between species. However, they manage to reflect reality quite well. For example, *Figure 12* shows data from the *Hudson's Bay Company* about the sizes of the populations of hares and lynxes over the period 1900–1920 (Mahaffy, 2010). Comparing the real-world data in *Figure 12* with the result in *Figure 11* that we obtained with the cellular automata model, we can see that they match very well – they both show the cycles of growth and shrinking of the populations.

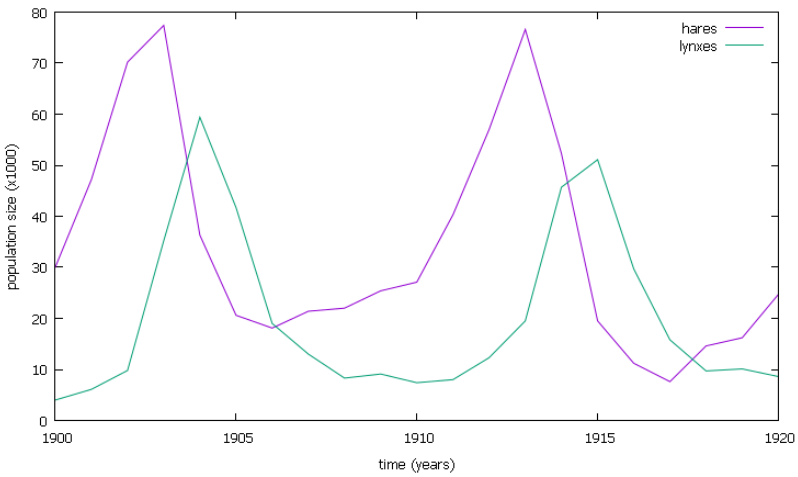


Figure 12. Graph of hares and lynxes' populations over time.

This model relies on several unrealistic assumptions. We ignore factors such as age, health, illness, and competition for territory and outside factors such as interactions with other species, especially humans. Also, the assumption that the food for the rabbits is unlimited doesn't reflect reality. Despite all these weaknesses, the model gives a realistic representation of the interaction between predators and prey.

Conclusion

Cellular automata are a powerful tool for representing a variety of discrete structures. Cellular automata allow simulating many processes, phenomena, and interactions in nature. In our work, we presented a set of rules for the behaviour of the cellular automaton. These rules allowed us to simulate a predator-prey system interaction. By studying the parameters of the cellular automaton, we were able to obtain data from the simulations that were very close to actual data. Therefore, cellular automata allowed us to simulate the predator-prey interaction in a simple yet reliable way.

Reference list

Chen, Q., 2009. Cellular automata. In: S. Jørgensen, T. Chon & F. Recknagel, eds. *Handbook of Ecological Modelling and Informatics.* s.l.:WITpress, pp. 283-306.

Kunis, F. & Dimitrov, M., 2020. Investigating Lotka-Volterra model using computer simulation. *Open Schools Journal for Open Science,* III(10).

Mahaffy, J. M., 2010. *Lotka-Volterra Models.* [Online]
Available at: https://jmahaffy.sdsu.edu/courses/f09/math636/lectures/lotka/qualde2.html
[Accessed 11 May 2022].

Rubingh, M., 2001. *Spiral formation in cellular automata of predator-prey systems.* [Online]
Available at: https://www.rubinghscience.org/evol/spirals1.html
[Accessed 11 May 2022].

Shiffman, D., 2012. *The Nature of Code: Simulating Natural Systems with Processing.* 1st ed. s.l.:s.n.