1. History

Cellular automata were initially conceived of in 1948 by John von Neumann who was searching for ways of modeling evolution. He was trying to find out what primitive interactions were necessary for the development of complex forms like those of living organisms. Von Neumann, father of the Von Neumann architecture, which forms the basis of modern digital computers, also believed that if he could simulate simple organisms, he could use this same technology to develop a system capable of emulating the human brain. Von Neumann had the idea of using discrete dynamics which at the time was quite revolutionary and was able to construct a two dimensional automata capable of self-reproduction. Due to the fact that cellular automata require computers for simulation, little work was done in the field until computers became more widespread. In 1970, John H. Conway came up with the game of life, which we will look at in a bit. This simple cellular automata was able to demonstrate quite complicated dynamics which look much like those of microorganisms. First we need some mathematical background before we explore further.
Let $L$ be a discrete lattice of cells, where our cellular automata can live. This lattice can be thought of as a line with tick marks if we are working in one dimension, and as a grid in two, three, or higher dimensions. The structure does not have to be rectangular actually, it could be hexagonal, octagonal, or random, but a grid is usually a good way to visualize it in most cases. Each cell in the lattice, meaning each point on the grid, has a finite number of states it can be in at any time. For the moment let us consider the one dimensional case. We can write

$$\sigma_i(t) \in \Sigma = \{0, 1, 2 \cdots k - 1\}$$

where $\sigma_i(t)$ is the state of the $i^{th}$ cell at time $t$. In most simple cases we take $\Sigma$, which is the possible states to be integers modulo $k$, $\mathbb{Z}_k$. It is assumed that the cellular automata live in an infinite space so in order to model them we use periodic boundary conditions meaning we have say $N$ cells and we associate the $N + 1^{th}$ cell with the $1^{st}$ cell, etc. So we have essentially wrapped our infinite line into a circle. In two dimensions we take our grid and bend it to form a torus.

Since we want our automata to do something we need a way to make them change states. So we have a rule

$$\theta: \underbrace{\Sigma \times \Sigma \times \cdots \Sigma}_n \rightarrow \Sigma$$

where $n$, is the number of cells in a neighborhood. A neighborhood of a cell is exactly what you would think it is, it is some number of surrounding cells. we let $\mathcal{N}\{i\}$ denote the neighborhood of the $i^{th}$ cell.
We can rewrite the rule as

\[ \sigma_i(t + 1) = \theta(\sigma_j \in \mathcal{N}\{i\}). \]

in one dimension this becomes

\[ \sigma_i(t + 1) = \theta(\sigma_{i-r}(t), \cdots, \sigma_i(t), \cdots, \sigma_{i+r}(t)), \quad \theta: S^{2r+1} \rightarrow S \]

So what we have defined is a system of cells on a grid, who have a state at each point in time. As time moves forward the state of each cell changes based on its state and the state of all surrounding cells.

For example, let us assume \( \Sigma = \{0, 1\} \), so we are working in \( \mathbb{Z}_2 \). We will let \( r = 1 \). So each cell is influenced by the cell to its right and left.

We define our rule \( \theta: \{0, 1\}^3 \rightarrow \{0, 1\} \) by

\[ \theta(\sigma_{i-1}(t), \sigma_i, \sigma_{i+1}(t)) = \sigma_{i-1}(t) \oplus_2 \sigma_{i+1}(t), \]

where we are performing addition modulo 2. We can explicitly define the actions of this function.

\[
\begin{align*}
111 & \mapsto 0 \\
110 & \mapsto 1 \\
101 & \mapsto 0 \\
100 & \mapsto 1 \\
011 & \mapsto 1 \\
010 & \mapsto 0 \\
001 & \mapsto 1 \\
000 & \mapsto 0 
\end{align*}
\]
This idea of a rule that defines the future state of the cell based on the states of everything in its neighborhood, can be extended to two, three and higher dimensions with the only modification being that a neighborhood becomes a square, cube, or some other shape, however the idea is the same. Also additional more complicated rules and state structures can be imposed which increase the complexity of the system. The discrete states can be made real valued. Some degree of randomization can be imposed. In fact the amount of modifications possible are quite enormous, however all cellular automata are based off of the simple ideas presented here.
Although the one dimensional cellular automata we defined seem simplistic, even they can demonstrate some strikingly complicated behavior. In the one dimensional case, these are often plotted with time moving down, and with the automata beginning with only one or two cells in a nonzero state. Each pixel represents one cell and is white if the cell is in state 0, black if the cell is in state 1. The resulting picture is quite interesting and complicated often having a fractal structure. In two dimensions each frame represents one unit of time, and these can be played like a movie. These two dimensional cellular automata usually begin with only one or two active cells and yet develop into rich complicated organized structures, often with a very organic quality to them. I have some websites which we can look at to illustrate this.
4. **Implications of Conway’s Game of Life**

Conway’s game of life, as we mentioned earlier, is simply a particular rule for a two dimensional cellular automata. In this cellular automata we will think of a neighborhood of a cell, as those cells above, below, to the right and left, and diagonally offset from the center cell. This type of neighborhood is called a Moore Neighborhood, $\mathcal{N}_{Moore}$. The rule for this cellular automata is

\[
\begin{align*}
\text{Birth: } & \text{ a previously dead cell comes alive if exactly 3 neighbors are alive.} \\
\text{Death: } & \text{ isolated living cells with no more than one live neighbor die;} \\
& \text{ those with more than three neighbors die.} \\
\text{Survival: } & \text{ living cells with 2 or 3 neighbors survive.}
\end{align*}
\]

