Computer Science 1MD3

Lab 8 – Deterministic Finite State Machines

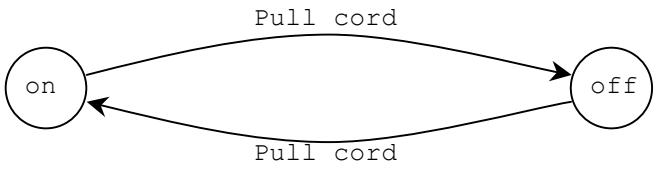
A directed graph is a graph in which the edges that connect the nodes have an implied direction, that is, you may have an edge that connects A to B but not B to A. A finite state machine is a directed graph where the nodes represent certain states and the edges between represent transitions.

SIMPLE FINITE STATE MACHINES

In its simplest form, a machine may be regarded as a 'black box' with an input and output channel. Whenever we put something into the machine, the machine acts on it in some way and produces output. This is analogous to a mathematical function.

Consider a lamp, which is turned off and on by pulling a cord. This lamp has two states, *off* and *on*, which are transitioned by '*pull cord*'. The corresponding finite state machine will look as follows:

FSM 1

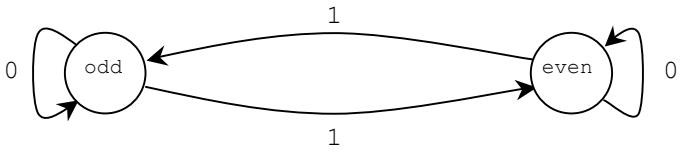


Of course we will need to know a *starting state*, which we will denote by S. Without a starting state any transition would be meaningless since we would not know which node to transition from. In FSM 1 we will define S=`on`.

PARITY MACHINES

Consider a machine which determines if the number of ones in a binary number is odd or even. Note that we will assume that an empty binary number contains an even number of ones, which implies that S=`even`.

PARITY MACHINE



Lets verify this machine by the applying the two following examples, 10010110 and 10101.

10010110 S=even
(1) → odd, (0) → odd, (0) → odd, (1) → even, (0) → even, (1) → odd, (1) → even.

10101 S=even
(1) → odd, (0) → odd, (1) → even, (0) → even, (1) → odd.

INPUT LANGUAGE

The parity machine could *recognize* any sequence of ones and zeros; a sequence with say a two in it, like 100120102 would be *unrecognizable*. Now say we further restrict the sequences by only allowing those that end in the odd state to be recognized. Any sequence satisfying these criteria is a *word* and would be in the *language* of the parity machine.

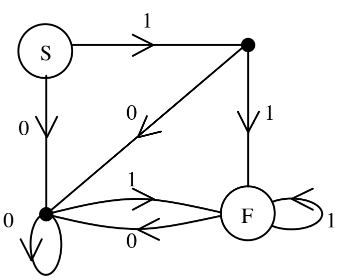
In general we have:

> **The language of a finite state machine**
>
> Let S and F be the start state and final state for a finite state machine respectively. The language of a finite state machine is the set of all sequences consisting of recognizable elements that start at S and end at F.

If we denote a recognizable sequence by Φ and define $x^n \equiv \underbrace{x x \cdots x}_{n-times}$ where $n \in \mathrm{Z}$ and $n \in F$ where $F = \{recognizable\ \ input\}$, (i.e. $1^4 0^3 = 1111000$) we could define the language for any finite state machine quite elegantly.

Consider:

FSM 2



The sequences 11, 101, and 01, are all in the language of FSM 2. All other sequences in the language should boil down to these sequences after eliminating redundancies and loops. A redundancy is an element in your sequence that doesn't bring you to a new state, i.e. 0000001=0(00000)1 where (00000) doesn't have any effect on the final state. A loop is part of your sequence that starts and ends in the same place, i.e. 1101=11(01) where removing (01) doesn't effect the arrival at F.

Starting at 11 we may start adding redundancies and loops without consequently changing the outcome of the sequence. At F, 1^{n1} is a redundancy since it doesn't cause a change of state and the loop 01 that has its own redundancy 0^{n2}, (so really $0^{n2}1$), leaves F only to arrive back at F. So we have $\Phi_1 = [11][1^{n1}][0^{n2}1]$, however, it is fairly clear that we can add a redundancy or a loop as many times as we want, so $\Phi_1 = [11][[1^{n1}][0^{n2}1]]^{n3}$.

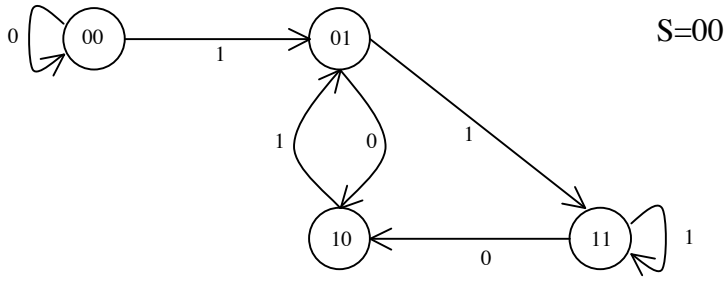
With similar process we can arrive at the following language

> **FSM 2 input language**
>
> $$\Phi_1 = [11][[1^{n1}][0^{n2}1]]^{n3} \quad \Phi_2 = [10][0^{n1}1^{n2\geq 1}]^{n3} \quad \Phi_3 = [0^{n1>1}1^{n2>1}]^{n3}$$

N-MOMENT DELAY MACHINE

A moment delay machine is a machine that can 'remember' the last N digits of input. The following diagram illustrates a two-moment delay machine that 'remembers' the two previous inputs of a binary number.

Two-moment delay machine



Self test problems

1) Draw the three-moment delay machine that 'remembers' the list three digits of binary input.

2) What is the language of the following machines?