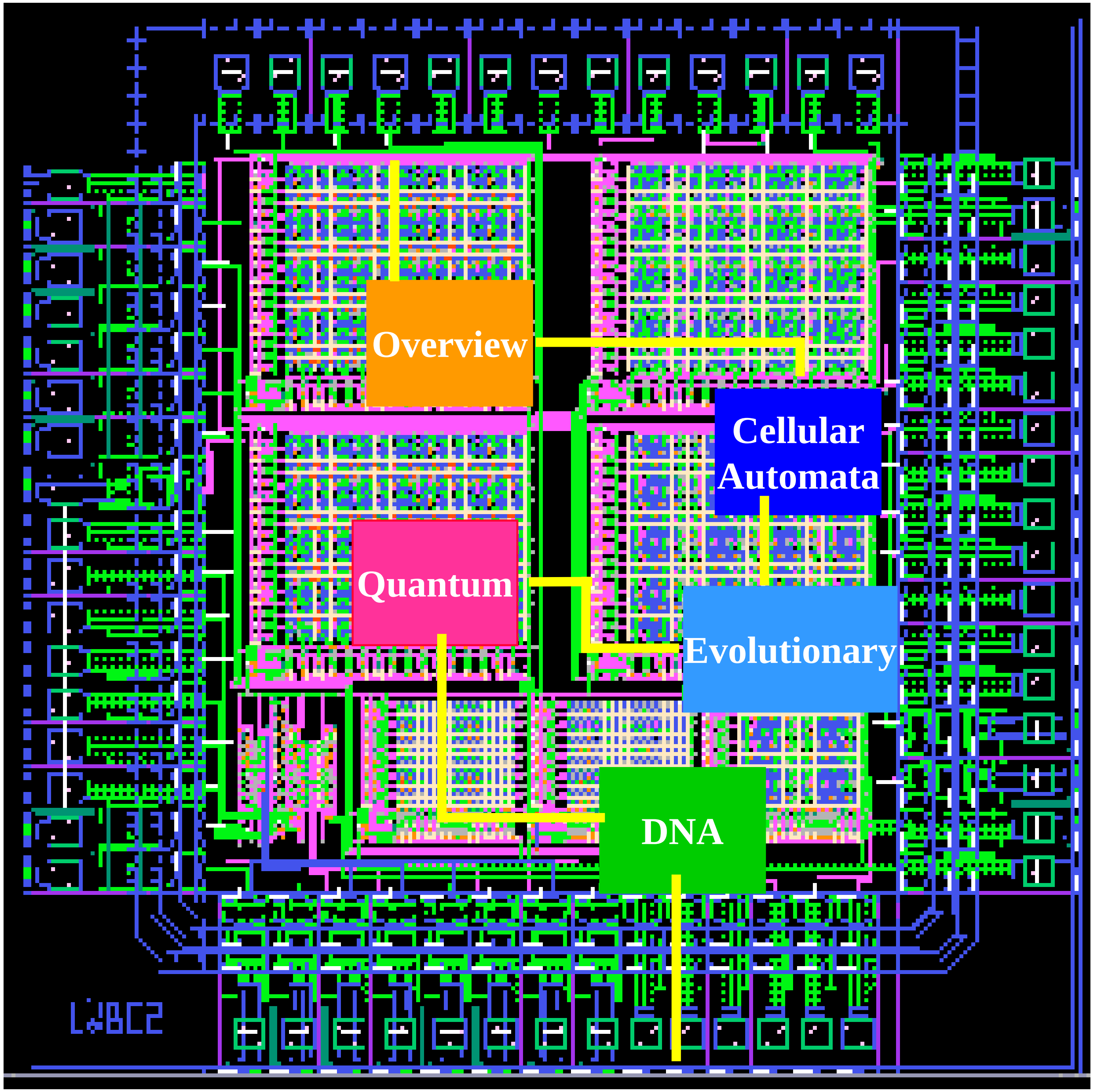
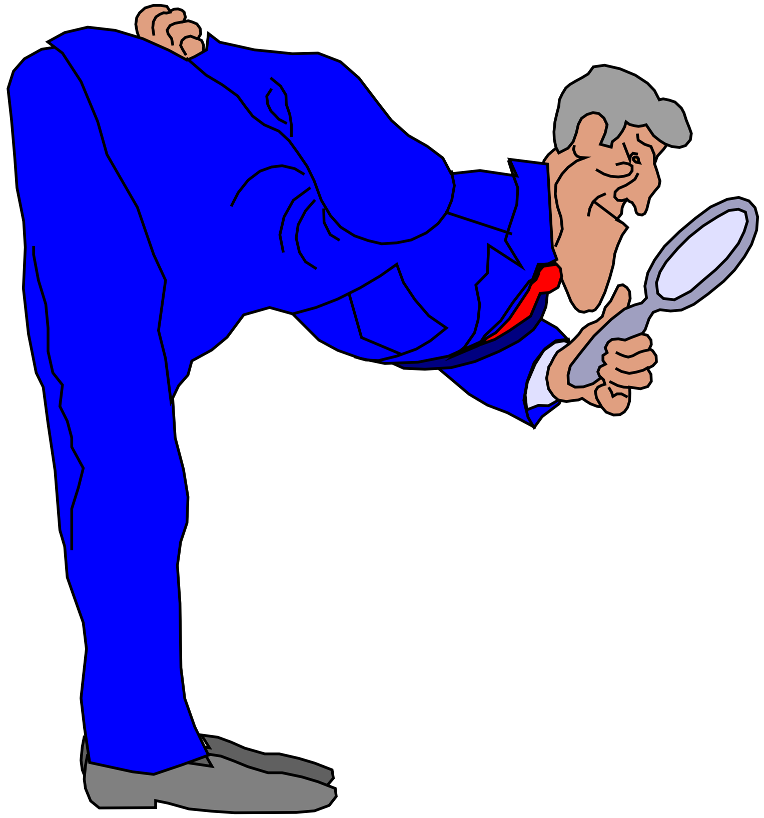
Cellular Automata

