MIT ES.S20 Lecture 7: Zero-Player Games

Lecture Outline

1. Zero-Player Games
2. Binary Cellular Automata
3. Generalized CA
4. CA as Physical Models
5. Other Physical Zero-Player Games
6. Replication Dynamics and Chaos
7. Games and Statistical Mechanics

This is going to be a very broad lecture, so I will treat a lot of subjects with very little depth – this is more to give you a feel for what ideas are out there and why this is interesting

I am going to move the replicator dynamics to another lecture because this seems to be a lot of material.

Zero-Player Games

1. What do I mean?

   There are a lot of agents, but all of them make fixed actions. Usually, there is a lot of regularity to the structural organization (network) of the agents. Also, this is the sort of domain that has a *huge* number of applications, so I will jump around a bit – the general ideas are pretty intuitive, and hopefully you pick them up.

   I'm not going to talk about Genetic Programming here: it's a fine optimization technique, but won't help us understand how these systems work.

2. Why do I care?

   What if our behavior is fixed anyway? This would make our analysis looks like a microscopic *physical model*, very much like those used in Statistical Mechanics. This is suggestive that models like CA would be useful in a scientific context.

Can we make generalizations of behavior? Often, we see simple models embedded in larger, more complicated structures like poker.

Can we make simplifications of behavior? Determine a mapping `State -> N` that makes complicated games or systems look simple.

They can be used to show complexity results? :)

Binary Cellular Automata

1. Introduction: 1D Cellular Automata

   Take an arbitrary string of bits and for each bit, perform some binary propagation rule: `b[n] &lt;- f(b[n-1], b[n], b[n+1])` for some function f. If we make a function table, we see that there are `2^8` possible rules. These are the 1D cellular automata described initially by Stephen Wolfram.

2. Game of Life

   Being mathematicians, we can, of course, generalize this to an arbitrary number of dimensions (we only care about neighbors); most of them are chaotic and not well understood; but the famous example is Conway's Game of Life, a 2D Cellular Automaton that exhibits a large variety of chaotic behavior.

   Generally, we can refer to games like GoL in terms of Birth and Death: a cell survives iff 2 or 3 of its neighbors are alive.

   Open Problem: How do we generate stable configurations?

3. Embedded Computation

   There are various types of stable elements in the game of life (and similar chaotic automata): Gliders, Spaceships, Guns, Still Lifes, etc. (Demonstration)

   What if our propagation rule lets us make a NAND? We can make a Turing Machine! (Demonstration) In this way, GoL is a universal computation substrate… and correspondence can be used to

There is something analagous to computability and complexity for these circuits: the Garden of Eden and timing limitations. Predicting whether a given CA will be stable is undecidable.

Which CA are reversible?

4. How do we compute this efficiently?

   Blocked Cellular Automata: cut up regions of the cellular automaton and turn it into a *local* operation and a *mixing* operation. This has the added subtlety that if we don't care about simulating the actual automaton properly, we can put different rules in (like seeds, etc. Demonstration).

   QuadTrees: construct a tree of regions in the CA and solve them out independently. This looks like a *sparse matrix representation*; it plays well with mostly-empty fields.

   HashLife: some patterns are more common than others, so what if we hashed the regions in a Cellular Automaton. We can also embed successors in the hash table – this is why programs like `Golly` are so fast.

   These problems are closely related to various *stencil*-type PDE solvers; they just don't make assumptions of linearity.

Generalized Cellular Automata

1. Nth-Order CA

   Cell depends on more than one previous state: the propagation rule looks like a feedback polynomial with translations in time and space.

2. Vornoi Diagrams for Cellular Automata

   We can make Cellular Automata on any tiling (regular or not) of our phase space. To treat this generally, we can look at Vornoi Diagrams (which are constructable in linear time in the number of points).

```
Vornoi Region of p_i = {points q | d(q, p_i) &lt; d(q, p_j) for all p_j}
```

   Other ideas have been suggested, such as using hexagonal grids, Penrose tilings, more dimensions, etc. These are mostly unhelpful.

3. Automata on Other Topologies

   We can put Cellular automata on whatever topology we want. They look the same, but the idea is rather neat.

   Can we tell what topology our CA is on?

4. SmoothLife

   What neighborhood are we computing on? SmoothLife constructs radial neighborhoods:

```
m = (1/M) * (Integral of the CA from r=0 to r_a)
n = (1/N) * (Integral of the CA from r=r_a to r_b)

f(x, t+1) = nonlinear(sigmoid(mix(m, n), threshold)) * f(x, t)
```

   In this way, we can embed the birth and death relations into the threshold and mix function

   How do we make this efficient under an FFT?

