

CPSC 362 Lecture - poof

Chris Nutter*

September 16, 2020

Timestamps

07:38:48 PM

Okay so he's talking mainly about the project and how far along people are. People tend to not be incredibly far only a handful of people have created an FSM. He said he is considering adjusting the project depending on our position and understanding of the project.

07:48:16 PM

Now he is going back to talking about how to implement a DFMS into code. He's sorta doing psuedo-code mentioned below. *Figure 4*

08:06:36 PM

Taking a break then going to go over non-deterministic FSM (and NFAs).

08:13:54 PM

You can get 90% on the project if you document a FSM and diagram it without code. FYI. Basically 90% for desired output, 100% for intended FSM.

08:48:38 PM

Basically he's been making corrolations between video game idles and NFSM. He also mentioned that this will be useful for regexp next week.

09:11:51 PM

Next time, we are converting NFSM to DFMS. So make sure to understand FSM. lol

*Dedicated to @QuesoGrande

Contents

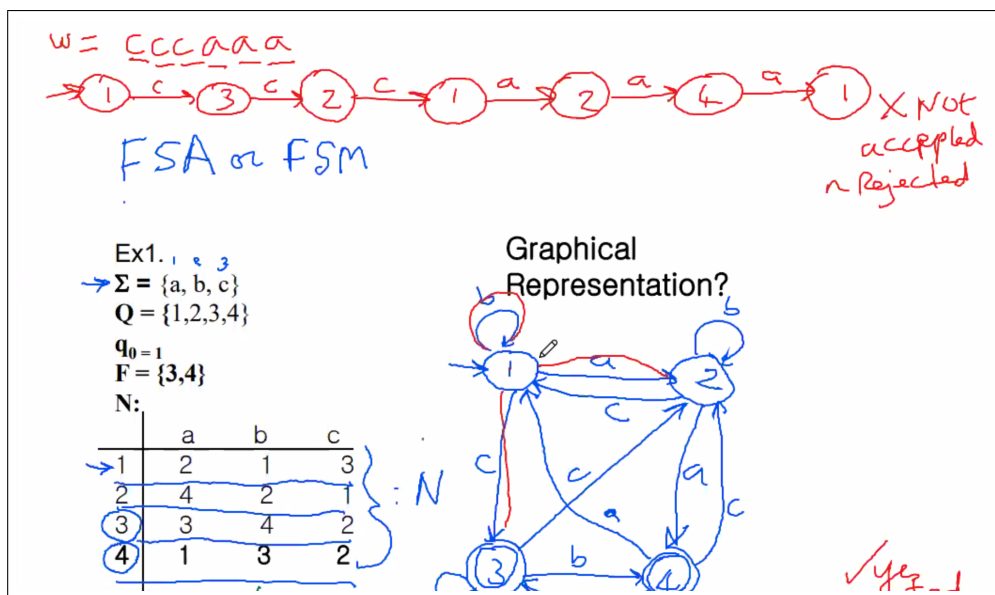
1	FSM Recap	3
2	Chapter 2.2 - Deterministic FSM	4
3	Chapter 2.3 - Non-Deterministic FSM (NFSM)	5

1 FSM Recap

A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.

An FSM is defined by a list of its states, its initial state, and the inputs that trigger each transition. Finite-state machines are of two types—deterministic finite-state machines and non-deterministic finite-state machines.

A deterministic finite-state machine can be constructed equivalent to any non-deterministic one.



2 Chapter 2.2 - Deterministic FSM

Ex. Given the following FSM

$M = ($
 $\Sigma = \{0,1\}$
 $Q = \{A,B,C,D\}$
 $q_0 = A$
 $F = \{B,D\}$
 $N:$

	0	1
A	B	A
B	C	C
C	D	B
D	A	D

)

$w = 11001$

✓ yes accepted

11

c) Implementation of a DFSM

```

function DFSM (ω : string)
  table = array [1..nstates, 1..nalphabets] of integer; /* Table N
  for the transitions*/
  state = 1; (the starting state)
  for i = to length (ω) do
    {
      col = char_to_col (ω[i]);
      state = table[state, col];
    }
  if state is in F then return 1 /* accept */
  else return 0
  }
  
```

init {

loop {

final state

$\Sigma = \{a, b, c\}$ $q_0 = 1$
 $Q = \{1, 2, 3, 4\}$
 $F = \{3, 4\}$

$\omega = \text{aca}$
 Call DFSM (aca)

init {
 State = 1;
 for i = 1 to 3 do =>
 i = 1, col = 1, state = ntable (1,1) = 2
 i = 2, col = 3, state = ntable (2,3) = 1
 i = 3, col = 1, state = ntable (1,1) = 2
 }