In our grid each cell has a value of 1 if it is alive, 0 if it is dead. We could rewrite the rule as

\[
\sigma_{i,j}(t+1) = \phi_{LIFE} \left[ \sigma(t) \in \mathcal{N}_{Moore} \right]
\]

\[
\begin{align*}
\sigma_{i,j}(t+1) &= \begin{cases} 
\sigma_{i,j}(t), & \text{if } \sum_{\sigma \in \mathcal{N}_{Moore}} \sigma(t) = 2, \\
1, & \text{if } \sum_{\sigma \in \mathcal{N}_{Moore}} \sigma(t) = 3, \\
0, & \text{otherwise},
\end{cases}
\end{align*}
\]

This simple system generates quite complicated and very organic behaviour which is similar to that of many micro-organisms. A variety of types of ”life-forms” are created which have been catalogued and named even. Even more interesting than the dynamics of this system, are its implications. It can be proven that the game of life is actually a universal computer. This means that given the correct set of initial conditions it can be used to perform any computation that any
standard digital computer is capable of. Using this for computation is impractical. To understand why this is significant we state loosely speaking a theorem proven by Turing in 1936, known as Halting’s theorem. This theorem states that there is no way of knowing that a general computer program will terminate. Since the game of life is a universal computer, this implies that it is impossible to in general state whether a particular starting configuration will eventually die out, and if so when. The only way to determine the behavior of a set of initial conditions is to compute it. Additionally since the Game of Life is a universal computer, isomorphic if you will to any digital computer, it can with a large enough lattice, and the correct initial conditions, display arbitrarily complicated behavior. The implications of this are quite profound, we will speak a bit more about this later.
5. Applications

So how are cellular automata useful. Cellular automata present a way of effectively modeling any complex system which is composed of a large number of interacting objects. These objects could be air particles in the atmosphere, cells, people, etc. By creating a set of rules that cause the automata to react the way we want them to, it is possible to use them to simulate almost anything. For example, in one article I found an SIR model that was simulated using cellular automata. In this case the lattice was a two dimensional plane (torus actually), and each vertex could either be empty or occupied by an individual that was either susceptible, infected, or removed. The rule for the model was that

1. Susceptibles become infected by contact, meaning a susceptible may become infected with a probability $p_i$, if, and only if it is in the neighborhood of an infected. The exact rule is that a susceptible with $z$ infected neighbors has a probability of $(1 - (1 - p_i)^z)$ of becoming infected.

2. Infected individuals transition to the removed category with probability $d_i$.

3. In addition to the previous rules which are applied at each time step, the model includes a site exchange rule. This means that at each time step, some number of individuals chosen at random can move to one vertex chosen at random. If the vertex chosen is occupied, he will not move. In a short range move the chosen vertex is any of the four neighboring vertices. In a long range move it is any vertex. The average number of times an individual
is chosen to move at each time step is the parameter $m$. This parameter $m$, represents the degree of mixing of the populations. This model can, and is of course later expanded to include separate birth rates, death rates, and other improvements.

Another simple model similar to this one could be used to model forest fires. In this model each vertex on the lattice represents a site which can be occupied by nothing, a live tree, or a burning tree. The rules are:

1. A burning tree becomes an empty site.
2. A green tree becomes a burning tree if at least one of its neighbors is a burning tree.
3. At an empty site, a tree grows with probability $p$.
4. A tree without a nearest burning neighbor becomes a burning tree during one time step with probability $f$ (lightning).

Although this model is simple, it could easily be expanded, and it takes the spatial structure of the forest into account.

In statistical physics and fluid dynamics, cellular automata have been used as a modeling tool. For example, in one simple model, the lattice is occupied by moving particles which are subject to two rules regarding collision and motion. The collision rule determines how particles that enter the same site change their trajectory. The motion phase moves particles to the site they are heading towards. This model effectively simulates dispersion and also wave propagation. For example. If a uniformly distributed system of particles has one region of high density, an expanding shockwave will be produced. More complicated models very much like this have been used to model for example fluid flow over
airplane wings, and combustion in jet engines.

In addition to the examples I have spoken about, the ideas behind cellular automata form the basis for most neural networks, and genetic algorithms. The models described here are a few out of many and they represent some of the simpler ones. Hopefully you can see how cellular automata provide an effective modeling technique by including spatial elements into discrete systems.
6. Limitations and Modifications

Cellular automata do have limitations, the main one being that they are nearly impossible to analyze. As we spoke about regarding the Game of Life many Cellular automata can be proven to be impossible to analyze, meaning the only way to predict their future states is to actually compute them. Approximation with any sort of partial differential equation for example is impossible. This is not to much of a limitation, especially since simulation with cellular automata is usually very effective and gives very clear and meaningful insight into the behavior of the system. In the case of fluid dynamics, Cellular automata reduces problems that are considered intractable, or impossible to solve in reasonable amount of time, into simple discrete simulations.

References