Undergraduate Projects in the Application of Artificial Intelligence to Chemistry. III. Cellular Automata

Hugh M. Cartwright* and Lisa T. Yiasoumis

Physical and Theoretical Chemistry Laboratory, Oxford University, South Parks Road, Oxford OX1 3QZ, England, Hugh.Cartwright@chem.ox.ac.uk

Abstract: Cellular Automata are widely recognised as the basis of "The Game of Life". It is less widely known that they can be used to study a variety of scientific problems. This paper outlines the mode of operation of a Cellular Automaton, and illustrates its application to the simulation of bacterial growth on a virtual Petri dish.

1. Introduction

It is not often that a computer screen saver can be used to solve problems in science, but this is the case with the technique described in this paper—cellular automata. The cellular automaton (CA) is widely known as the basis of the "Game of Life," used as a screen saver on many computers; however, this technique from the field of artificial intelligence (AI) is much more than just entertainment. CAs have evolved from a pencil-and-paper exercise into a valuable simulation tool. This paper, the third in a series on the way in which artificial intelligence may be used in science [1, 2], outlines how CAs work and how they may be used to simulate and help us understand such processes as the growth of bacteria or the development of snowflakes.

2. What are Cellular Automata?

The Hungarian mathematician John Von Neumann invented the cellular automaton in the late 1940s; however, it was not until two decades later, at the University of Cambridge in 1970, that the eccentric mathematician John Horton Conway conceived the most well-known CA, the Game of Life. In this paper we consider the Game of Life briefly, but our main focus is on how the technique can be used to solve problems in science.

Cellular Automata provide a means by which we can model computationally the development of evolving systems, such as the growth of snowflakes or bacterial colonies, without needing to use complicated equations. They are, in the words of Stephen Wolfram [3], "examples of mathematical systems constructed from many identical components, each simple, but together capable of complex behavior."

This feature, complex behavior resulting from the interaction of many simple units, is typical of several AI methods. While Wolfram's statement may give the impression that CA is a complicated technique, its principles are in fact very easy to understand. It is possibly the most straightforward area related to artificial intelligence in which to write computer programs.

3. Cellular Automata—the Mechanics

Imagine a set of boxes positioned in a regular arrangement. Most commonly we position the boxes in a two-dimensional square grid (Figure 1), but alternative geometries such as two-dimensional triangular or hexagonal grids, and one-dimensional or three-dimensional grids are possible (Figure 2), although less frequently used.

We shall call each box a *cell*. Each cell has one key property associated with it, known as its *state*, whose value indicates the current condition of the cell. The cell's state often has a simple physical interpretation; for example, the cell might be described as filled or empty, or it might be alive or dead.

At the start of a CA calculation, the states of all cells are defined by the user. These states can be chosen in whatever way he or she chooses, but for a scientifically informative calculation, they should be selected with some goal in mind. Then, in a series of generations or cycles, the state of every cell is updated repeatedly according to predetermined rules. The same set of rules is applied to every cell.

The rules, which determine the state of each cell in one generation from the state of that cell and those around it in the immediately preceding generation, are normally very simple. The changes in the state of a single cell as the generations pass are therefore not complex; however, it is not the changes that occur in one particular cell which are of interest, but the way in which the state of *groups* of cells change. As we shall see, this behavior of clusters of cells is informative and sometimes beautiful.

In the simplest model, the state of each cell is either "1" (on) or "0" (off). To help follow the evolution of a CA on a computer screen, cells in different states are generally drawn at their positions in the grid in a color determined by their state, so that one can readily follow the progress of the algorithm. The CA then evolves, cycle by cycle, the cells in each generation changing their states (and therefore their colors) according to the rules that have been set by the user.

Von Neumann's CA, conceived nearly 60 years ago, was very complex, with 200,000 cells, each in one of 29 states—far beyond the capabilities of computers in the mid-20th century to deal with. Conway thought that it must be possible to devise a simpler CA that was still capable of complex behavior. His application, the Game of Life, is a convenient illustration of how a CA operates and will be familiar to many readers.

In any CA the rules specify how the state of a cell in one generation is determined from its state, and that of its neighbors in the previous generation; so, we need to define what is meant by the term "neighbors." Conway defined a neighborhood of eight cells that surround a particular target