Based mostly on Dr. Richard Spillman class on Alternative Computing in

Summer 2000


Overview
Cellular Automata
Quantum
Evolutionary
DNA
LWBC2

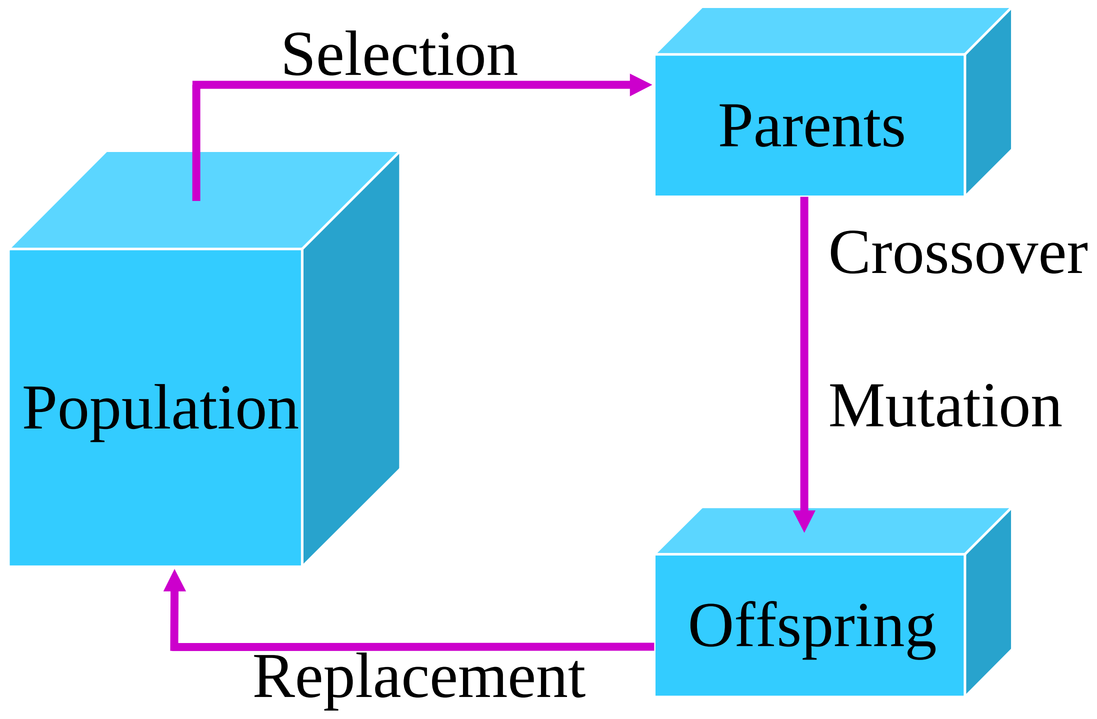
Review



- Binary Logic
- Multiple-Valued Logic
- Reversible Logic
- Automata - Finite State Machines
- Cellular Automata
- Introduction to Hardware Evolution
- Reconfigurable Computing
- Hardware Evolution Details

Idea – Genetic Algorithms

- The Evolutionary Process:



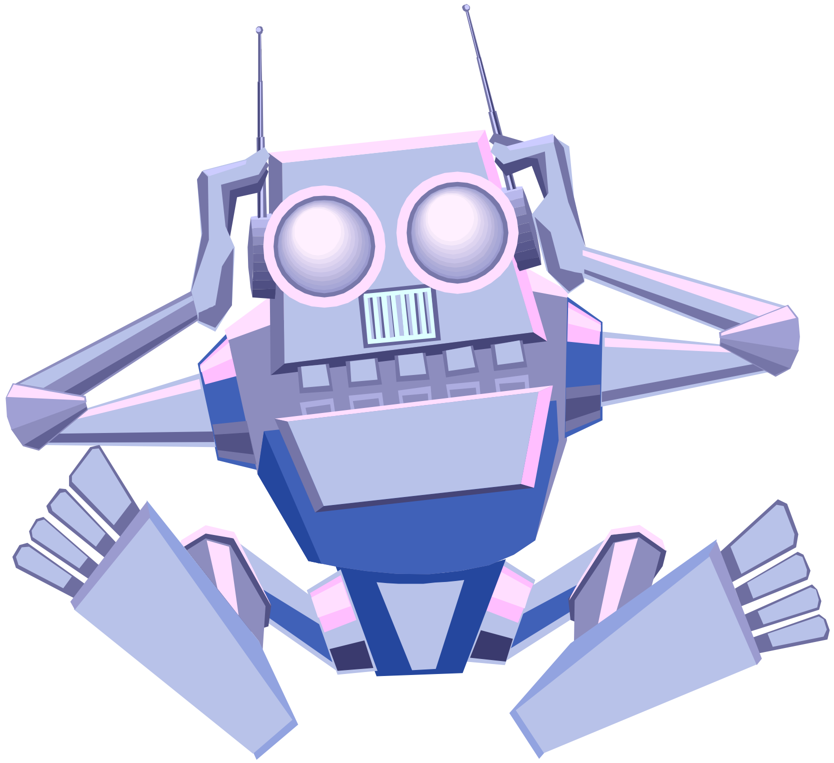
Review – Cellular Automata

- **A Cellular Automata is consists of:**
  - **An *n*-dimensional array of simple cells**
  - **Each cell may in any one of *k*-states**
  - **At each tick of the clock a cell will change its state based on the states of the cells in a local neighborhood**

- **The three main components of a Cellular Automata are:**
  - **The array dimension**
  - **The neighborhood structure**
  - **The transition rule**

Synchronous!!

OUTLINE



- Some Properties of CA
- Cellular Automata Rules
- Genetic Algorithms and Cellular Automata
  - their relations and possible extensions

Introduction to Cellular Automata

- Moshe Sipper defines three principles of cellular computing
  - **Simplicity**: the basic processing element, the cell, is simple
  - **Vast Parallelism**: Cellular computing can involve 10^5 or more cells
  - **Locality**: all interactions take place on a purely local basis, a cell can communicate with a few other cells

Cellular Computing = simplicity + vast parallelism + locality

Advantages of CAs

- Cellular Automata offer many advantages over standard computing architecture including:
  - **Implementation**: CAs require very few wires
  - **Scalability**: It is easy to upgrade a CA by adding additional cells
  - **Robustness**: CAs continue to perform even when a cell is faulty because the local connectivity property helps to contain the error

