Chapter Five:
Nondeterministic Finite Automata

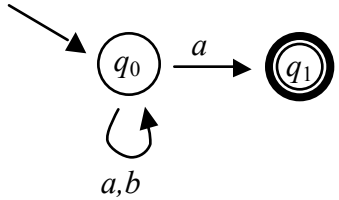
From DFA to NFA

- A DFA has exactly one transition from every state on every symbol in the alphabet.
- By relaxing this requirement we get a related but more flexible kind of automaton: the nondeterministic finite automaton or NFA.

Outline

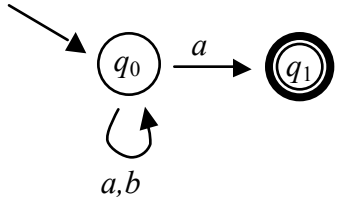
- 5.1 Relaxing a Requirement
- 5.2 Spontaneous Transitions
- 5.3 Nondeterminism
- 5.4 The 5-Tuple for an NFA
- 5.5 The Language Accepted by an NFA

Not A DFA



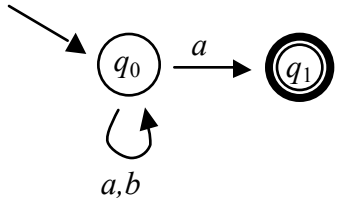
- Does not have exactly one transition from every state on every symbol:
  - Two transitions from q_0 on a
  - No transition from q_1 (on either a or b)
- Though not a DFA, this can be taken as defining a language, in a slightly different way

Possible Sequences of Moves



- We'll consider all possible sequences of moves the machine might make for a given string
- For example, on the string *aa* there are three:
  - From q_0 to q_0 to q_0, rejecting
  - From q_0 to q_0 to q_1, accepting
  - From q_0 to q_1, getting stuck on the last *a*
- Our convention for this new kind of machine: a string is in $L(M)$ if there is *at least one* accepting sequence

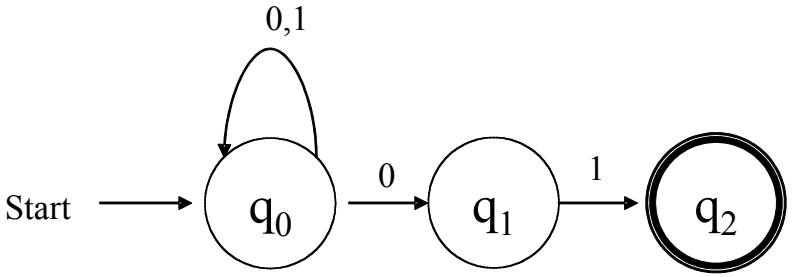
Nondeterministic Finite Automaton (NFA)



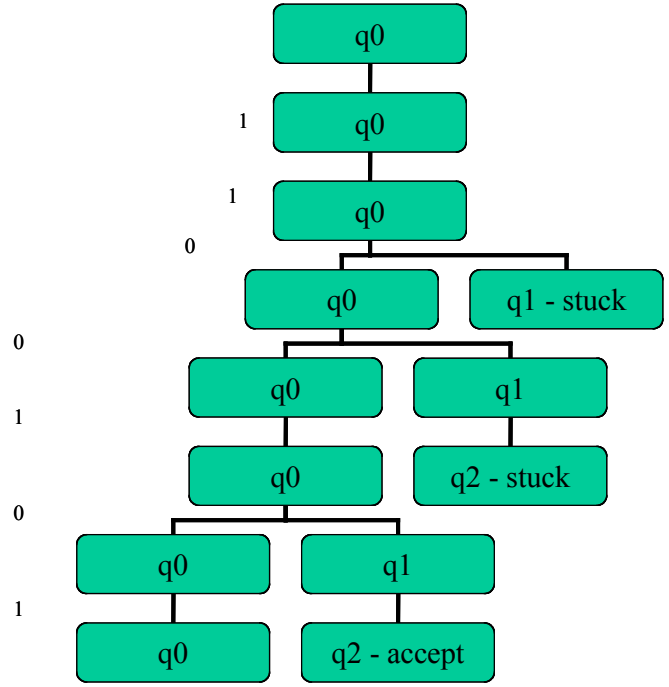
- $L(M)$ = the set of strings that have *at least one* accepting sequence
- In the example above, $L(M) = \{xa \mid x \in \{a,b\}^*\}$
- A DFA is a special case of an NFA:
  - An NFA that happens to be deterministic: there is exactly one transition from every state on every symbol
  - So there is exactly one possible sequence for every string
- NFA is *not necessarily* deterministic