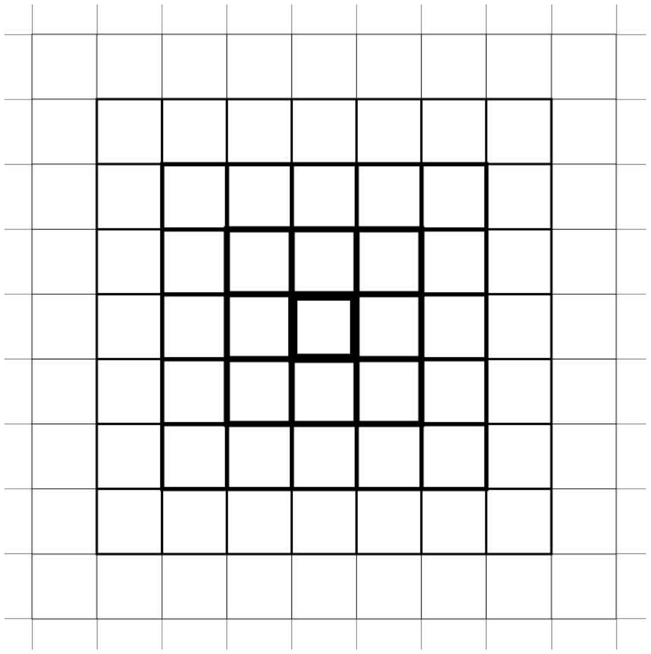


Figure 1. The most common cell geometry for a CA calculation, a large rectangular or square grid.

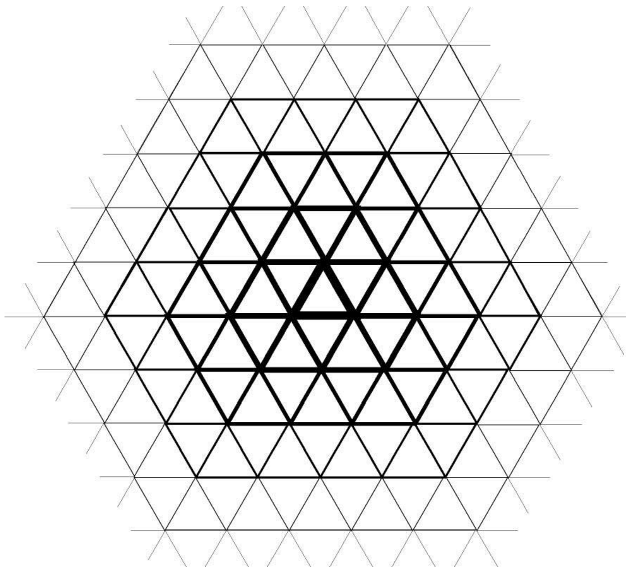


Figure 2. An alternative cell geometry for a CA calculation.

cell (Figure 3) and three rules to determine how the state of that cell changes from generation to generation:

Conway's rules are:

1. If a target cell has two neighbors, it retains its current state in the next generation.

2. If a target cell has three neighbors, the state of the cell will be "on" in the next generation.

3. If a target cell has fewer than two or more than four neighbors that are "on" the state of the cell will be "off" in the next generation.

When these simple rules are applied repeatedly to the cells in a square or rectangular array, a variety of complex patterns may develop, whose form is determined unambiguously by the

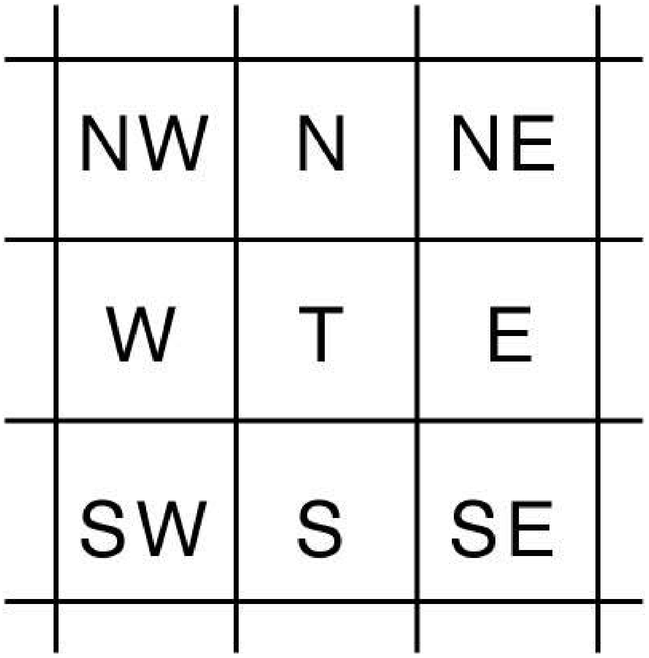


Figure 3. Conway's neighborhood around a target cell.

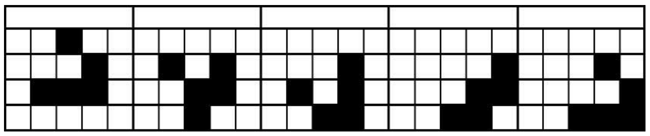


Figure 4. The movement of a glider over five generations during the Game of Life.

initial arrangement of occupied cells. Starting patterns in Conway's Game of Life that differ only slightly from one another can produce quite different behavior as generations pass. Most patterns die out, or reach stable (or oscillatory) forms; a small number grow indefinitely or generate patterns of cells in the "on" state that move across the grid. The most well-known example of a traveling pattern has been termed a *glider*. Four generations are needed for the glider to recreate its original pattern, by which time it has been displaced by one square diagonally (Figure 4).