Example of hypercube and Intel parallel processors

Applications of CAs

- CAs have been (or could be) used to solve a wide range of computing problems including:
  - **Image Processing**: Each cell correspond to an image pixel and the transition rule describe the nature of the processing task
  - **Random Number Generation**: CAs can generate large sequences of random numbers
  - **NP-Complete Problems**: CAs can address some of the more difficult problems in computer science

Possible Homeworks on CAs

- **Image Processing**:
  - 1. Design Cellular Architectures for various Edge Detection algorithms.
  - 2. Design a Cellular Architecture for thinning.
  - 3. Design a CA for finding a contour based on exoring.
- **Random Number Generation**:
  - 1. Design a controlled random number generator with smaller aliasing rate than the architecture discussed in class (a linear counter based on shift register and EXOR gates).
- **NP-Complete Problems**:
  - 1. Design a CA for arbitrary NP-complete problem, such as graph coloring or satisfiability.

Cellular Automata Rules

- The transition rules define the operation of a cellular automata
  - For a **1-d binary CA** with a 3-neighborhood (the right and left cells) there are 256 possible rules
  - These rules are divided into "legal" and "illegal" classes
    - Legal rules must allow an initial state of all 0's to remain at all 0's
    - Legal rules must be reflection symmetric
      - 100 and 001 have identical values
      - 110 and 011 have identical values
    - There are only 32 legal rules

