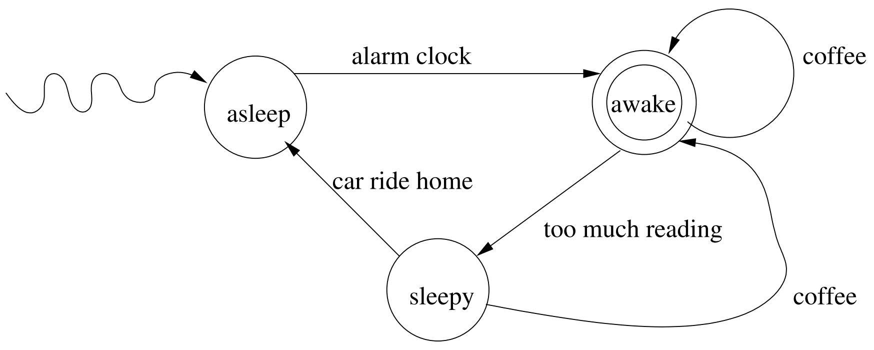
Everything you need to know about Finite Automata/Finite State Machines

Courtesy of Troy Vasiga (`troy.vasiga@uwaterloo.ca`)
Continuing Lecturer
David R. Cheriton School of Computer Science
University of Waterloo

1 Motivation

- We are all in various *states* at various times: happy, sad, tired, sleeping, eating, snoring, ...
- Sometimes, we change states based on what people say, what the weather is,...: these *inputs* change our state
- We have to start in a *starting state*
- We call the day good if we end up in a good *final state*. If we don't end up in a good final state, we reject that day, and hope that it never happens again
- CPU's are really just Finite State Machines (they do a very limited number of things (like adding, subtracting, moving information) to a finite number of inputs)

2 A picture



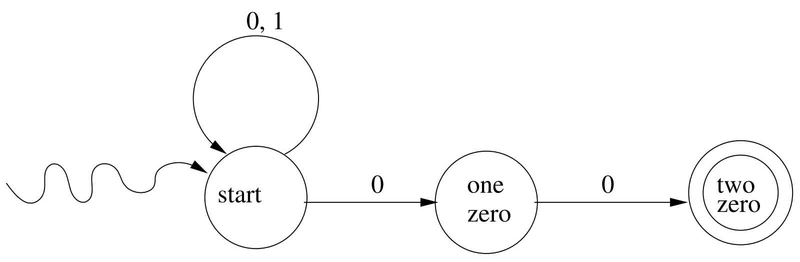
Note that the start state is "asleep" and our final state is "awake".

3 Deterministic Finite Automata

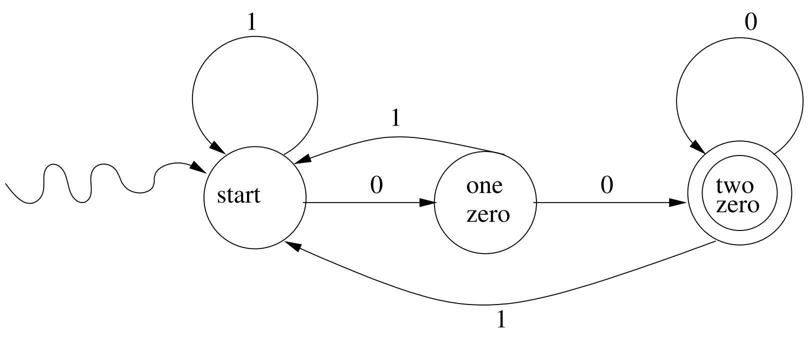
The above picture is a deterministic finite automata (DFA). *Deterministic* means there is at most one state I can go to if I have read a particular input.

4 Nondeterministic Finite Automata

Here is a NFA, which accepts all strings composed of 0's and 1's which end in two 0's:



You can notice that reading 00 places you (or your clones) in all three states simultaneously. Of course, all NFA's can be converted into a DFA. So, we have the equivalent DFA for the same language, shown below:



As an exercise, draw a NFA with 4 states which accepts all words composed of 0's and 1's which have 010 as a substring (so 10101 and 00101010 have 010 as a substring). Then, try to draw a DFA for this same language.

5 Lambda (λ)

Sometimes, we need to move from one state to another without reading any input. In order to do this, we use the Greek symbol lambda (λ) to denote "nothing." The use of λ is useful for two main reasons:

1. Combine together two (or more) FSM into one. For example, if you had a FSM for $(abb^*)^*$ and another one for cc^*cc, we could combine them by creating a new start state, and connecting this new start start to the old start states by way of a λ transition.
2. Move between states by reading nothing. That is, suppose you want to allow an "optional" character (for example, in postal codes, you can write them as `N2L3G1` or `N2L 3G1`: the space is optional). So, we can use a transition which has both space and λ on it in to allow this optional space.

6 Regular Expressions

Sometimes, finite automata seem like a clumsy way of writing down a language. We would like a more formal (and shorter) way of writing a language. We use *regular expressions* in this case.

So, for the language of all words composed of 0's and 1's which have 010 as a substring, we would have a regular expression like:

$$(0|1)^*(010)(0|1)^*$$

This says, we can begin with any number of 0's or 1's (including zero 0's and zero 1's), then we must have 010 in the string, followed by any number of 0's or 1's.

7 Web Resources

The hands-on lab session used a program called JFLAP, which can be used on the web or download onto your own machine.

The original files (for download) can be found at:

`http://www.jflap.org`