NFA Example

- This NFA accepts only those strings that end in 01



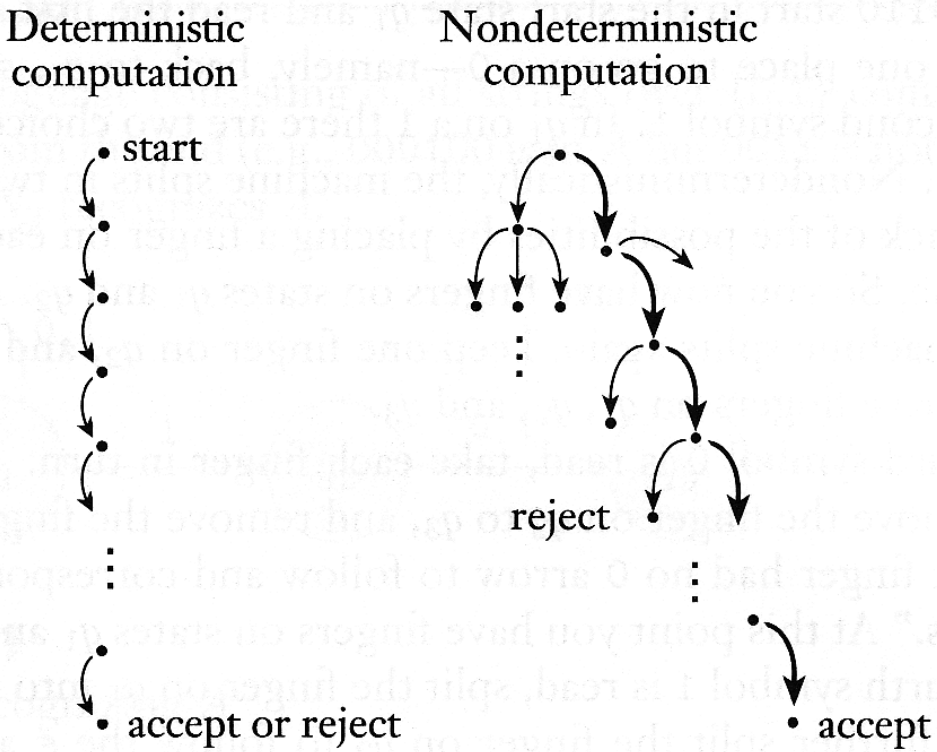
- Running in “parallel threads” for string 1100101



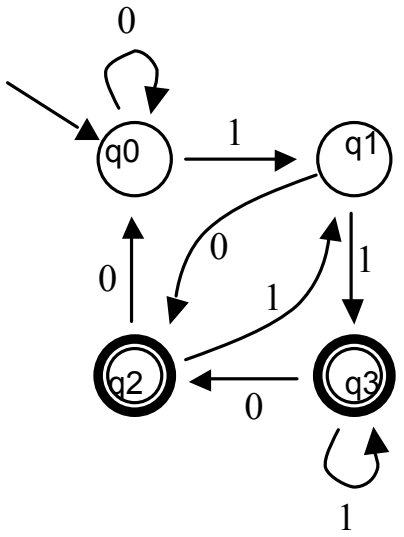
Nondeterminism

- The essence of nondeterminism:
  - For a given input there can be more than one legal sequence of steps
  - The input is in the language if at least one of the legal sequences says so
- We can achieve the same result by computing all legal sequences in parallel and then deterministically search the legal sequences that accept the input, but…
- ...this nondeterminism does not directly correspond to anything in physical computer systems
- In spite of that, NFAs have many practical applications