4. A Scientific Application of the Cellular Automaton

A generation of computer science students has toyed with the Game of Life, but while the creation of patterns on a computer screen may be entertaining, it does not of itself make the CA useful. An Italian Ph.D. student, Tommaso Toffoli, argued, "the importance of cellular automata lies in their connection with the *physical world* " [11]. This brings us to the CA's role in science.

During the early development of the CA it was not obvious to most people (perhaps not even to Conway) that this technique could be of value in the modeling of physical or biological systems and thus help to rationalize the behavior of such systems. Such has proven to be the case, however. Conway proposed a particular set of rules that determined the evolution of his CA, but there is no "correct" set of rules by which every CA *must* evolve. *Any* set of rules may be proposed which, given the state of a cell and its neighbors in one generation, determines the state of that cell in the next generation. Nevertheless, there is no certainty that any particular rules will lead to interesting or useful behavior.

To construct a CA of predictive value, the effects of the local environment on the target cell must be considered carefully and rules constructed accordingly. It is essential to identify those variables and properties that are important and to relate them to the physical system being modeled so that understanding of the system is maximized. This often means that some logical and clearly apparent relationship must exist between the form of the rules and the physical process being modeled, but this is not always the case.

The example that will be used to explain both the principles behind the CA and the coding of a typical CA program is the growth of bacteria on a Petri dish in the presence of limited supplies of nutrient. Because of the ubiquitous nature of bacteria and the way in which they—quite unexpectedly—show cooperative behavior, this is an example of some importance.

Bacteria are vastly more numerous in the world than are humans. The behavior of an individual bacterium is comparatively simple, but clusters of bacterial cells can show complex cooperative behavior as a response to environmental stress, such as the depletion of a food source or the introduction of a toxic chemical into their environment. We note that this behavior, which is simple on the individual level but more complex on the group level, is just the kind of behavior that the CA exemplifies, so it should be no surprise that it is possible to mimic this behavior and to an extent interpret it using a CA.

Most people are familiar with the compact circular patterns typically produced by a bacterial colony on a standard Petri dish containing high concentrations of agar as a support medium and excess peptone as the nutrient source.

It has been noted [4] that colonies of certain types of bacteria, for example, *Bacillus Subtilis*, grow in forms that exhibit a variety of branching and fractal patterns when exposed to unfavorable conditions, such as low agar and peptone levels. A CA can be used to simulate this growth and produce patterns similar to those seen experimentally [4, 5].

4.1. The Grid of Cells. To model a bacterial colony using a CA, we first construct a grid containing a large number of cells in a regular array. Each position in the grid represents a location, on a virtual Petri dish that may potentially be occupied by a single bacterial cell. A 500-by-500-square grid (250,000 cells) is a sufficiently large number of locations to allow us to investigate the behavior of a simulated bacterial colony in some detail.

A real bacterial colony can be initiated from just one cell, which then divides to form further cells, and so the colony grows by apposition of cells at the periphery [6, 7]. A virtual CA colony can be formed in a similar fashion from a small starting group of cells. Conway's two-state system is a little too simple to simulate bacterial growth, because a minimum of three states is required. Gird positions may be occupied or unoccupied, and the bacteria in filled cells may be alive or dead. Accordingly, the state of each cell in our simulation can be set as 0 (empty), 1 (occupied and alive), or 2 (occupied and dead).

4.2. Factors Influencing the Growth Rate. Several factors may affect the rate and form of expansion of a real bacterial colony, and it is essential to consider how these factors affect growth, because an understanding of them will allow us to tailor the rules for a virtual colony, so that real behavior can be predicted. Cell growth depends on the environment, which changes as the colony evolves; therefore, growth is determined by the dynamic interaction of the cells with the medium in which they grow [8]. The primary factors that influence this growth are the peptone and agar concentrations and the density of bacterial cells.

4.2.1. The Peptone Concentration. Bacteria need peptone as a nutrient to survive and reproduce, so the ambient concentration of peptone and its rate of uptake by new cells (for growth) and by mature living cells (for sustenance) is a primary factor determining the rate at which the colony can develop. It follows that the peptone concentration in each grid position needs to be monitored by the algorithm. Thus, it is clear that in this application (and in other similar applications) we need to keep a record of not just the state of each cell, but also some ancillary parameters.

