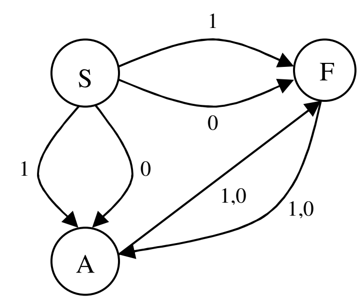
Computer Science 1MD3

Lab 9 – Nondeterministic Finite State Machines

The finite state machines discussed last week are **deterministic**, since for each state and input value there is a unique next state given by the transition. There is another important type of finite-state machine in which there may be several possible next states for each pair of input value and state. That is, state A may go to state B *or* C on input 1. Such machines are called **nondeterministic** finite state machines.

A SIMPLE EXAMPLE

NDFSM 1



Given that we are starting at S, is {111} a *word*, i.e. does {111} end in the F state?

We immediately get stuck at S because a decision has to made to go to either A or F. To deal with this ambiguity we simple go to both A and F, getting the transitions S-A-F-A and S-F-A-F. Since the later of these transitions ends in F we say that {1,1,1} is word in the language.

LANGUAGE OF A NDFSM

What does it mean for a nondeterministic finite-state machine to recognize a sequence $\Phi = \{x_1 x_2 ... x_n\}$ given the states $s_0 ... s_n$.

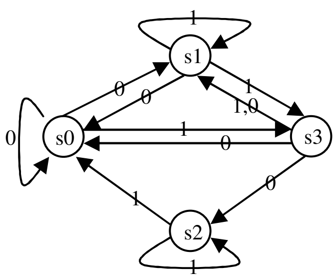
The first input x_1 takes us from s_0 to a set S_1 of states. The next input x_2 takes each of the sets in S_1 to a new set of states. Let S_2 be the union of these sets. We continue this process till we exhaust all elements in the sequence leaving us with S_n. $\Phi = \{x_1 x_2 ... x_n\}$ is a word in the language if any of the transitions end in one the final states. That is if $A = \{\text{final states}\}$, $A \cap S_n$ is nonempty.

To solidify this idea consider the following finite state machine and note that it is sometimes convenient to represent a FSM or NDFSM as a chart. This often clarifies an otherwise confusing digraph.

We will test the sequence $x = \{0, 1, 1, 0, 1\}$ using the technique describe previously and see if it is a valid word. We also let $s_n(x_n)$ be the set of all transitions reached from state s_n given input x_n.

NDFSM 2

Table		
State	**Input**	
	0	**1**
s_0	s_0, s_1	s_3
s_1	s_0	s_1, s_3
s_2		s_0, s_2
s_3	s_0, s_1, s_2	s_1
start = s_0		
end = F = $\{s_2, s_3\}$		



$$x_0 = 0 \rightarrow S_1 = \{s_0(0)\} \rightarrow S_1 = \{s_0, s_1\}$$

$$x_1 = 1 \rightarrow S_2 = \{s_0(1), s_1(1)\} \rightarrow S_2 = \{\vdots\, s_3, \vdots\, s_1, s_3 \vdots\} = \{s_1, s_3\}$$

$$x_2 = 1 \rightarrow S_3 = \{s_1(1), s_3(1)\} \rightarrow S_3 = \{\vdots\, s_1, s_3 \vdots\, s_1 \vdots\} = \{s_1, s_3\}$$

$$x_3 = 0 \rightarrow S_4 = \{s_1(0), s_3(0)\}$$
$$\rightarrow S_4 = \{\vdots\, s_0 \vdots\, s_0, s_1, s_2 \vdots\} = \{s_0, s_1, s_2\}$$

$$x_4 = 1 \rightarrow S_5 = \{s_0(1), s_1(1), s_2(1)\}$$
$$\rightarrow S_5 = \{\vdots\, s_3, \vdots\, s_1, s_3, \vdots\, s_0, s_2 \vdots\} = \{s_0, s_1, s_2, s_3\}$$

Where $S_5 \cap F = \{s_2, s_3\}$, which is non-empty so x is in the language.

NDFSM vs FSM

If the language *L* is recognized by a nondeterministic finite-state machine M_0, then *L* is also recognized by a deterministic finite-state machine M_1.

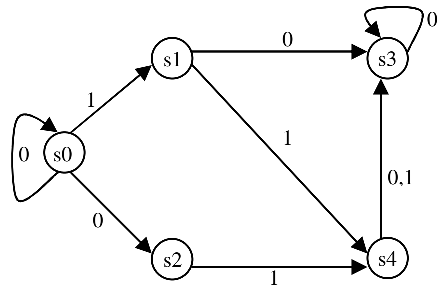
There is a proof to this which would be easy to understand, but it is beyond the scope of this lab. We encourage the student to formulate the proof as an exercise.

So given NDFSM 3, find a deterministic finite-state machine that recognizes the same language as the nondeterministic finite-state machine.

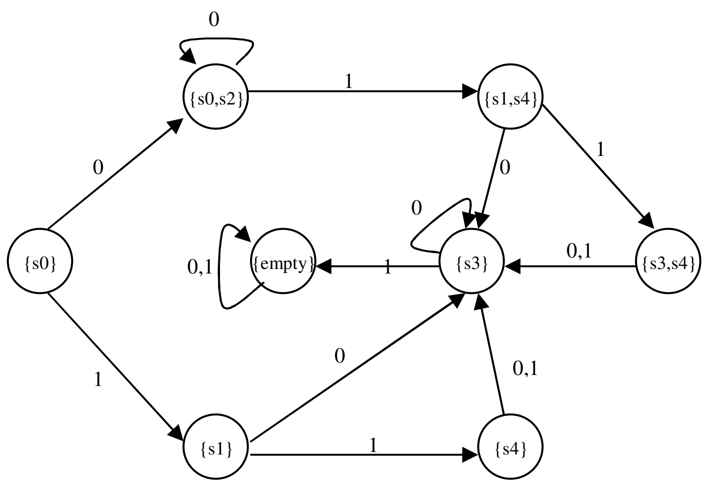
Answer:
The states of the FSM are the subsets of the set of all states of the NDFSM. For instance, on input of 0, $\{s_0\}$ goes to $\{s_0, s_2\}$. On input one, the corresponding set $\{s_0, s_2\}$ goes to $\{s_0(1), s_2(1)\} = \{s_1, s_4\}$. Through similar process we can generate all such relations arriving at FSM describe on the next page.

NDFSM 3

Table		
State	**Input**	
	0	1
s_0	s_0, s_2	s_1
s_1	s_3	s_4
s_2		s_4
s_3	s_3	
s_4	s_3	s_3
Start = s_0		
end = $\{s_0, s_4\}$		



FSM 3



Self test Problem

1. Draw the NDFSM for the following table and determine its language.

Table		
State	**Input**	
	0	**1**
S_0	S_0, S_1	S_2
S_1	S_0	S_1, S_3
S_2	S_1, S_3	
S_3	S_0, S_1, S_2	S_1
start = s_0		
end = F = $\{s_0, s_3\}$		

2. Draw the corresponding deterministic finite-state for this.