In[ ]:=
```
SetDirectory[ParentDirectory[NotebookDirectory[]]];
Needs["SSSiCv100`"];
```

ECA (elementary cellular automata):

[Note: the **ONLY** reason for including an FAQ on ECAs here is that it provides a simpler example of simple rules resulting in complex behavior, and is extensively treated in the NKS book. Wolfram's index coding of ECAs is a very easy analog of the sessie enumeration we use.]

To define the rules for ECAs, you don't have a rule set such as for substitution systems. Instead, at each time "click" each cell (a bit, binary digit: 0 or 1) is replaced according to a specified algorithm, depending on its current value **and** those of its neighbors to the left and right. Each ECA is completely identified by its algorithm and its initial state.

Stephen Wolfram's insight was the realization that all possible ECAs could be enumerated in an unambiguous way, by listing in order the 2^3 = 8 possible values for each triplet of cells (3 bits: left neighbor, middle cell, right neighbor), and then reading off the new values (1 bit for each case) in the order. These 8 bits give rise to a decimal number between 0 and 255 = 2^8 - 1, which is the index of the ECA. See examples below.

(Dissenting viewpoint: http://generaltheory.webs.com/Ring-of-rules.pdf)

5/16/2016

I: A Cellular Automaton is a discrete model studied in mathematics, computability theory, physics, complexity sciences, theoretical biology and microstructure modelings. Other names for cellular automata include cellular spaces, tessellation automata, homogeneous structures, cellular structures, tessellation structures, and iterative arrays.

A cellular automaton consists of a grid of cells with an initial time state (t=0) and as time advances (t=1), a fixed rule determines a new state. Typically, the rule for updating old states into new states does not change over time and is applied to the grid simultaneously.

Conway's famous "Game of Life" is an example of a 2-d cellular automaton. (Find reference. Wikipedia?)

II: Elementary cellular automata are one dimensional and only have two possible states (labelled as 0 or 1). It is a very simple model of computation.

The Numbering System: There are 8 possible configurations for a cell and its two neighbors, and for each configuration 2 possible new states for the cell, and therefore 2^8 = 256 possible elementary cellular automata. Stephen Wolfram's code assigns each rule a number from 0 to 255. Each configuration is written in order, 111, 110,…, 001, 000, etc, and when a replacement rule occurs, the substitution likewise occurs in binary code.

(Note: It is an arbitrary design choice to order the ECA rule parts in ascending order right-to-left, but there is some logic to it, since 000 is the first pattern to look for, and 111 is the last (when considered as binary numbers), so this choice identifies each pattern as a bit of an 8-bit number, with the LSB (least significant bit) representing the 000 pattern, the MSB representing the 111 pattern:

Example:

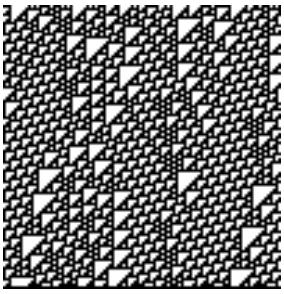
$110_d=96_d+14_d$ written in binary is 01101110_2. So rule 110 is defined by the transition rule:

111	110	101	100	011	010	001	000	current pattern	P=(L,C,R)
0	1	1	0	1	1	1	0	new state for center cell	N_{110_d}= (C+R+C*R+L*C*R)%2

(Note: The only benefit in writing 110 as 96 + 14 above is to think of it as two hexadecimal numbers rather than one byte, or 8 bits. The way I think of it is:

110 (base 10) = 64 + 32 + 8 + 4 + 2
= 2^6 + 2^5 + 2^3 + 2^2 + 2^1
= 0*2^7 + 1*2^6 + 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0
= 01101110 (base 2)

After that we use the bits found as the list of new states to use as shown in the table.)



More details: https://en.wikipedia.org/wiki/Rule_110

For a complete illustration of every one of the 256 rules:
https://en.wikipedia.org/wiki/Elementary_cellular_automaton