4.2.2. The Agar Concentration. The medium within which the bacteria on a Petri dish grow is normally agar. On a soft agar plate (that is, one in which the agar concentration is low), nutrient diffuses faster than on one in which the agar concentration is high. There is, therefore, a complex interaction between the nutrient (which is continually consumed by the living bacteria), the agar (which provides a medium whose density determines the rate at which fresh nutrient can diffuse to the colony), and the growth of the colony itself. We shall assume that the agar concentration is invariant.

4.2.3. The Number of Other Living Cells in the Neighborhood. If a bacterium is overcrowded (and thus finds itself competing with other bacteria for nutrient) or exposed (and so is more likely to be affected by adverse conditions in the environment, such as the presence of toxic chemicals), then it is likely that it will not survive, or at very least it is likely to have a shorter life than bacteria finding themselves in a more favorable environment. It follows that there is an optimum number of active cells surrounding an empty cell for a new bacterium to form and flourish.

In the CA described in this paper each of these factors is taken into account in order to produce a working model.

4.3. A Working CA Program to Simulate Bacterial Growth. To simulate the development of a bacterial colony, several arrays are required to store information about the conditions at each position in the grid.

The state of each cell is clearly an essential parameter, because the entire algorithm is directed towards an investigation of the evolution of these state values. At the start of the program, the state array is initialized to "0" (empty) in the manner shown below for the Java applet used to create the figures in this paper.

```
for (row = 0; row <= 499; row++) {
for (col = 0; col <=499; col++) {
readGrid[col][row] = 0;

}
}
```

The peptone grid, which stores the amount of nutrient available in each cell, is initialized to "1" in a similar fashion.

Because the bacterial colony cannot be created out of nothing, the state of a small group of cells near the center of the grid is set to 1 (filled, alive) to provide a nucleus colony from which bacterial growth is initiated.

```
        readGrid[249][249] = 1;
        readGrid[250][249] = 1;
        readGrid[249][250] = 1;
        readGrid[250][250] = 1;
```

A record must also be maintained of the generation in which a grid position was first occupied by a bacterium. This is required as each bacterium has a finite lifetime, and once this lifetime has been exceeded, the cell dies. Further, the age of the bacterium determines the quantity of nutrient it will absorb each cycle.

Each generation, the following steps are performed which determine how the CA evolves.

4.3.1. Uptake of Nutrient. Live cells consume a small amount of food for sustenance each generation, so at every grid position occupied by a live bacterium the peptone concentration is diminished by the appropriate amount as each generation passes. Fresh bacteria, that is, those created in the last generation, consume a greater quantity of food than mature bacteria to account for their new growth. So, at each generation, the quantity of peptone at each cell on the grid is adjusted according to the state of the cell.

```
for (row = 1; row <= 498; row++) {
for (col = 1; col <=498; col++) {
if(readGrid[col][row]== 1 && genGrid[col][row]==(gen-1))
peptoneGrid[col][row]= (peptoneGrid[col][row]-eaten);
if(readGrid[col][row] == 1 && genGrid[col][row]!=(gen-1))
peptoneGrid[col][row]=(peptoneGrid[col][row]-eaten/2.0);
}
}
```

4.3.2. The Crowding Factor. At each position in the grid, a crowding factor, *S*, is calculated. This will be required shortly to determine whether bacteria in filled cells will die through overcrowding, or whether vacant grid positions should become birth cells. The factor *S* is incremented by one for every living neighboring cell that is less than two generations old.

```
for (row = 1; row <= 498; row++) {
for (col = 1; col <=498; col++) {
s = 0;
if(readGrid[col-1][row]==1 && genGrid[col-1][row]>=gen-2) s = s+1;
if(readGrid[col][row-1]==1&& genGrid[col][row-1]>=gen-2) s = s+1;
if(readGrid[col][row+1]==1&& genGrid[col][row+1]>=gen-2) s = s+1;
if(readGrid[col+1][row]==1&& genGrid[col+1][row]>=gen-2) s = s+1;
if(readGrid[col+1][row+1]==1&&genGrid[col+1][row+1]>=gen-2) s = s+1;
if(readGrid[col-1][row-1]==1&&genGrid[col-1][row-1]>=gen-2) s = s+1;
if(readGrid[col-1][row+1]==1&&genGrid[col-1][row+1]>=gen-2) s = s+1;
if(readGrid[col+1][row-1]==1&&genGrid[col+1][row-1]>=gen-2) s = s+1;
// s is stored at this point.
}
}
```

