Cellular Automata

Fysikklubben

March 30, 2017

1. Definition, examples in 1 and 2D.
2. Properties

   Pretty Pictures

   Universal Computing

   Modelling of physical, chemical, biological and social systems
3. Example model: Game of Life
4. Example model: The neuron model
5. Example model: reaction-diffusion model
6. Comments on Fredkin's Finite Nature

1 Introduction

Cellular Automata are dynamical systems in d dimensions with discrete time steps. The system consists of "cells" which are located in some lattice (or, in general, cells sit on nodes in a graph). Each cell can be found in some number, k, of different states. Often, when we try to model some specific system or mimic some effect in the world, the states naturally correspond to "physical states", e.g. we may have two states: "Dead" and "Alive" or three: "Inhibited", "Excited", "Firing", depending on the application. The update rules are local, meaning that the state of cell i at time t depends on the state of neighbouring cells at time $t-1, t-2, ..., t-\ell$ for some finite ℓ, which is called the "order" of the model. A theory with a given neighbourhood (defined in some way), a number of states per cell and an update-rule is called a Cellular Automaton. The restriction of locality in time and space implies that there are a finite number of possible cellular automata, but of course we may vary initial conditions, change the number of states per cell, change the definition of "neighbourhood", and change the order of the theory. This leads to an incredibly rich menagerie of automata - but in fact even deceivingly simple 1-D cellular automata display complex behaviour.

Let's look at 1D automata to begin with.

1D Cellular Automata Let us assume that the neighbourhood for a cell is always symmetrical. Then there are only three parameters to worry about: the number of neighbours, i.e. the size of the neighbourhood, the order of the system and the number of states per cell. Let us consider a first order system with three neighbours (which includes the cell itself) with two states (0 and 1) per cell. In this case there are 2^3 possible configurations of any given neighbourhood, and for each of these we must determine how the central cell evolves. Since there are two choices for the central cell per neighbourhood state we find $2^{2^3} = 256$ possible update-rules/cellular automata. We can order the possible neighbourhood states in some arbitrary way, and then indicate a given automaton by a string of 1s and 0s. As an example, the automaton in figure 1 is given by the string 10010110. Interpreting this a number in base 2 yields an enumeration of all the possible 1D, first order, k = 3 Cellular Automata.

Figure 1

I will not go through all 256 of these. They show varied behaviour ranging from trivial (e.g. rule 0) to periodic, to random, to somewhere in between periodicity and randomness (e.g. chaotic or complex CA). Before we move on, let me mention rule 110 = 01101110 (base 2). This falls in the class of complex automata. Starting from one cell in state 1 (coloured black) we find the evolution shown in figure 2

