Theory of Automata

Nondeterministic Finite Automata

Dr. Sabina Akhtar

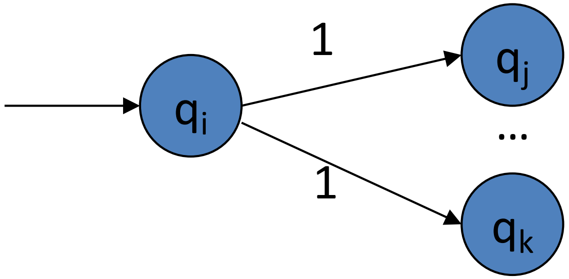
Revision

- Design DFA for
    - L = The set of all the strings whose 3rd last symbol is 0.

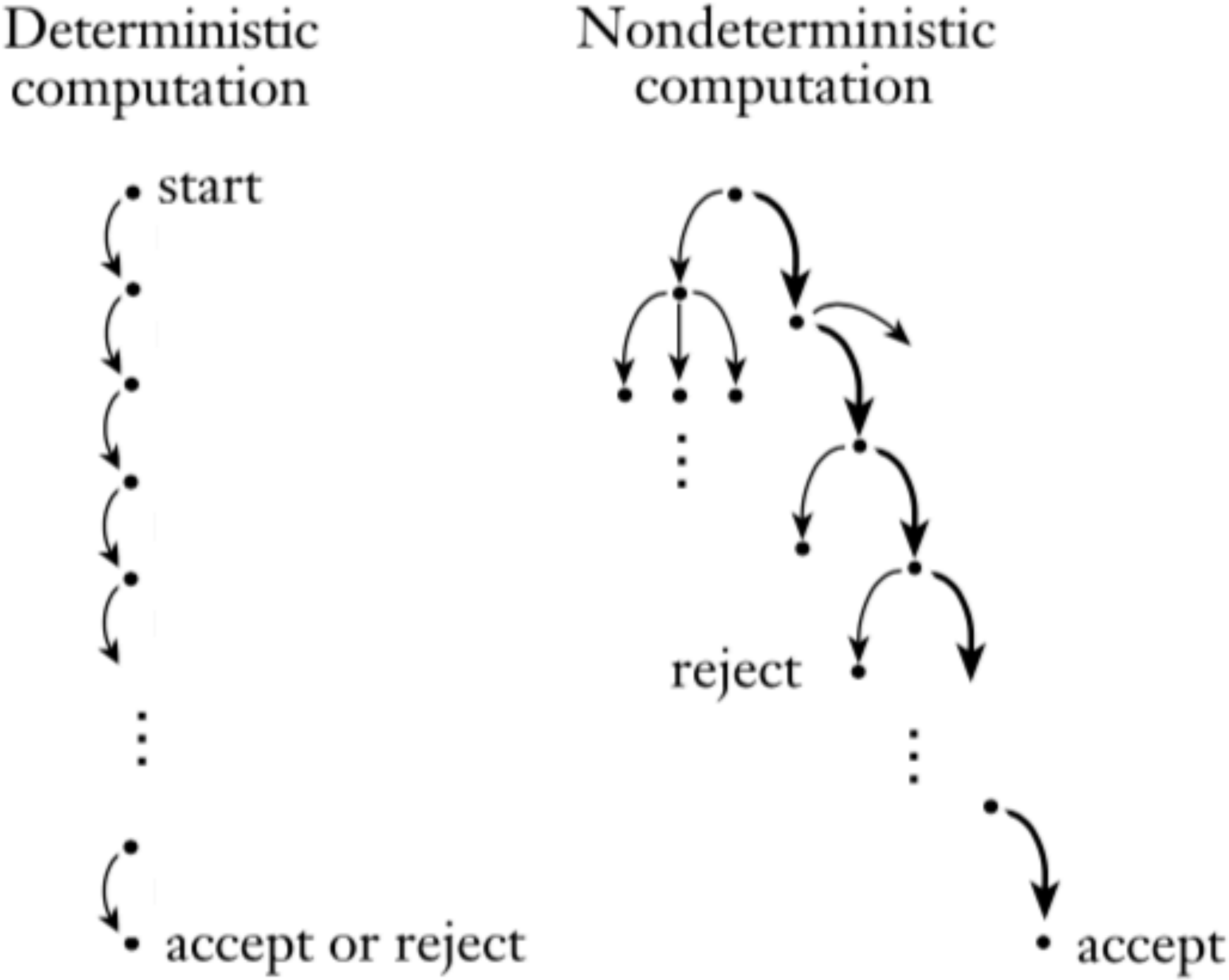
NONDETERMINISTIC FINITE AUTOMATA

Nondeterminism

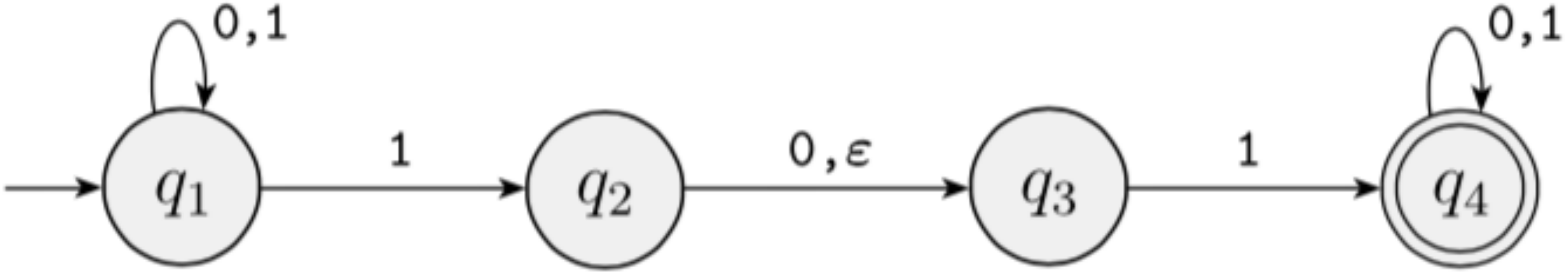
- A *nondeterministic finite automaton* has the ability to be in several states at once.
- Transitions from a state on an input symbol can be to any set of states.
  - Being non-deterministic



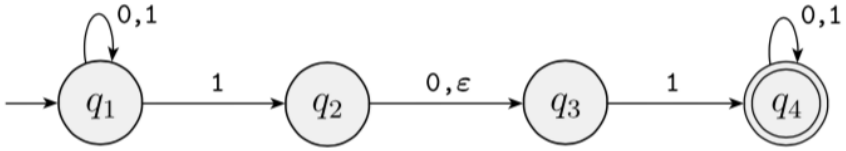
- Each transition function therefore maps to a set of states


Deterministic computation
Nondeterministic computation
start
accept or reject
reject
accept