4.3.3 Creation of New Cells. The number of live cells in the neighborhood of every grid position has now been determined. As proposed above, if a cell is empty, but has between two and four live neighbors, a new bacterium can potentially be created in this birth cell. If the number of neighbors is suitable and the peptone concentration exceeds a defined threshold (in this case 0.8), a new cell is created with a defined probability (0.4 here). This stochastic (probabilistic) factor for the creation of a new bacterium is introduced because real bacterial growth is not entirely deterministic; even if the conditions are right for growth the production of a new bacterium does not occur with complete predictability. We also recognize that there is some uncertainty in the rules proposed, which cannot exactly mimic reality.

```
if ((readGrid[col][row]==0) && (s>=2 && s<=4 ))
neighbourGrid[col][row]= 1;
if(neighbourGrid[col][row]==1) {
if((peptoneGrid[col][row]>=0.8)&&(z1*peptoneGrid[col][row])>=0.4{
writeGrid[col][row]=1;
genGrid[col][row]=gen+1;
}
}
```

4.3.4. Destruction of Old Cells. Like all living things, bacteria do not last forever. If a cell in state "1" has exceeded its age limit, then the cell will "die" into state "2". Similarly, if a live cell finds itself entirely surrounded by other cells, it dies as a result of overcrowding.

```
if ((readGrid[col][row]==1) && (s==8))
neighbourGrid[col][row]= 2;
```

In addition, if the food supply runs low the bacterium is increasingly likely to starve.

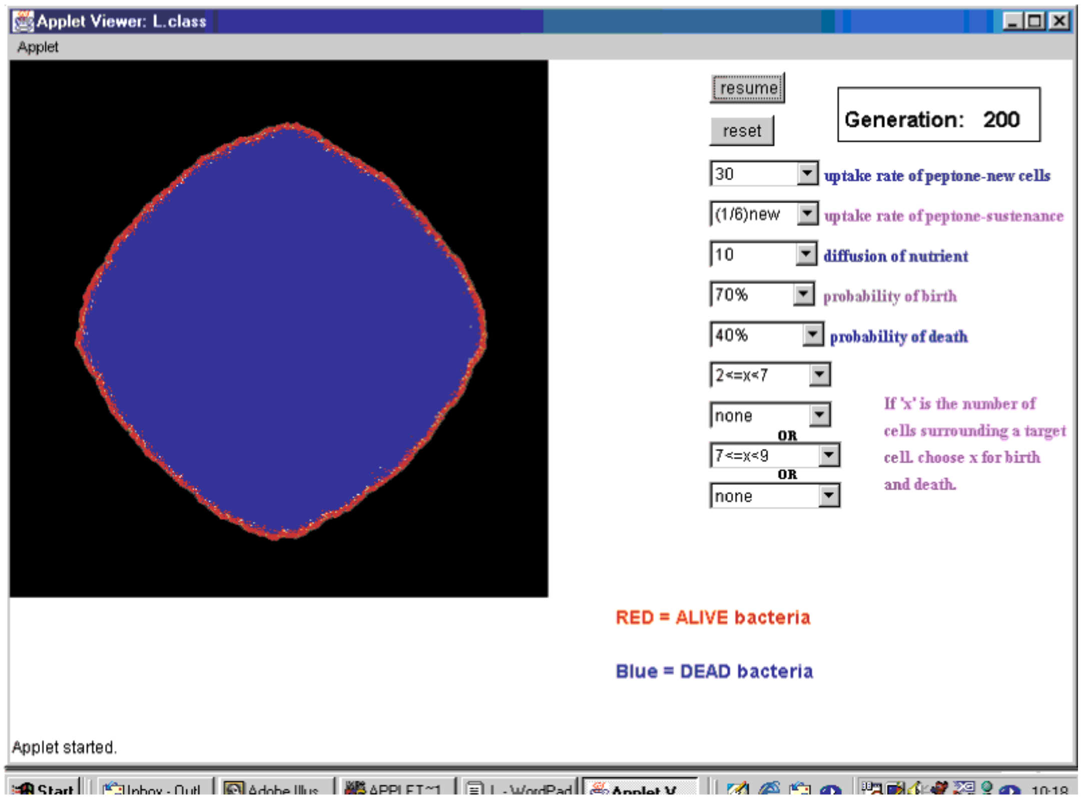
```
if((peptoneGrid[col][row]<=0.2) &&
(z1*peptoneGrid[col][row])<=0.5)
writeGrid[col][row]=2;
```

4.3.5. Diffusion of Food. As peptone is consumed by the growing bacteria, a nutrient concentration gradient develops. Peptone will then diffuse within the "Petri dish" from regions of high concentration to regions in which it is at lower concentration; for this, a simple diffusion model is used. The diffusion rate is dependent upon the agar concentration so this must also be taken into account. In this example, we use a simple linear diffusion model, which is sufficiently accurate if the diffusion gradients are not too great. This diffusion calculation is repeated for the cells north (N), south (S), east (E), and west (W) of the target cell; in other words, those which directly abut it.

```
for (row = 1; row <= 498; row++) {
for (col = 1; col <=498; col++) {
if((peptoneGrid[col-1][row]-peptoneGrid[col][row])>0) {
difference=peptoneGrid[col-1][row]-peptoneGrid[col][row];
peptoneGrid[col][row]= peptoneGrid[col][row]+
((difference)/(500*agarConc));
peptoneGrid[col-1][row]= peptoneGrid[col-1][row]-
 ((difference)/(500*agarConc));
}
```