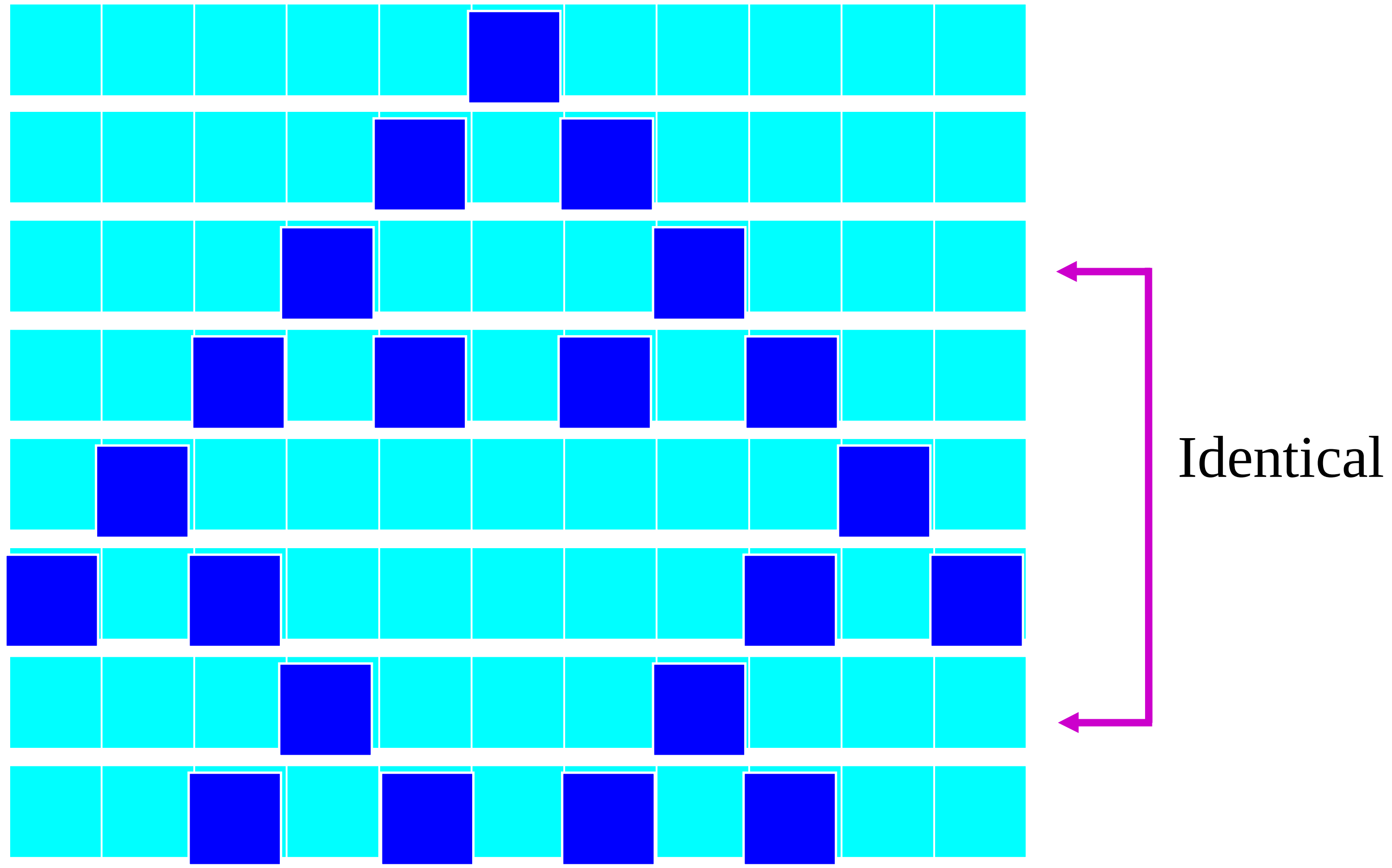
One-Dimensional Rules. Wolfram Works

- The performance of rules are studied in two ways:
  - **1.** By their impact on a CA with an initial state of a single 1 cell
  - **2.** By their impact on a CA with a random initial state
- Wolfram has determined the behavior of all 32 legal rules,
  - starting with an initial state of a single 1 cell

Example

- Consider rule 18: 0 1 0 0 1 0 0 0

a b c	Y
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	0



Rule Comparison

- One method of comparing different CA rules involves looking at the behavior of the rule on two similar initial conditions
  - Does the rule produce similar patterns or does it produce completely different patterns?