Example

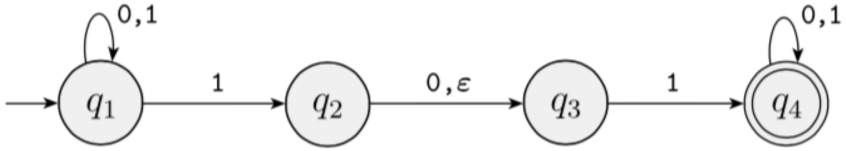


Example

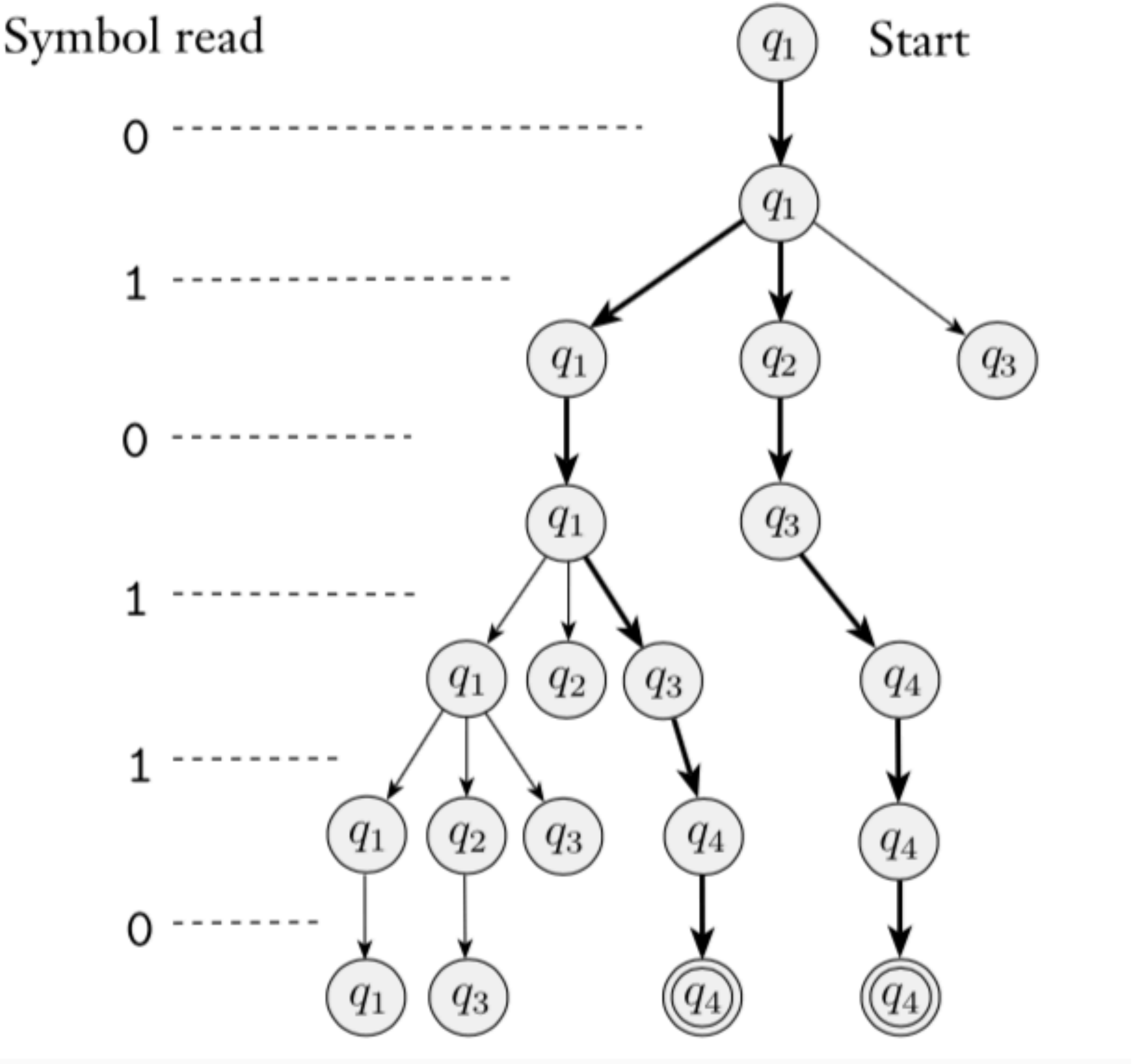


w = 010110

Example



w = 010110



Non-deterministic Finite Automata (NFA)