   Can we memoize this in any way? How about in one dimension?

5. Learning Cellular Automata

   Much like Neural Networks, we can embed a learning mechanism into each cell of a CA. We can get the

   What can we model with this? Rumor diffusion, commerce, mesh networks, …

6. Other CA

   Asynchronous CA: do they really need to all act in step? What does this do?

   Langtons Ant: Reformulate 2D CA as a turing machine

```
Square is black -> turn right, flip square, move forward
Square is white -> turn left, flip square, move forward
```

Patersons worms are a similar idea with different rules and tiling.

7. Algebraic Models

   Comonads: The state is a *zipper*, a general structure that can be shifted (left and right in the current model). Then, a *comonad* is a wrapper that

```
instance Comonad U where
    cojoin a = U (tail $ iterate left a) a (tail $ iterate right a)
    coreturn (U _ b _) = b
    x =&gt;&gt; f = fmap f (cojoin x)

rule (U (a:_) b (c:_)) = not (a && b && not c || (a==b))
```

   Totalisitic CA: the values of each cell are represented by an integer on a cyclic ring and the propagation rule looks like the group operation with its neighbors. These have similar Game Of Life-like properties and infinite mathematical potential.

Cellular Automata as Physical Models

1. The Limits of CA

   It is extremely tempting to overstep on making claims about Cellular Automata because they make a lot of sense of things and have a lot of mathematical elegance: Stephen Wolfram, for example, is a proponent of the idea that the universe itself is a Cellular Automaton of high dimension. What we *know* is that they make cool patterns and certain CA formulations limit to looking like physical phenomena. In speaking about physical models, we have to maintain a very high standard of proof.

2. Cellular Automata Fluids

   It is common with PDEs to discretize the angular and spatial domain anyway, so a Cellular Automaton is just a very coarse analogy. Take a hexagonal grid and give each *edge* a particle momentum. The transition rules are defined at each node and look like microscopic particle dynamics. This has been shown to correspond with Navier-Stokes.

   This sort of technique is used sometimes in games when we don't really care about granularity.

3. Quantum Cellular Automata

   Each cell is a Qubit and the evolution is by spin-coupling with a cell's neighbors. This has been suggested as an alternative to gate-based models which are very difficult in quantum systems because moving things around is difficult.

   This is somewhat related to the method used by D-Wave, which can solve quantum annealing (minimization) problems in one dimension.

4. WireWorld

   Construct a CA so that gliders, gates, and wires are very simple, and we can simulate digital circuits on discrete time steps.

   Can we simulate Analog?

Other Physical Zero-Player Games

1. Bacteria Simulation (BacSim)

   There are resources distributed in space and individual bacterial agents with some reasoning capacity that can be encoded (in the style of Genetic Programming). This sort of system is used to simulate the growth of bacterial colonies. *This is a classic example of a Multiagent System*

2. Neural Networks (CoDi)

   Asynchronous Learning Cellular Automata on Vornoi networks? That's pretty similar to a Neural Net. Actual simulation is hard, but the primary notion is that of statistics on *firing rates*, and the firing of a neuron is contingent on a certain density of firing at its dendrites. (see *Theoretical Neuroscience* for a biophysical treatment)

   CoDi is a cellular automaton that does this at the very high level of WireWorld, with specific colors for axons, dendrites, and signals traveling through them. It is also an example of a Cellular Automaton that was constructed intricately enough as to enable learning.

3. Self-Organizing Maps / Neural Gases

What if the learning in a CA looks like movement? We can have a network converge to minima or to a topological map via some learning algorithm. The distance between nodes can look like the gradient on the state space (here, D):

```
Wv(s + 1) = Wv(s) + Theta(u, v, s) Alpha(s) (D(t) - Wv(s))
```

Where Wv is the length of the edge, D(t) is the data point, Alpha is a learning rate, v is our node index, and u is the closest node. Theta is a restraint from moving all of the nodes around (so that we get an actual map instead of a mean).

Replication Dynamics and Chaos

We will have most of a lecture on differential equations and replicator dynamics later; but I'd like to introduce it now.

1. Evolutionary Dynamics

   `dx_i = x_i * (f(x_i) - phi)` where `phi = sum_j{ x_j * f(x_j) }` where x is the proportion of agents of a given category (this can be of arbitrary dimension) and f is the fitness function. If we assume the fitness function is a matrix, this is a very simple functional form.

   What happens when there are optimal ratios of different species? This information can all be embedded in phi (it looks like a payoff matrix).