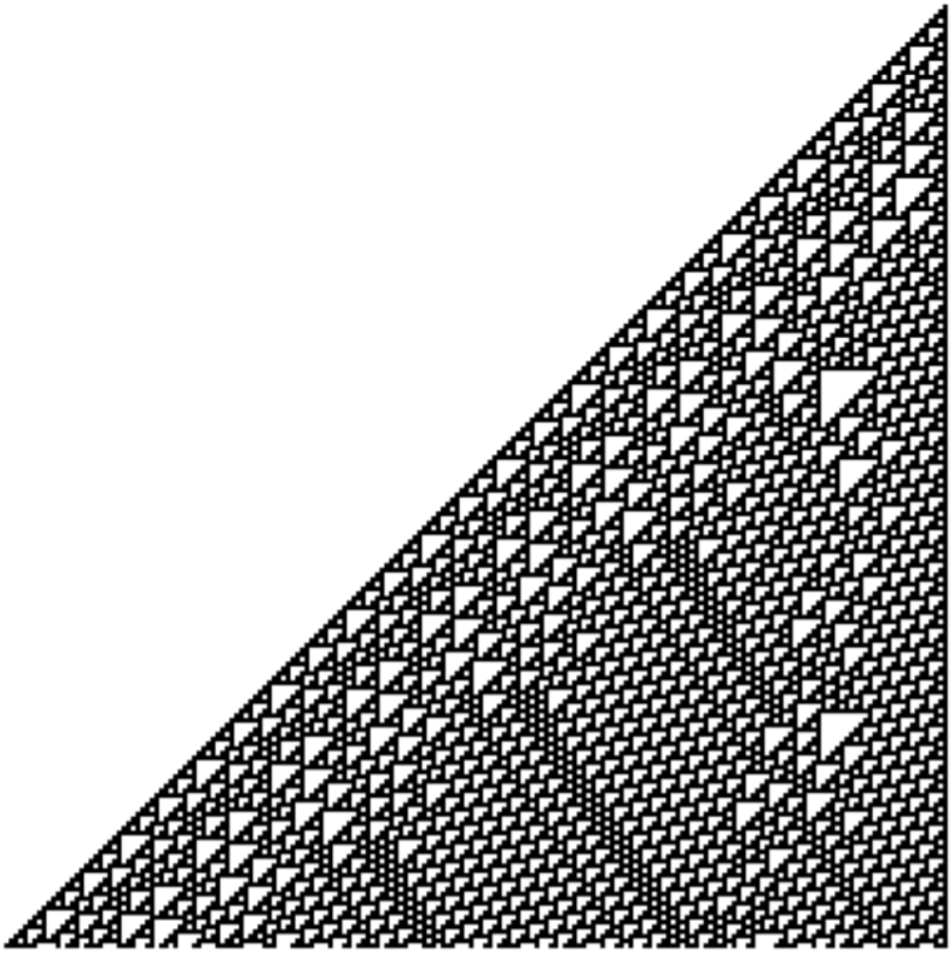

Figure 2

We see complex, seemingly random triangular patterns. Compare with this sea-shell:

Figure 3

It turns out that these deceptively simple systems can display exceedingly complex behaviour on large scales. This makes them interesting objects of study for a number of reasons.

1.1 Some properties of Cellular Automata

Because of the complexity they display, automata have found use in fields ranging from physics to economics. Clearly, they can be considered as simple toy models used to explore emergent phenonema in many-body systems. Researchers have had succes in describing characteristic features of galaxy formation, pigmentation in animals, segregation of cities and the dynamics of chemical reactions.

However, the complexity of cellular automata make them interesting in their own right. One can study such things as the statistical mechanics, dynamical properties and computational complexity of automata. In fact, certain automata have been found to be *universal computers* meaning they can perform any computation that is possible to perform - that is, these cellular automata are capable of as complex and varied behaviour as any computer. Rule 110 above is an example of such a universal computer. Stephen Wolfram postulates that the automata capable of universal computation are exactly the ones that are chaotic. An interesting notion.

The final reason (I can think of) why one might be interested in cellular automata is that they (sometimes) produce pretty pictures.

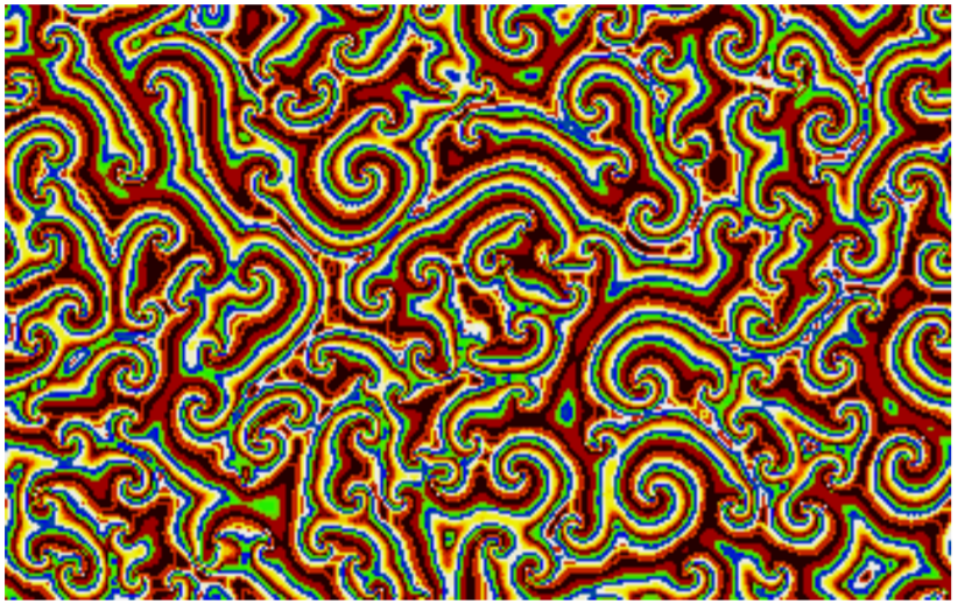

Figure 4: The hodgepodge machine.

2 Examples

2.1 Game of Life

Probably the most well-known cellular automaton is Conway's Game of Life. The cells can be in one of two states: "Dead" or "Alive". The cells live on a 2D grid and the state of a cell at the next time-step depends on the state of its 8 closest neighbours at the current time-step. The update rule is the following:

1. Any cell with 1 or 0 living neighbours dies
2. Any living cell with 2 or 3 neighbours lives
3. Any dead cell with exactly 3 neighbours becomes alive
4. Any cell with more than three neighbours dies

Despite the fact that the state-space and rules are quite simple this automaton shows really complex behaviour. There are indefinitely stable "Lifeforms", Lifeforms that reproduce in a certain sense and there are many complex, periodic Lifeforms known. In fact, the Game of Life is capable of universal computing. I will only give two examples and no code because it is very easy to find both online.

Figure 5: A bunch of period 2 Lifeforms.

Figure 6: A period 3 Lifeform.