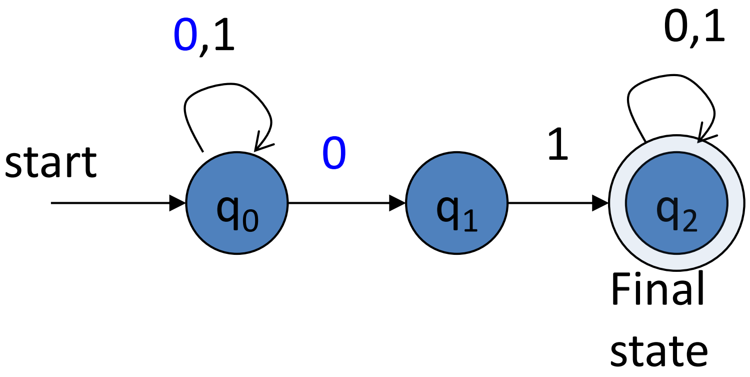
- A Non-deterministic Finite Automaton (NFA) consists of:
  - Q ==> a finite set of states
  - ∑ ==> a finite set of input symbols (alphabet)
  - q_0 ==> a start state
  - F ==> set of accepting states
  - δ ==> a transition function, which is a mapping between Q x ∑ ==> subset of Q
- An NFA is also defined by the 5-tuple:
  - {Q, ∑ , q_0,F, δ }

How to use an NFA?

- Input: a word w in ∑*
- Question: Is w acceptable by the NFA?
- Steps:
  - Start at the “start state” q_0
  - For every input symbol in the sequence w do
    - Determine all possible next states from all current states, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed and if at least one of the current states is a final state then *accept w;*
  - Otherwise, *reject w.*

NFA for strings containing 01

Why is this non-deterministic?



What will happen if at state q_1 an input of 0 is received?

- Q = $\{q_0,q_1,q_2\}$
- $\Sigma = \{0,1\}$
- start state = q_0
- F = $\{q_2\}$
- Transition table

symbols

δ	**0**	**1**
→ **q_0**	$\{q_0,q_1\}$	$\{q_0\}$
q_1	Φ	$\{q_2\}$
***q_2**	$\{q_2\}$	$\{q_2\}$

states

Example

- Build an NFA for the following language:

  L = { w | w ends with 111 as a substring}
- Provide formal specification and transition table as well.

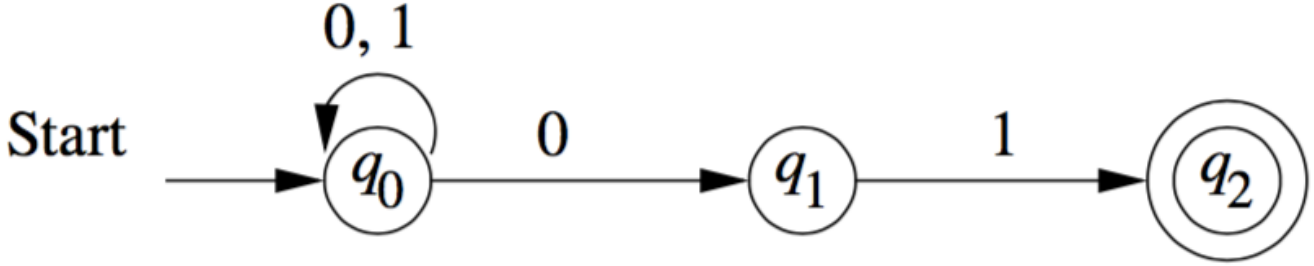
Class Activity

- Build an NFA for the following language:
  L = { w | w contains a 1 on its 3rd last position}
- Provide formal specification and transition table as well.

Language of an NFA

- An NFA accepts *w* if *there exists at least one* path from the start state to an accepting (or final) state that is labeled by *w*
- $L(N) = \{ w \mid \hat{\delta}(q_0,w) \cap F \neq \Phi \}$