- A convenient measure of distance between binary cellular automata configurations is the **"Hamming Distance"**
  - It's the number bits which differ between two binary strings

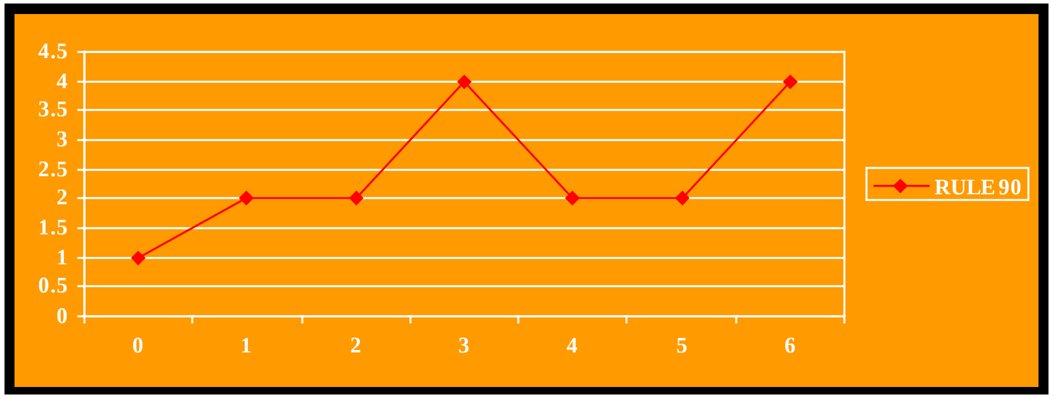
```
1 0 0 1 1 0 1 0
0 1 1 1 1 0 0 0
x x x         x     = 4
```

Example One - the same rule on different initial data

- Consider Rule 90:

000	001	010	011	100	101	110	111
0	1	0	1	1	0	1	0

		Hamming Distanc
0 1 1 0 0 0 1 1 0 0 0	0 1 1 0 1 0 1 1 0 0 0	1
1 1 1 1 0 1 1 1 1 0 0	1 1 1 0 0 0 1 1 1 0 0	2
1 0 0 1 0 1 0 0 1 1 1	1 0 1 1 0 1 1 0 1 1 1	2
1 1 1 0 0 0 1 1 1 0 0	1 0 1 1 0 1 1 0 1 0 0	4
1 0 1 1 0 1 1 0 1 1 1	0 0 1 1 0 1 1 0 0 1 1	2
1 0 1 1 0 1 1 0 1 0 0	1 1 1 1 0 1 1 1 1 1 1	2
0 0 1 1 0 1 1 0 0 1 1	0 0 0 1 0 1 0 0 0 0 0	4