4.3.6. The Graphics. In the program described here (and accessible through the web site [12]), the entire grid is painted onto an off-screen image once the new set of states has been calculated and then to the screen.

```
      for (row = 1; row <= 498; row++) {
      for (col = 1; col <=498; col++) {
            if(peptoneGrid[col][row]!= 1) {
                  offscreen.setColor(Color.white);
                  offscreen.fillRect(col,row,1,1);
            }
            switch(readGrid[col][row]) {
                  case 0:
                  if(peptoneGrid[col][row]==1) {
                        offscreen.setColor(Color.black);
                        offscreen.fillRect(col,row,1,1);
                        break;
                  }
                  else
                  break;
                  case 1:
                  offscreen.setColor(Color.magenta);
                  offscreen.fillRect(col,row,1,1);
                  break;
                  case 2:
                  offscreen.setColor(Color.blue);
                  offscreen.fillRect(col,row,1,1);
                  break;
            }
      }
}
```



Generation: 200

30 uptake rate of peptone-new cells

(1/6)new uptake rate of peptone-sustenance

10 diffusion of nutrient

70% probability of birth

40% probability of death

2<=x<7

none
OR
7<=x<9
OR
none

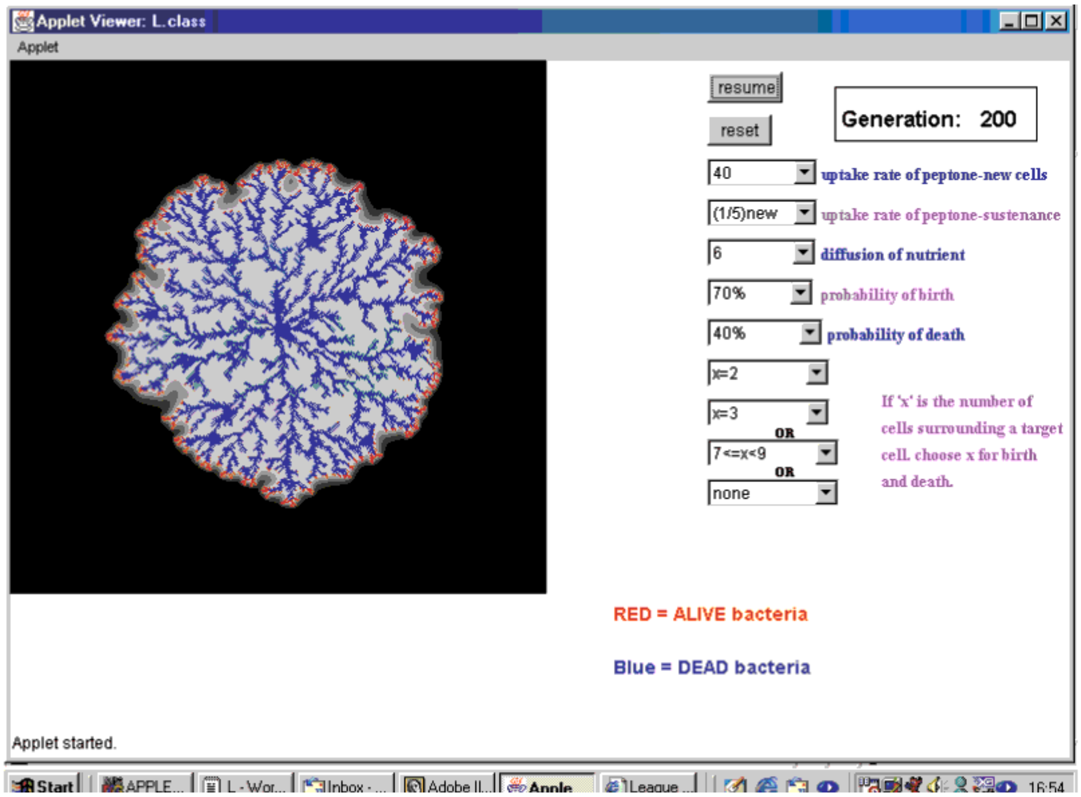
If 'x' is the number of cells surrounding a target cell, choose x for birth and death.

RED = ALIVE bacteria

Blue = DEAD bacteria

Applet started.

Figure 5. A CA-simulated bacterial colony (I). Occupied cells are shown in blue, birth cells (in which a new bacterium has just been created) are shown in red, and nutrient is shown in black. The colony has a roughly circular form and, as the red points indicate, is growing strongly in all directions away from the center of the colony.