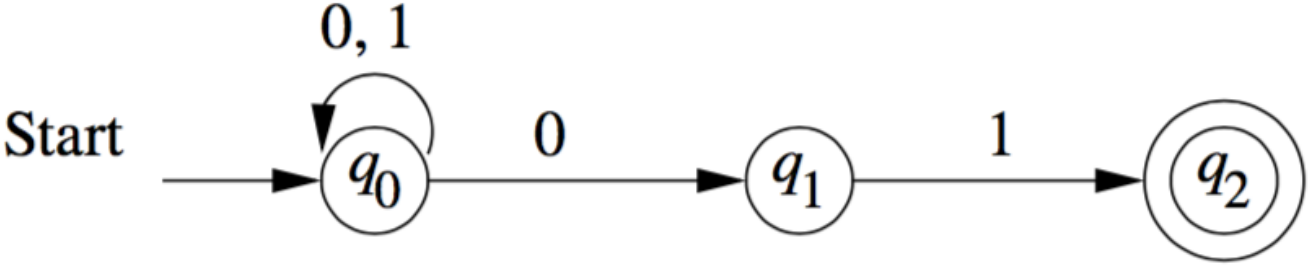
Extended Delta



- Show the processing of extended delta using the above NFA for 00101.

w = 00101

Extended Delta



1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$.

Class Activity– Extended delta

		symbols	
	δ	**0**	**1**
states	**→q_0**	$\{q_0,q_1\}$	$\{q_0\}$
	q_1	Φ	$\{q_2\}$
	***q_2**	$\{q_2\}$	$\{q_2\}$

Show the processing of extended delta using the above transition table.

$$\hat{\delta}(q_0,110101) =?$$

Class Activity

- Build an NFA for the following language:

  L = { w | w ends in 111}

- Provide formal specification and transition table as well.

FA with ε-Transitions

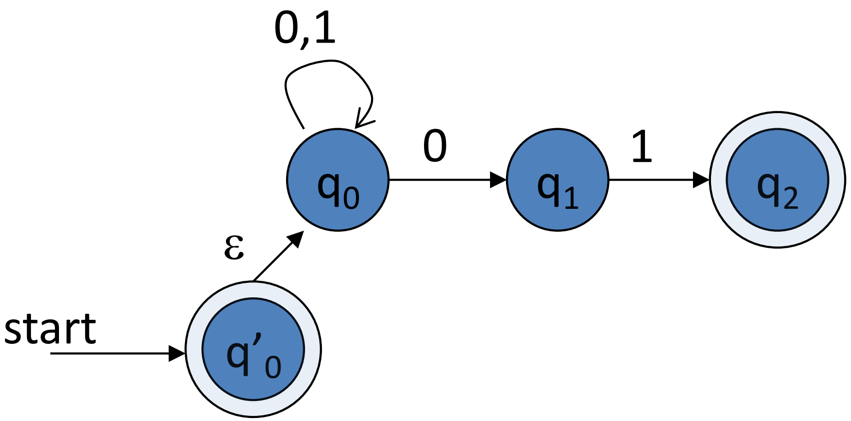
- We can allow explicit ε-transitions in finite automata
  - i.e., a transition from one state to another state without consuming any additional input symbol
  - Explicit ε-transitions between different states introduce non-determinism.
  - Makes it easier sometimes to construct NFAs

Definition: ***ε -NFAs are those NFAs with at least one explicit ε-transition defined.***

- ε -NFAs have one more column in their transition table

Example of an ε-NFA

L = {w | w is empty, or if non-empty will end in 01}



- ε-closure of a state q, ***ECLOSE(q)***, is the set of all states (including itself) that can be *reached* from q by repeatedly making an arbitrary number of ε-transitions.

δ_E	0	1	ε
→ *q'_0	Ø	Ø	$\{q'_0, q_0\}$
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	Ø	$\{q_2\}$	$\{q_1\}$
*q_2	Ø	Ø	$\{q_2\}$

ECLOSE(q'_0)

ECLOSE(q_0)

ECLOSE(q_1)

ECLOSE(q_2)

