Computing Beyond Silicon Summer School

Physics becomes the computer

Norm Margolus
Physics becomes the computer

- **Emulating Physics**
  - Finite-state, locality, invertibility, and conservation laws

- **Physical Worlds**
  - Incorporating comp-universality at small and large scales

- **Spatial Computers**
  - Architectures and algorithms for large-scale spatial computations

- **Nature as Computer**
  - Physical concepts enter CS and computer concepts enter Physics

CBSSS 6/24/02
Emulating Physics
Why emulate physics?

• Comp must adapt to microscopic physics
• Comp models may help us understand nature
• Rich dynamics

• Start with locality: Cellular Automata
Conway’s “Game of Life”

In each 3x3 neighborhood, count the ones, not including the center:

- **If total = 2:** center unchanged
- **If total = 3:** center becomes 1
- **Else:** center becomes 0

256x256 region of a larger grid. Glider gun inserted near middle.
Conway’s “Game of Life”

- Captures physical locality and finite-state

*But,*

- Not reversible (doesn’t map well onto microscopic physics)
- No conservation laws (nothing like momentum or energy)
- No interesting large-scale behavior

256x256 region of a larger grid. About 1500 steps later.
Reversibility & other conservations

• Reversibility is conservation of information
• Why does exact conservation seem hard?

The same information is visible at multiple positions

For rev, one $n^{th}$ of the neighbor info must be left at the center
Adding conservations

- With traditional CA’s, conservations are a non-local property of the dynamics.
- Simplest solution: redefine CA’s so that conservation is a manifestly local property.
- $CA \equiv \text{regular computation in space & time}$
  - Regular in space: repeated structure
  - Regular in time: repeated sequence of steps
Use 2x2 blockings. Use solid blocks on even time steps, use dotted blocks on odd steps.

Even steps:  rotate cw or ccw

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>b</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>

Odd steps:  rotate cw or ccw

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>b</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>

We “randomly” choose to rotate blocks 90-degrees cw or ccw (we actually use a fixed sequence of choices for each spot).
Diffusion rule

We “randomly” choose to rotate blocks 90-degrees cw or ccw (we actually use a fixed sequence of choices for each spot).
Diffusion rule

Even steps: \[ \text{rotate } \text{cw or } \text{ccw} \]

\[
\begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cc}
  c & a \\
  d & b \\
\end{array}
\quad \text{or} \quad
\begin{array}{cc}
  b & d \\
  a & c \\
\end{array}
\]

Odd steps: \[ \text{rotate } \text{cw or } \text{ccw} \]

\[
\begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cc}
  c & a \\
  d & b \\
\end{array}
\quad \text{or} \quad
\begin{array}{cc}
  b & d \\
  a & c \\
\end{array}
\]

*We “randomly” choose to rotate blocks 90-degrees cw or ccw (we actually use a fixed sequence of choices for each spot).*
Use 2x2 blockings. Use solid blocks on even time steps, use dotted blocks on odd steps.

Even steps: rotate cw

\[
\begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cc}
  c & a \\
  d & b \\
\end{array}
\]

Odd steps: rotate ccw

\[
\begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cc}
  b & d \\
  a & c \\
\end{array}
\]

Except: 2 ones on diag, nc

\[
\begin{array}{cc}
  \text{solid} & \text{dotted} \\
  \text{solid} & \text{dotted} \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cc}
  \text{solid} & \text{dotted} \\
  \text{solid} & \text{dotted} \\
\end{array}
\]
TM Gas rule

Even steps: update solid blocks.

Even steps: rotate cw

```
  a b  c a
  c d  d b
```

Odd steps: rotate ccw

```
  a b  b d
  c d  a c
```

Except: 2 ones on diag, nc

```
   □ □ □ □
   □ □ □ □
   □ □ □ □
   □ □ □ □
```
Even steps: \textit{rotate cw}
\[
\begin{array}{cc}
a & b \\ c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
c & a \\ d & b \\
\end{array}
\]

Odd steps: \textit{rotate ccw}
\[
\begin{array}{cc}
a & b \\ c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
b & d \\ a & c \\
\end{array}
\]

Except: \textit{2 ones on diag, nc}
\[
\begin{array}{cc}
\text{Odd step: update dotted blocks} \\
\end{array}
\]
TM Gas rule

Even steps: rotate cw

Odd steps: rotate ccw

Except: 2 ones on diag, nc

Even step: update solid blocks
Even steps: rotate \textit{cw}

\begin{array}{cc}
    a & b \\
    c & d \\
\end{array} \quad \rightarrow \quad \begin{array}{cc}
    c & a \\
    d & b \\
\end{array}

Odd steps: rotate \textit{ccw}

\begin{array}{cc}
    a & b \\
    c & d \\
\end{array} \quad \rightarrow \quad \begin{array}{cc}
    b & d \\
    a & c \\
\end{array}

Except: 2 ones on diag, \textit{nc}

Odd step: update \textit{dotted} blocks
TM Gas rule

Even steps: rotate cw

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
c & a \\
d & b \\
\end{array}
\]