2.2 The Neuron Model

Introduction

The next model we will talk about is due to Pytte. State of the art methods for describing neural systems involve huge numbers of differential equations, e.g. for 10,000 neurons one must solve 250,000 coupled differential equations. The reason for this complexity is that the mathematical description attempts to describe the underlying physical system very accurately: one accounts for currents in and connectivities of the axons connected to each neuron. In this kind of situation a cellular automaton is interesting for two reasons. Cellular automata are generally much faster to simulate than a differential equation description of the same system, so if one can find a CA that describes the system "well" one can perform (maybe preliminary) analyses much faster. Secondly, simplification is interesting in its own right, since it may allow one to single out the features of the system that produce characteristic phenomena. This can often provide a better understanding of the fundamental principles that are in play.

Pytte's CA attempts to model the hippocampi which are regions of the brain that are critical to the formation of memories and in spatial navigation. A lot of experimental data is apparently available on the hippocampi. I do not know how many of the features of Pytte's model are specific to the hippocampi and how many are generally applicable to neural systems. With that disclaimer, let us begin.

CA Hippocampus

There are three types of neurons in our model: excitatory (e), fast inhibitory (f), and slow inhibitory (s). Each neuron is a two-state system: "resting" and "firing". We specifiy the number of each type of neuron, and the number of other neurons each type is connected to. These connections are distributed randomly. The connections are directed, meaning the signal only travels in one direction along the connection. This means this CA lives on a general directed graph and we don't really have a notion of spatial dimension. We assume that each neural connection has a "bond strength" which is determined by the neuron from which the connection originates. Bond strengths are given by K_e, K_f, K_s.

Even though each neuron is either firing or not, because propagation along axons is not instantaneous, neurons fire for some period of time and because neurons may spontaneously fire we need to include extra automaton states to account for these effects.

Signals from excitatory neurons to inhibitory neurons are the fastest, so here we assume no latency; if (e) is firing at time t then connected inhibitory neurons feel this effect at the next time-step. For other connections we introduce latencies τ_e, τ_f, τ_s. τ_q is the latency for signals to propagate from a neuron of type q to an excitatory neuron. Inhibitory neurons are not affected by other inhibitory neurons so we do not need to consider such connections.

Additionally, each type of neuron fires for some amount of time. In brains the slow inhibitory neurons fire considerably longer than the other types. We indicate the length of the firing times by φ_q for a neuron of type q.

Only excitatory neurons fire spontaneously. The time before spontaneneous firing is σ; this number is random and different for each neuron.

Finally the excitatory neurons have a refractory period given by ρ.

The states of the excitatory neurons consists of several pieces of data: the current state (resting or firing), the time since last fired (for excitatory neurons) and the spontanteneous firing time. This last datum does not change under time-evolution but is generally different for each neuron. For inhibitory neurons we do not need the last two pieces of data.

$$(e) : \underbrace{0}_{resting}, \underbrace{1, 2, ..., \varphi_e}_{firing}; \quad \alpha; \quad \sigma, \tag{2.1}$$

$$(q) : \underbrace{0}_{resting}, \underbrace{1, 2, ..., \varphi_q}_{firing}. \tag{2.2}$$

Note also that because of the latency effects we have to store the states for a time $\max\{\tau_e, \tau_f, \tau_s\}$ $(= \tau_s)$. The update rules are as follows:

$$(e) : \alpha \to \alpha + 1, \qquad \text{if } S = 0, \tag{2.3}$$

$$\alpha = 1, \qquad \text{if } S \neq 0, \tag{2.4}$$

$$0 \to 1, \qquad \text{if } m_e(t)K_e - (m_f(t)K_f + m_s(t)K_s) > h(S(t)), \tag{2.5}$$

$$0 \to 1, \qquad \text{if } \alpha \geq \sigma, \tag{2.6}$$

$$S \to S + 1, \qquad \text{if } S > 0. \tag{2.7}$$

Here $m_q(t)$ counts the number of connected, firing neurons of type q at time $t - \tau_q$, and

$$h(S(t)) = \begin{cases} F(\rho - \alpha)/\rho & \alpha \leq \rho, \\ 0 & \alpha \geq \rho, \end{cases} \tag{2.8}$$

where F is the "threshold".

For inhibitory neurons the state update rules are

$$(q) : 0 \to 1, \qquad \text{if } b_e(t) > 0, \tag{2.9}$$

$$S \to S + 1, \qquad \text{if } S > 0. \tag{2.10}$$

Here $b_e(t)$ counts the number of connected, firing excitatory neurons at time $t - 1$.

To instantiate the model we need to specify the number of excitatory, fast inhibitory and slow inhibitory neurons, N_e, N_f, and N_s, as well as the number of excitatory, fast and slow inhibitory neurons any given neuron is connected to, z_e, z_f and z_s. The full set of system specifications are shown in the table.

Parameter	Symbol
Neuron type	$q = e, f, s$
Number of neurons	N_q
Number of connections	z_q
Bond strength	K_q
Signal latency	τ_q
Refractory period	ρ
Spontaneous firinge time	σ
Maximal threshold	F
Firing period	φ_q