Topic 5: Interdisciplinary Problems and Python Scripting

Lecture 5-2: More Cellular Automata

Friday, April 2, 2010

Contents

1 One dimensional sandpile automaton

This is a simple one-dimensional cellular automaton. It's behavior is not very interesting, but it is relatively simple to understand, and to visualize using OpenGL graphics.

Imagine a one-dimensional array of L columns of sand grains with non-negative integer heights

$$h_i \ , \qquad i = 0, 1, \ldots, L{-}1 \ .$$

At each column position we define the local *slope* of the pile

$$s_i \equiv h_i - h_{i+1} \ .$$

We also define a *critical slope* $s_{\rm c}$: the pile is *unstable* if $s_i > s_{\rm c}$ for any column i; or equivalently, the pile is *stable* only if $s_i \leq s_{\rm c}$ for all columns.

The local rule for updating the sand pile is as follows:

- If the pile is unstable, then for each column such that $s_i > s_{\rm c}$, let $h_i \longrightarrow h_i{-}1$ and $h_{i+1} \longrightarrow h_{i+1}{+}1$, that is, move the top sand grain on column i to column $i{+}1$. This update is carried out *synchronously* on all columns.
- If the pile is stable, then add a sand grain to a randomly chosen column.

To make the model well defined, we need to say what happens at the two boundaries. Note that an unstable column in this model can only topple to the right (which is not very realistic). Thus the left boundary at

$i = 0$ can be considered to be a rigid retaining wall. We can, for example apply open boundary conditions at the right boundary $i = L$ by fixing $s_L = 0$: thus if $h_{L-1} > s_c$ the unstable grains are removed from the system, and can be considered to fall off the edge of the table on which the sandpile is being built.

The C++ OpenGL code sand1.cpp animates this sandpile model.

2 Two-dimensional Cellular Automaton Models

John von Neumann is usually credited inventing cellular automaton models. He was interested in the question of whether a digital machine based on a deterministic algorithm might be capable of reproducing itself. In collaboration with Stanislaw Ulam, he succeeded in finding such an automaton model based on a two-dimensional lattice of cells, see `http://en.wikipedia.org/wiki/Von_Neumann_cellular_automaton`. In this model, each cell has 29 states, and there is a complex set of local rules for synchronous update of the cells. The local neighborhood of a cell was taken to include the four nearest neighbors: such a neighborhood is called a *von Neumann neighborhood*.

3 Conway's Game of Life

The basic features of cellular automata are nicely illustrated by the *Game of Life* automaton, which was invented by the mathematician John Conway in 1970. This can be thought of as a model of an ecological

system of creatures living on a 2-D substrate.

- Space is discrete: the cells are taken to form a 2-D square grid.
- Each cell can be in one of only two states:
  - dead (0), or
  - alive (1).

This makes it a *Boolean cellular automaton*.

- The local neighborhood of a cell is taken to be the *Moore neighborhood*:

```
X  X  X
X  0  X
X  X  X
```

  which consists of the cell itself and 8 surrounding neighbors. Von Neumann's original automaton used the *von Neumann neighborhood*

```
   X
X  0  X
   X
```

- The cell value is updated by counting the number of live neighbors, which can take values $0, 1, \ldots, 8$:

```
t              t + 1
-    ------------------------
0 -> 0  0  0  1  0  0  0  0  0
     0  1  2  3  4  5  6  7  8  <= No. of live neighbors
1 -> 0  0  1  1  0  0  0  0  0
```

This automaton has many fascinating properties including a variety of *life forms* and the fact that it is capable of universal computation!

The Game of Life Java Applet implements these rules.

References

[W-CA] Wikipedia: Cellular automaton, http://en.wikipedia.org/wiki/Cellular_automaton.