Odd steps: rotate ccw

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
b & d \\
a & c \\
\end{array}
\]

Except: 2 ones on diag, nc

Even step: update solid blocks
TM Gas rule

Even steps: rotate cw

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th></th>
<th>c</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
<td>-------</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

Odd steps: rotate ccw

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th></th>
<th>b</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
<td>-------</td>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>

Except: 2 ones on diag, nc

Odd step: update dotted blocks
TM Gas rule

Even steps: \(\text{rotate } \text{cw}\)

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
c & a \\
d & b \\
\end{array}
\]

Odd steps: \(\text{rotate } \text{ccw}\)

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
b & d \\
a & c \\
\end{array}
\]

Except: 2 ones on diag, nc

\[
\begin{array}{cc}
& \\
& \\
\end{array}
\rightarrow
\begin{array}{cc}
& \\
& \\
\end{array}
\]

CBSSS 6/24/02
TM Gas rule

Even steps:  rotate cw

\[
\begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
  c & a \\
  d & b \\
\end{array}
\]

Odd steps:  rotate ccw

\[
\begin{array}{cc}
  a & b \\
  c & d \\
\end{array}
\rightarrow
\begin{array}{cc}
  b & d \\
  a & c \\
\end{array}
\]

Except: 2 ones on diag, nc

\[
\begin{array}{cc}
  \text{ } & \text{ } \\
  \text{ } & \text{ } \\
\end{array}
\rightarrow
\begin{array}{cc}
  \text{ } & \text{ } \\
  \text{ } & \text{ } \\
\end{array}
\]
TM Gas rule
TM Gas rule
Lattice gas refraction

- **Half the time:** HPP gas rule everywhere:
  - Even & odd: *swap along diags*
  - Except: *two ones on diag flip*

- **Half the time:** HPP gas rule outside of blue region, *ID rule* inside (no change).
Lattice gas hydrodynamics

Six direction LGA flow past a half-cylinder, with vortex shedding. System is 2Kx1K.
Dynamical Ising rule

We divide the space into two sublattices, updating the gold on even steps, silver on odd.

A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.
Dynamical Ising rule

Even steps: update gold sublattice

Odd steps: update silver sublattice

A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.
Dynamical Ising rule

Even steps: update gold sublattice
Odd steps: update silver sublattice

A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.
Bennett’s 1D rule

Gold/silver 1D lattice

Even steps: update gold sublattice

Odd steps: update silver sublattice

At each site in a 1D space, we put 2 bits of state. We’ll call one the “gold” bit and one the “silver” bit. We update the gold bits on even steps, and the silver on odd steps.

A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.
Bennett’s 1D rule

Even steps: update gold sublattice

Odd steps: update silver sublattice

A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.
If the heat bath is initially much cooler than the spin system, then domains grow as the spins cool.
2D “Same” rule

Gold/silver checkerboard
We divide the space into two sublattices, updating the gold on even steps, silver on odd.

Even steps: update gold sublattice
A spin is flipped if all 4 of its neighbors are the same. Otherwise it is left unchanged.

Odd steps: update silver sublattice
2D “Same” rule

Even steps: update gold sublattice

Odd steps: update silver sublattice

A spin is flipped if all 4 of its neighbors are the same. Otherwise it is left unchanged.
2D “Same” rule

Even steps: update gold sublattice

Odd steps: update silver sublattice

A spin is flipped if all 4 of its neighbors are the same. Otherwise it is left unchanged.
3D “Same” rule
Reversible aggregation rule

Gold/silver checkerboard

Even steps: update gold sublattice

Odd steps: update silver sublattice

We update the gold sublattice, then let gas and heat diffuse, then update silver and diffuse.

When a gas particle diffuses next to exactly one crystal particle, it crystallizes and emits a heat particle. The reverse also happens.

for more info, see cond-mat/9810258
Reversible aggregation rule

Even steps: update gold sublattice

Odd steps: update silver sublattice

When a gas particle diffuses next to exactly one crystal particle, it crystallizes and emits a heat particle. The reverse also happens.

for more info, see cond-mat/9810258
Adding forces irreversibly

Particles six sites apart along the lattice attract each other.

3D momentum conserving crystallization.
Adding forces irreversibly

Crystallization using irreversible forces  (Jeff Yepez, AFOSR)
Conservations summary

To make conservations manifest, we employ a sequence of steps, in each of which either

1. the data are rearranged without any interaction, or
2. the data are partitioned into disjoint groups of bits that change as a unit. Data that affect more than one such group don’t change.

Conservations allow computations to map efficiently onto microscopic physics, and also allow them to have interesting macroscopic behavior.
Physics becomes the computer

Emulating Physics
» Finite-state, locality, invertibility, and conservation laws

Physical Worlds
» Incorporating comp-universality at small and large scales

Spatial Computers
» Architectures and algorithms for large-scale spatial computations

Nature as Computer
» Physical concepts enter CS and computer concepts enter Physics