Generation: 200

40 uptake rate of peptone-new cells

(1/5)new uptake rate of peptone-sustenance

6 diffusion of nutrient

70% probability of birth

40% probability of death

x=2

x=3
OR
7<=x<9
OR
none

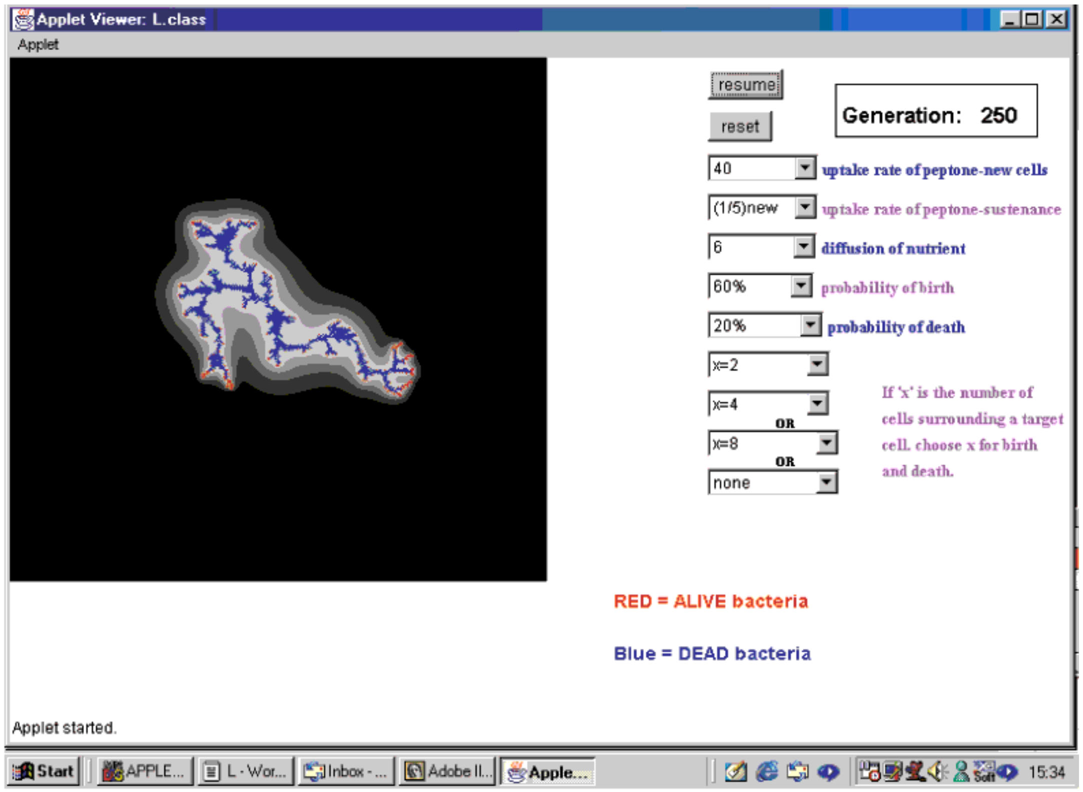
If 'x' is the number of cells surrounding a target cell, choose x for birth and death.

RED = ALIVE bacteria

Blue = DEAD bacteria

Applet started.

Figure 6. A CA-simulated bacterial colony (II). This figure illustrates the effect which depletion of nutrient may have on the growing colony. As the growing points of the colony (shown in red) move outwards, the diminution of nutrient (shown in shades of gray) is often sufficient to prevent the formation of new bacterial cells at the outer edge of the colony, and the colony then develops in a fractal fashion. This is an example of diffusion-limited aggregation.



Generation: 250

40 uptake rate of peptone-new cells

(1/5)new uptake rate of peptone-sustenance

6 diffusion of nutrient

60% probability of birth

20% probability of death

x=2

x=4

OR

x=8

OR

none

If 'x' is the number of cells surrounding a target cell. choose x for birth and death.

RED = ALIVE bacteria

Blue = DEAD bacteria

Applet started.

Figure 7. A CA-simulated bacterial colony (III). In this system, the probability of birth is comparatively low, and around most cells, the level of nutrient falls sufficiently rapidly that a new cell is never formed. In this system, there is a delicate balance between the chance that a bacterial cell will form and the chance that nutrient will run out before a new cell appears. Thus, there are few new growing points in the colony, and a sparse structure is formed. There is a good chance that such a colony will eventually die out.

5. Comment

When the program is run, complex patterns, whose form depends intimately upon the choice we have made for the values of adjustable parameters, may emerge. These patterns resemble those produced by bacteria growing on a Petri dish under conditions of varying degrees of stress. The patterns may be dense or sparse, depending on the conditions.