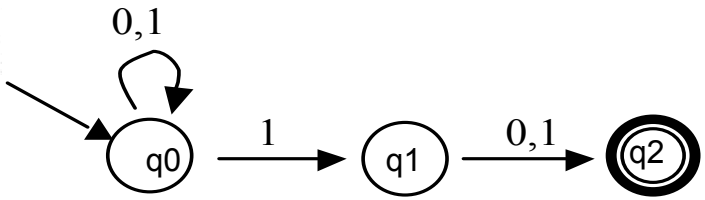
Nondeterminism



DFA:



NFA:



Now consider string: 0110

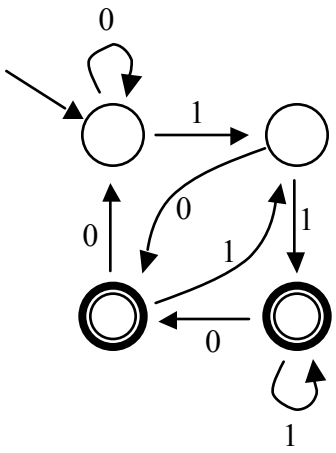
DFAs and NFAs

- DFAs and NFAs both define languages
- DFAs do it by giving a simple computational procedure for deciding language membership:
  - Start in the start state
  - Make one transition on each symbol in the string
  - See if the final state is accepting
- NFAs do it by considering all possible transitions *in parallel*.

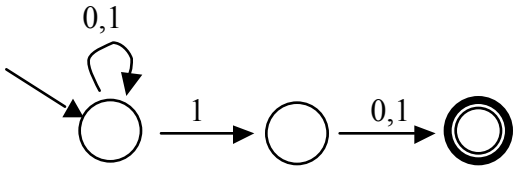
NFA Advantage

- An NFA for a language can be smaller and easier to construct than a DFA
- Let L={x ∈ {0,1}*|where x is a string whose next-to-last symbol is 1}
- Construct both a DFA and NFA for recognizing L.

DFA:



NFA:

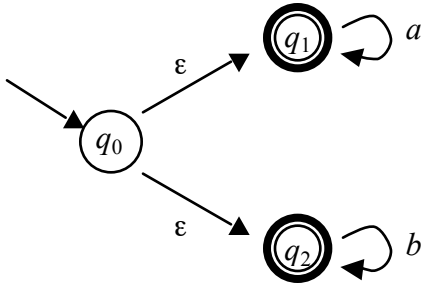


Outline

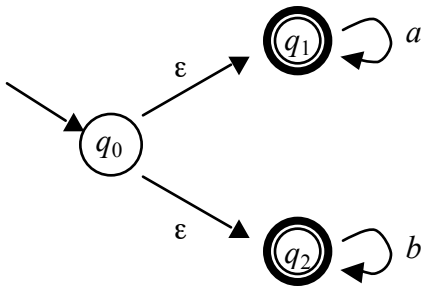
- 5.1 Relaxing a Requirement
- 5.2 Spontaneous Transitions
- 5.3 Nondeterminism
- 5.4 The 5-Tuple for an NFA
- 5.5 The Language Accepted by an NFA

Spontaneous Transitions

- An NFA can make a state transition spontaneously, without consuming an input symbol
- Shown as an arrow labeled with ε
- For example, {*a*}* ∪ {*b*}*:

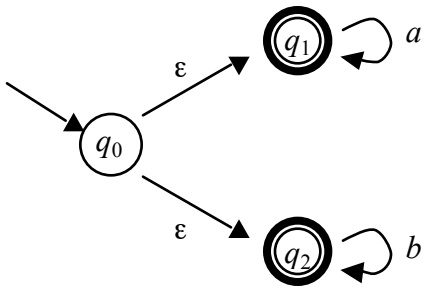


ε-Transitions To Accepting States



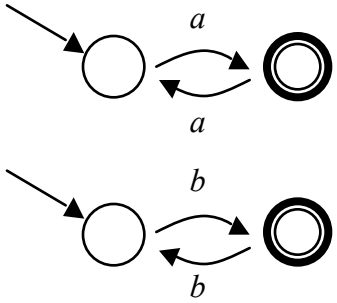
- An ε-transition can be made at any time
- For example, there are three sequences on the empty string
  - No moves, ending in q_0, rejecting
  - From q_0 to q_1, accepting
  - From q_0 to q_2, accepting
- Any state with an ε-transition to an accepting state ends up working like an accepting state too