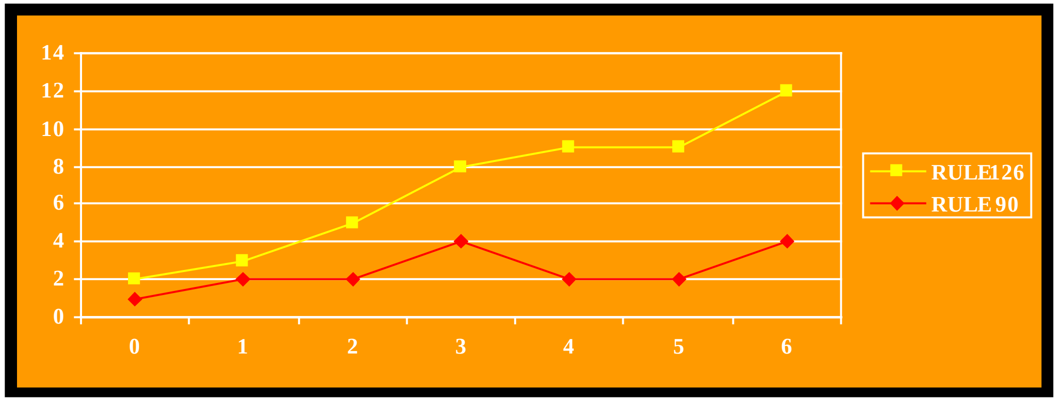


Example Two

- Now try Rule 126:

000	001	010	011	100	101	110	111
0	1	1	1	1	1	1	0

		Hamming Distance
0 1 1 0 0 0 1 1 0 0 0	0 1 1 0 1 0 1 1 0 0 0	1
1 1 1 1 0 1 1 1 1 0 0	1 1 1 1 1 1 1 1 1 0 0	1
1 0 0 1 1 1 0 0 1 1 1	1 0 0 0 0 0 0 0 1 1 1	3
1 1 1 1 0 1 1 1 1 0 0	1 1 0 0 0 0 0 1 1 0 0	4
1 0 0 1 1 1 0 0 1 1 1	1 1 1 0 0 0 1 1 1 1 1	7
1 1 1 1 0 1 1 1 1 0 0	0 0 1 1 0 1 1 0 0 0 0	4
1 0 0 1 1 1 0 0 1 1 1	0 1 1 1 1 1 1 1 0 0 0	8



Complex rules (like 126) are more sensitive to the initial condition than simple rules (like 90)

Possible Homeworks

•1. **Perform Wolfram-like analysis of rules for two-dimensional CAs.**

•2. **Find Complex rules for 2D CA (like an equivalent of rule 126 in 1D CAs) and show that they are are more sensitive to the initial condition than simple rules (like rule 90 in 1D CAs)**

• 3. **Can we link the “sensitivity” to the *“interestingness”* defined by us in the project?**

Two Dimensional CAs

- Two-dimensional cellular automata seem to model many physical processes such as:
  - Crystal growth
  - Diffusion systems
  - Turbulent flow patterns
- Like 1-d systems, 2-d CAs have transition rules:
  - A von Neumann neighborhood rule looks like:

$$q_{i,j}(t+1) = f[q_{i,j}(t), q_{i+1,j}(t), q_{i-1,j}(t), q_{i,j-1}(t), q_{i,j+1}(t)]$$

Observe that sometimes the cell itself is included to the neighborhood in definitions and sometimes it is not.