The behavior of the bacterial colony is ultimately dependent upon the rules chosen and factors such as the quantity of nutrient available, its rate of diffusion through the medium, whether concentration gradients exist at the start of the simulation, and so on. By varying these parameters, it is possible to generate colonies that show a variety of forms, ranging from those in which the bacterial growth resembles the "normal" round colonies (Figure 5) to those which exhibit branching or fractal patterns (Figures 6 and 7).

Because of the stochastic element in the calculation, repeated runs of the program using identical parameter sets may generate colony shapes that differ in detail. Nevertheless, important features of the colony, such as its fractal dimension and overall rate of growth, vary little from run to run. By studying how the growth patterns depend upon the adjustable parameters, it is possible to gain an understanding of the way in which bacterial growth depends upon changes in the environment of a colony.

6. The use of the CA in Science

Cellular automata have been used in science in a variety of ways; three are briefly described below.

6.1. Wolfram's Shells. Stephen Wolfram investigated a one-dimensional CA in which each cell touched only two other cells, one on either side, and each generation was displayed on a computer screen directly below its predecessor. A cell in generation *n* would determine its state by looking at the cell directly above it, in generation (*n*-1), and that cell's two neighbors. Wolfram, who had a collection of conch-like seashells, noticed that the patterns on the shells were remarkably similar to the scattered inverted triangles sometimes formed by the one-dimensional CA. Wolfram's explanation for this observation was that the growth of these pigmentation patterns follows cellular automaton rules, a simple example of self-organization [10].

6.2. Packard's Snowflakes. Norman Packard found that a two-dimensional CA in which the states "0" and "1" represented cells containing water vapor and ice, respectively, could be used to produce snowflake-like patterns. Typically, rules were based on whether the number of surrounding cells in state "1" was odd or even. The patterns that were generated by these rules were not quite as complex as real snowflakes, but they did have similar features and were easily recognizable as snowflakes [9].

6.3. Bacterial Growth. The bacterial growth model outlined in this paper is the subject of ongoing research at Oxford, where we are considering how external influences (both benign and adverse) may influence the rate and form of bacterial growth. Further details will be published in due course.

7. Conclusions

The concept of a Cellular Automaton is both easy to understand and to program, and it can be applied in a straightforward fashion to certain types of problems, those which can be cast in the form of evolving systems. The patterns that CAs produce often resemble experimental results to a remarkable degree. The method has advantages in speed, visualization, transparency, and predictability over more traditional methods of modeling and simulation where complicated and challenging equations may be required. The CA approach is a self-contained method that is particularly useful when attempting to reproduce images that occur as a result of physical processes, both in nature and in the laboratory.

The fascination of the CA is that such a simple methodology and concept can produce such complex, useful, and visually striking results. Web access to an interactive version of the program used to produce the figures of bacteria in this paper, is available [12].

References and Notes

1. Cartwright, H. M. Undergraduate Projects in the Application of Artificial Intelligence to Chemistry. I. Background. *Chem. Educator* **1999,** *4*, 238–241; DOI 10.1007/s0008979900335a.
2. Cartwright, H. M. Undergraduate Projects in the Application of Artificial Intelligence to Chemistry. II. Self-Organizing Maps. *Chem. Educator*, **2000,** *5,* 196–205; DOI 10.1007/s00897000400a. DOI 10.1007/s00897000400a.
3. Wolfram, S. Cellular Automata as Models of Complexity. *Nature* **1984,** *311,* 419–424.
4. Ohgiwari, M.; Matsuyama, T. Morphological Changes in Growth Phenomena of Bacterial Colony Patterns**.** *J. Phys. Soc. Japan* **1992,** *61*, 816–822.
5. Ben-Jacob, E.; Shmueli, H.; Shochet, O.; Tenenbaum, A**.** Adaptive Self-Organization during Growth of Bacterial Colonies. *Physica A*, **1992,** *187,* 378–424.
6. Pirt, S J. A. Kinetic Study of the Mode of Growth of Surface Colonies of Bacteria and Fungi. *J. Gen. Microbiology* **1967,** *47*(2), 181–97.
7. Peters, A. C.; Wimpenny, J. W. T. A Constant-Depth Laboratory Model Film Fermentor. *Biotechnol. Bioeng*. **1988,** *32*(3), 263–70.
8. Schindler, J. Rovensky, L. A Model of Intrinsic Growth of a Bacillus Colony. *Binary* **1994,** *6,* 105–108.
9. Stephen Wolfram. http://www.stephenwolfram.com/ (accessed May 2001).
10. Wolfram, S. Cellular Automata. *Los Alamos Science*, **1983**, 9*,* 2–21.
11. Toffoli, T. *Cellular Automata Mechanics: a new environment for modeling.* MIT Press, Cambridge, Mass. 1987.
12. http://physchem.ox.ac.uk/~hmc/papers/chedr2001a/chedr2001aindex.html (accessed July 2001).