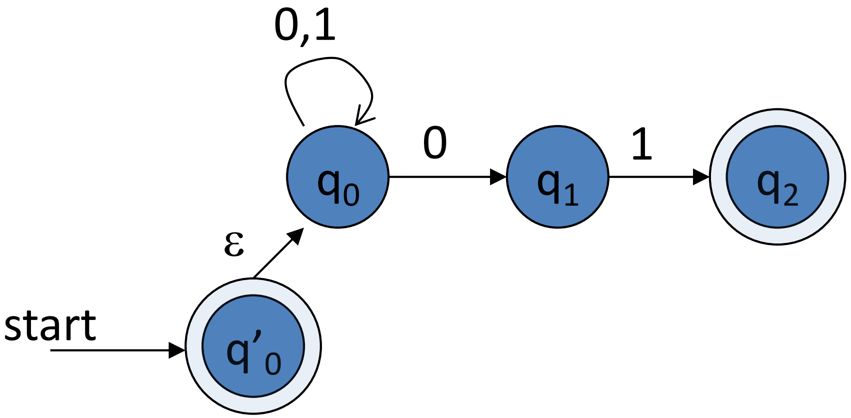
To simulate any transition:

Step 1) Go to all immediate destination states.

Step 2) From there go to all their ε-closure states as well.

Example of an ε-NFA

L = {w | w is empty, or if non-empty will end in 01}



Simulate for w=101:

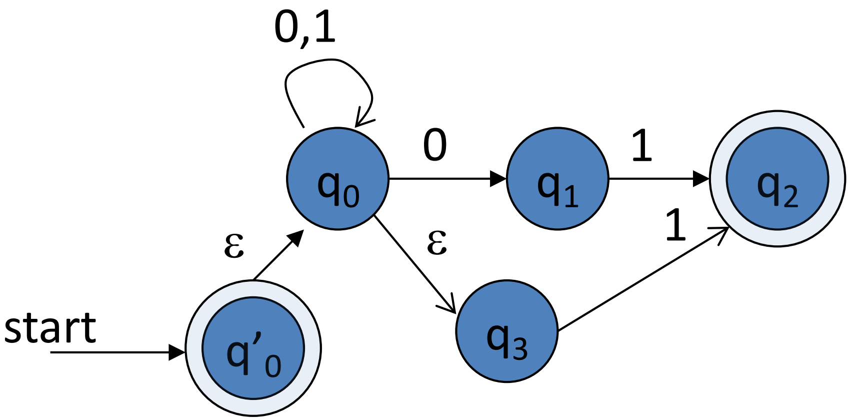
δ_E	0	1	ε
→ *q'_0	Ø	Ø	$\{q'_0,q_0\}$
q_0	$\{q_0,q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	Ø	$\{q_2\}$	$\{q_1\}$
*q_2	Ø	Ø	$\{q_2\}$

To simulate any transition:

Step 1) Go to all immediate destination states.

Step 2) From there go to all their ε-closure states as well.

Example of another ε-NFA



δ_E	0	1	ε
→ *q'_0	Ø	Ø	$\{q'_0,q_0,q_3\}$
q_0	$\{q_0,q_1\}$	$\{q_0\}$	$\{q_0,q_3\}$
q_1	Ø	$\{q_2\}$	$\{q_1\}$
*q_2	Ø	Ø	$\{q_2\}$
q_3	Ø	$\{q_2\}$	$\{q_3\}$

Simulate for w=101, w = 111

?

Differences: DFA vs. NFA

- **DFA**

1. All transitions are deterministic
   - Each transition leads to exactly one state
2. For each state, transition on all possible symbols (alphabet) should be defined
3. Accepts input if the last state visited is in F
4. Sometimes harder to construct because of the number of states

- **NFA**

1. Some transitions could be non-deterministic
   - A transition could lead to a subset of states
2. Not all symbol transitions need to be defined explicitly (if undefined will go to an error state – this is just a design convenience, not to be confused with "non-determinism")
3. Accepts input if *one of* the last states is in F
4. Generally easier than a DFA to construct

References

- Book Chapter 2
- Lectures from Stanford University
  - – http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE%20NOTES
- Lectures from Washington State University
  - – http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/