1.1 Rigid Models

Rigid computations have another node parameter: ***location*** or ***cell***. Combined with time, it designates the event uniquely. Locations have ***structure*** or ***proximity*** edges between them. They (or their short chains) indicate all neighbors of a node to which pointers may be directed.

Cellular Automata (CA)

CA are a parallel rigid model. Its sequential restriction is the ***Turing Machine (TM)***. The configuration of CA is a (possibly multi-dimensional) grid with a finite, independent of the grid size, alphabet of ***states*** to label the events. The states include, among other values, pointers to the grid neighbors. At each step of the computation, the state of each cell can change as prescribed by a ***transition*** function (also called program) applied to the previous states of the cell and its pointed-to neighbors. The initial state of the cells is the input for the CA. All subsequent states are determined by the transition function.

An example of a possible application of CA is a VLSI (very large scale integration) chip represented as a grid of cells connected by wires (chains of cells) of different lengths. The propagation of signals along the wires is simulated by changing the state of the wire cells step by step. The clock interval can be set to the time the signals propagate through the longest wire. This way the delays affect the simulation implicitly.

An example: the Game of Life (GL). GL is a plane grid of cells, each holds a 1-bit state (dead/alive) and pointers to the 8 adjacent cells. A cell remains dead or alive if the number i of its live neighbors is 2. It becomes (or stays) alive if i=3. In all other cases it dies (of overcrowding or loneliness) or stays dead.

A ***simulation*** of a machine M_1 by M_2 is a correspondence between memory configurations of M_1 and M_2 which is preserved during the computation (may be with some time dilation). Such constructions show that the computation of M_1 on any input x can be performed by M_2 as well. GL can simulate any CA (see a sketch of an ingenious proof in the last section of [Berlekamp, Conway, Guy 82]) in this formal sense:

We fix space and time periods a, b. Cells (i, j) of GL are mapped to cell $(\lfloor i/a \rfloor, \lfloor j/a \rfloor)$ of CA M (compressing $a \times a$ blocks). We represent cell states of M by states of $a \times a$ blocks of GL. This correspondence is preserved after any number t steps of M and bt steps of GL regardless of the starting configuration.

Turing Machines (TMs)

TM is a *minimal CA*. Its configuration – ***tape*** – is an infinite to the right chain of cells.
Each state of a cell has a pointer to one of the cell's two adjacent neighbors. No adjacent cells can both point away from each other. Only the two cells pointing at each other are ***active***, i.e. can change state.
The cell that just turned its pointer is the TM's moving head working on the tape symbol - its target.
The input is an array of non-blank cells (only one is rightward) followed by blanks at the right.

Another type of CA represents a TM A with several non-communicating heads. At most $O(1)$ heads fit in a cell. They can vanish, split, or merge only in the first cell (which, thus, controls the number of active cells). The input x makes an unchangeable "ink" part of each cell's state. The rest of the cell's state is in "pencil" and can be changed by A. The computation halts when all heads drop off. The output $A(x)$ is the pencil part of the tape's state. This model has convenient theoretical features. E.g. with linear (in T) number ($\|p\|^2 T$) of state changes (volume) one can solve the ***Bounded Halting Problem*** $H(p, x, T)$: find out whether the machine with a program p stops on an input x within volume T of computation (see 2.3).

Exercise: Find a method to transform any given multi-head TM A into another one B such that the value of the output of $B(x)$ (as a binary integer) and the volumes of computation of $A(x)$ and of $B(x)$ are all equal within a constant factor (for all inputs x). **Hint:** B-cells may have a field to simulate A and maintain (in other fields) two binary counters h (with $\Theta(1)$ density of heads) for the number of heads of A and v for A's volume. Their least significant digits are at the leftmost cell. h adds its most significant digit to the same position in v at each step of A. To move the carry 1 on v a head is borrowed from h. These 1-heads move right in v till they empty their carry into its 0 digit. Then empty 0-heads move back to h in a separate field/track, possibly first continuing right to find a free slot in this return track. (The heads area in v extends to k-th cell only by dropping the carry there, with frequency $O(2^{-k})$. Then it shrinks to $O(1)$ in $O(k)$ steps since heads enter it slower than they move away.) Borrowed or returned heads make low or high head-density areas in h which shift left until absorbed at the leftmost cell.