ε-transitions For NFA Combining



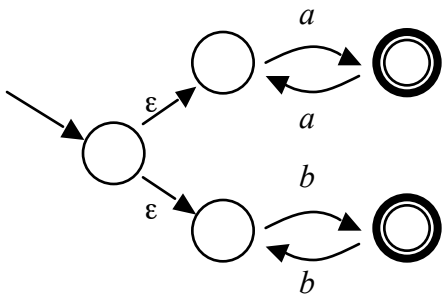
- ε-transitions are useful for combining smaller automata into larger ones
- This machine is combines a machine for {*a*}* and a machine for {*b*}*
- It uses an ε-transition at the start to achieve the union of the two languages

Revisiting Union



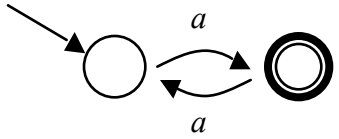
$A = \{a^n \mid n \text{ is odd}\}$

$B = \{b^n \mid n \text{ is odd}\}$

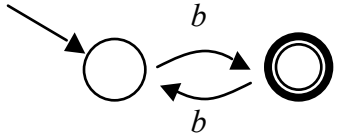


$A \cup B$

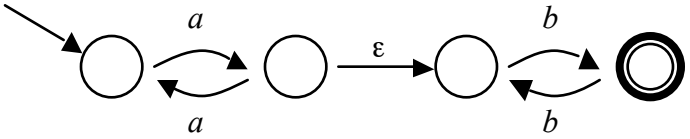
Concatenation



$A = \{a^n \mid n \text{ is odd}\}$



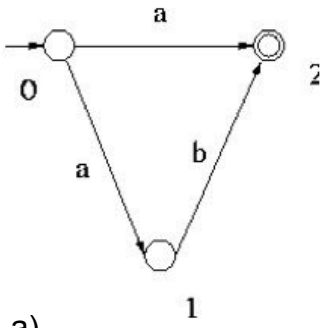
$B = \{b^n \mid n \text{ is odd}\}$



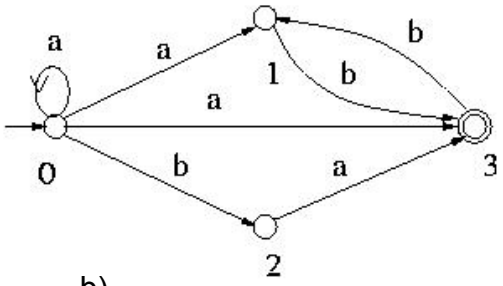
$\{xy \mid x \in A \text{ and } y \in B\}$

Some Exercises

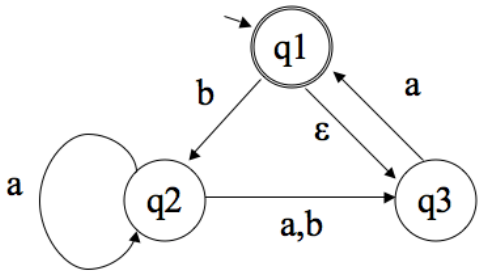
What is the language accepted by the following NFAs?



a)



b)

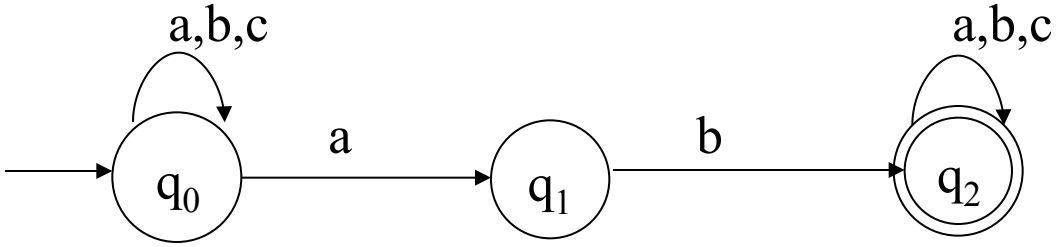


c)