State 2 is not in $F = \{3,4\}$
 => return 0(false)
 => This string is NOT accepted

N:

	a	b	c
q0=1	2	1	3
2	4	2	1
3	3	4	2
4	1	3	2

(1) (2) (3)
 ↑ ↓ | conversion problem

14

3 Chapter 2.3 - Non-Deterministic FSM (NFSM)

Def: NFSM = (Σ, Q, q_0, F, N)
 Where Σ = a finite set of input symbols
 Q = a finite set of states
 $q_0 \in Q$ is the starting state
 $F \subseteq Q$ is a set of accepting state
 $N: Q \times (\Sigma \cup \epsilon) \rightarrow P(Q)$

empty set = { }

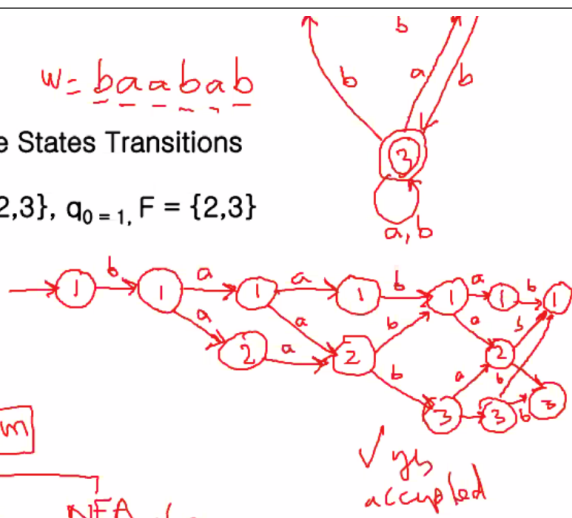
Two differences from DFSM

1. Input is expanded by epsilon => can go to a different state without reading any input
2. Can go to multiple states given an input

Ex 1. NFSM with Multiple States Transitions

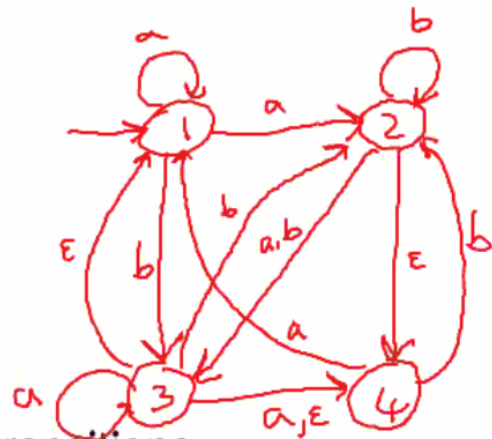
$M = (\Sigma = \{a,b\}, Q = \{1,2,3\}, q_0=1, F = \{2,3\})$
 N:

	a	b
→ 1	{1,2}	{1}
2	{2}	{1,3}
3	{2,3}	{1,3}



17

$\Sigma = \{a, b, \epsilon\}$
 $Q = \{1, 2, 3, 4\}$
 $q_0 = 1$
 $F = \{3\}$

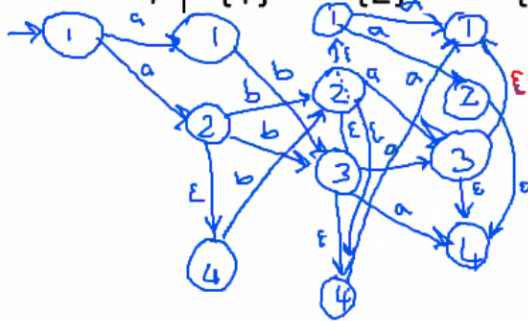


EX2) NSFM with Epsilon transitions

	a	b	ϵ
$q_0=1$	{1,2}	{3}	{ }
2	{3}	{2,3}	{4}
<u>3</u>	{3,4}	{2}	{1,4}
4	{1}	{2}	{ }

EX2) NSFM with Epsilon transitions

q0=	a	b	ϵ
1	{1,2}	{3}	{ }
2	{3}	{2,3}	{4}
<u>3</u>	{3,4}	{2}	{1,4}
4	{1}	{2}	{ }



empty set

$w = \underline{a} \underline{b} \underline{a}$