2-d Rules

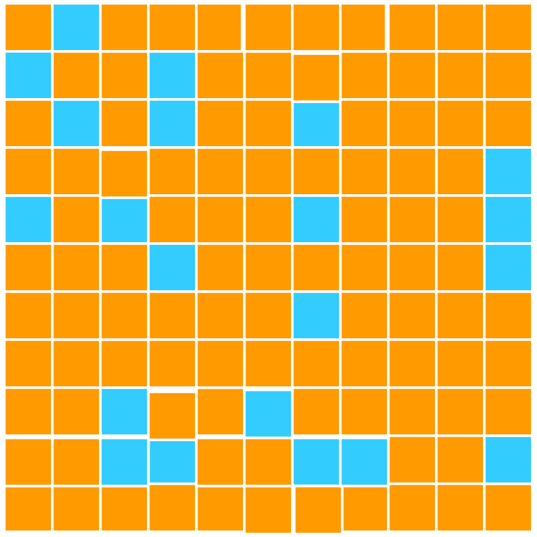
- The number of possible 2-d rules is quite large making a study of each individual rule impossible.

- For example:
  - There are 2^{32} or about 4×10^{9} von Neumann rules
  - There are 2^{512} or about 10^{154} Moore rules

- However, some observations can be made
  - Some rules produce regular patterns
  - Some rules produce structures with dendritic boundaries
  - Some rules produce slow growing patterns which tend to be circular

Example

- 2-d binary von Neumann Cellular Automata with a mod 2 sum rule
  - Start the array with a seed (a few 1’s)



We are interested in the sequence of patterns produced by this rule as compared to other rules

Some Links of GA and CA

- A genetic algorithm could be used to find a rule which produces targeted behavior in a CA

- A useful test problem for emergent computation is the **density-classification task**
  - if the initial configuration (IC) of cell states has a majority of 1's then it should go to the fixed-point configuration of all 1's in M steps, otherwise it should produce the fixed-point configuration of all 0's in M steps
  - this is called the $p_c = 1/2$ task
    - if p_0 is the density of 1's in the IC then the all 1's configuration should occur when $p_0 > p_c$
  - Is there a CA rule that will produce this behavior?
  - With only local information this is hard

density-classification task (cont)

Example GA

- On-going work at Santa Fe Institute by Mitchell, et. al.

- Initial attempts
  - GOAL: Search for a **r=3 CA rule** to perform the $p_c = 1/2$ task
  - Use a CA with N=149 cells

density-classification task

Representation

- The GA rule structure consisted of the output bits of the rule table in binary order
  - The **r=3 neighborhood** of a 1-d CA consists of the 3 cells on each side of the target cell
  - bit 0 is the rule for the 0 0 0 0 0 0 0 neighborhood, bit 1 is the output rule for the 0 0 0 0 0 0 1 neighborhood, etc
  - chromosome size is 128 bits

density-classification task

Fitness

- Each rule in the population was run on a sample of 100 ICs (initial conditions) randomly chosen
  - each CA was run until it arrives at a fixed point or for a maximum of M = 2N steps
  - fitness was the *fraction of ICs which produced the correct final behavior*
  - A different sample was selected at each generation
  - the random sample was biased to insure that the density of 1’s varied from 0 to 1

density-classification task

Parent Selection

- The population size was set at 100
- The CAs were ranked in order of fitness
- The top 20% (elite rules) were passed to the next generation without modification
- The remaining 80% of the new population were produced using crossover between parents randomly selection from the elite rules

density-classification task

Crossover/Mutation

- Single point crossover was used
- the offspring from each crossover were each mutated at exactly two randomly chosen positions

density-classification task

Results

- Each GA ran for a maximum of 100 generations

- While no general rule was discovered, the GA did find rules that worked on about 75% of the ICs

Similar Other CA-related Homeworks

- You could look at the behavior of several rules in a 1-d or 2-d system
  - Find patterns
  - Compare the rules on the basis of there impact on small changes in the initial conditions
- Build a GA to generate a CA
- Look into the connection between artificial life and cellular automata

Possible Homeworks

- **1.** Create and specify CA with 3D rules.
  How to implement them in know FPGAs?
- **2.** Discuss such issues in von Neumann rules versus Moore rules
- **3.** Build cells of CAs using Quantum Dot technology

Off to work!
Remember Final Exam
Remember Homework-Project on GA and CA