More Exercises

- Let Σ = {a, b, c}. Give an NFA M that accepts:

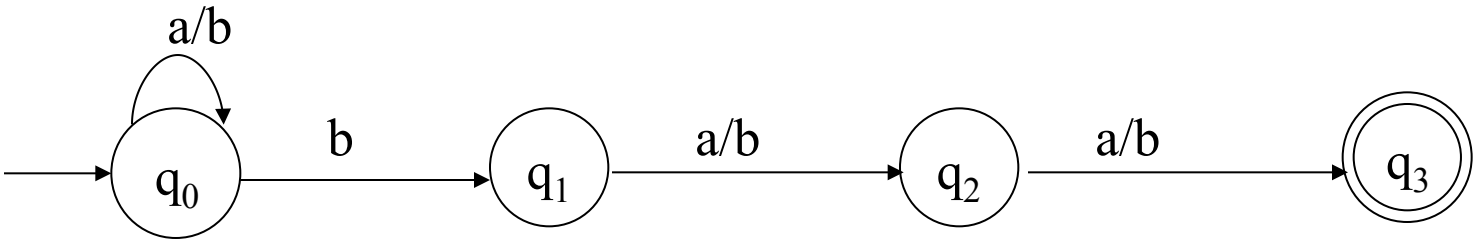
  L = {x | x is in Σ^* and x contains ab}



One More Exercise

- Let Σ = {a, b}. Give an NFA M that accepts:

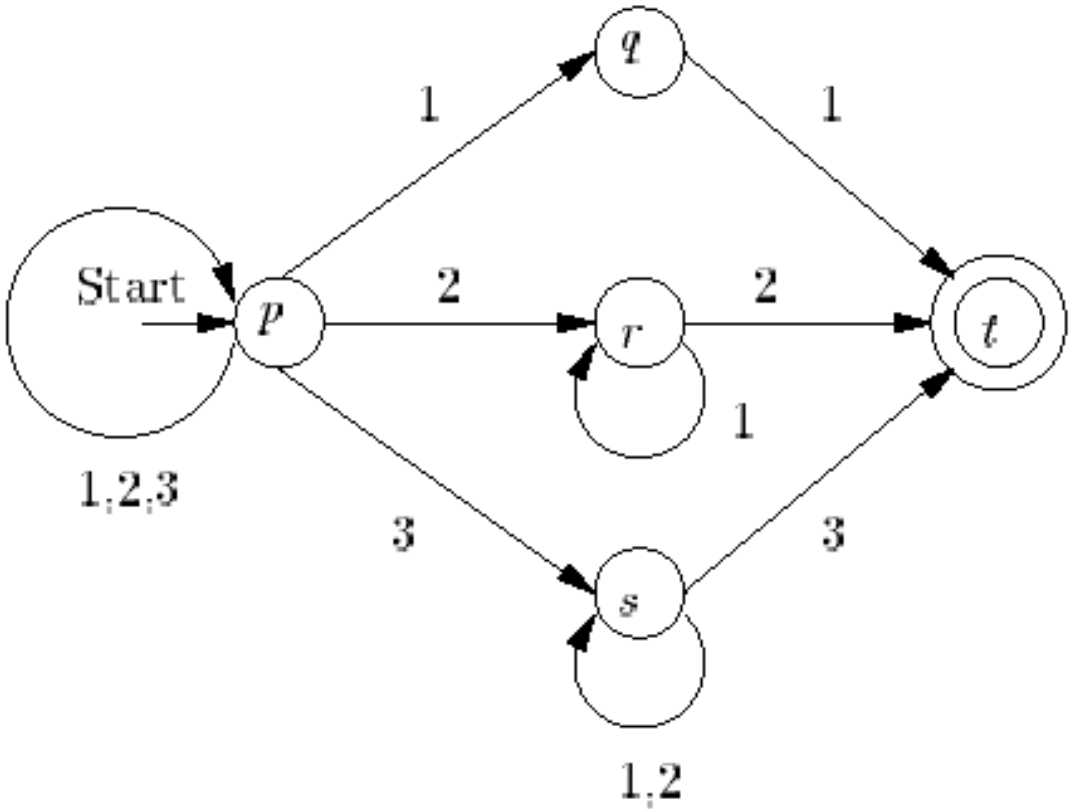
  L = {x | x is in Σ^* and the third to the last symbol in x is b}



