McMaster University
Department of Computing and Software
Dr. W. Kahl

FP 2006
Assignment 2

CAS 743 — Functional Programming

26 January 2006

Cellular Automata

1. One-Dimensional Cellular Automata — 70% of Midterm 2003

A one-dimensional cellular automaton operates in a linear **space** of **cells**. For the purposes of this problem, only finite spaces are considered.

In every **generation**, every cell is assigned one of a set of **states**. Neighbouring the first and last cells of the space, one should assume "virtual cells" that are always in a **default state**.

For calculating from an (old) generation the **next generation**, there is a **rule** associated with the finite automaton. This rule prescribes what the state of every element is in the next generation, depending on its own state in the old generation, and on the states of its left and right neighbours in the old generation.

A **run** of such a linear cellular automaton is a sequence of successive state assignments of the space.

For example, the **Sierpinski cellular automaton** has two states, "empty" (default) and "full", and the rule states that a cell is full in the next generation if out of the three considered cells (i.e., itself and its left and right neighbours), exactly one or two are full in the old generation.

If every **line** represents **one generation**, with full cells represented by '@' characters and empty cells by space characters, then an example run of the Sierpinski cellular automaton, starting with exactly the middle cell full, can be represented by the **sequence of lines *printed on the next page***.

(a) Produce and document Haskell data type declarations for one-dimensional cellular automata.

(b) Give a type signature for the function calculating the next generation from an old generation for a one-dimensional cellular automaton.

(c) Give a definition for that function.

(d) Give a definition of an element of your one-dimensional cellular automata data type from (a) for the Sierpinski cellular automaton.

(e) Using the next-generation function from (b, c) and the Sierpinski automaton data from (d), define the function *runSierpinksi* such that the invocation "*runSierpinksi* 64" in Hugs or GHCi interaction in a sufficiently large terminal produces the above representation of the example run.

2. Two-Dimensional Cellular Automata

(a) Define data types and transition function for Conway's "Game of Life", a well-known two-dimensional cellular automaton. (You can find information for example from http://cafaq.com/.)

(b) Define a way to animate your "Game of Life" — an easy way would be to use XTerm control sequences. For example, in an xterm (or rxvt), try "*putStr* "\ESC[H\ESC[2J"", (and read also about character literal escape sequences in the Haskell report).

(c) Define an interesting starting state involving for example at least two gliders as part of your solution. Program this starting state using appropriate abstractions (such as "glider").