NFA Exercise

- Construct an NFA that will accept strings over alphabet {1, 2, 3} such that the last symbol appears at least twice, but without any intervening higher symbol, in between:
  - e.g., 11, 2112, 123113, 3212113, etc.
- Trick: use start state to mean “I guess I haven't seen the symbol that matches the ending symbol yet.” Use three other states to represent a guess that the matching symbol has been seen, and remembers what that symbol is.

NFA Exercise (answer)



Outline

- 5.1 Relaxing a Requirement
- 5.2 Spontaneous Transitions
- 5.3 Nondeterminism
- 5.4 The 5-Tuple for an NFA
- 5.5 The Language Accepted by an NFA

Powerset

- If *S* is a set, the *powerset* of *S* is the set of all subsets of S:

$$P(S) = \{R \mid R \subseteq S\}$$

- This always includes the empty set and *S* itself
- For example,

P({1,2,3}) = {{}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}

The 5-Tuple

An NFA M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where:

Q is the finite set of states

Σ is the alphabet (that is, a finite set of symbols)

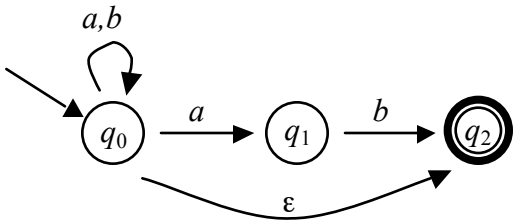
$\delta \in (Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accepting states

- The only change from a DFA is the transition function δ
- δ takes two inputs:
  - A state from Q (the current state)
  - A symbol from $\Sigma \cup \{\varepsilon\}$ (the next input, or ε for an ε-transition)
- δ produces one output:
  - A subset of Q (the set of possible next states - since multiple transitions can happen in parallel!)

Example:



- Formally, $M = (Q, \Sigma, \delta, q_0, F)$, where
  - $Q = \{q_0, q_1, q_2\}$
  - $\Sigma = \{a, b\}$ (we assume: it must contain at least *a* and *b*)
  - $F = \{q_2\}$
  - $\delta(q_0, a) = \{q_0, q_1\}$, $\delta(q_0, b) = \{q_0\}$, $\delta(q_0, \varepsilon) = \{q_2\}$,
    $\delta(q_1, a) = \{\}$, $\delta(q_1, b) = \{q_2\}$, $\delta(q_1, \varepsilon) = \{\}$
    $\delta(q_2, a) = \{\}$, $\delta(q_2, b) = \{\}$, $\delta(q_2, \varepsilon) = \{\}$
- The language defined is $\{a, b\}^*$

Outline

- 5.1 Relaxing a Requirement
- 5.2 Spontaneous Transitions
- 5.3 Nondeterminism
- 5.4 The 5-Tuple for an NFA
- 5.5 The Language Accepted by an NFA

The δ^* Function

- The δ function gives 1-symbol moves
- We'll define δ^* so it gives whole-string results (by applying zero or more δ moves)
- For DFAs, we used this recursive definition
  - $\delta^*(q,\varepsilon) = q$
  - $\delta^*(q,xa) = \delta(\delta^*(q,x),a)$
- The intuition is similar for NFAs taking parallel transitions into account, but the ε-transitions add some technical difficulties

NFA IDs

- An *instantaneous description* (ID) is a description of a point in an NFA's execution
- It is a pair (q,x) where
  - $q \in Q$ is the current state
  - $x \in \Sigma^*$ is the *unread* part of the input
- Initially, an NFA processing a string x has the ID (q_0,x)
- An accepting sequence of moves ends in an ID (f,ε) for some accepting state $f \in F$

The One-Move Relation On IDs

- We write

  $I \mapsto J$

  if I is an ID and J is an ID that could follow from I after one move of the NFA
- That is, for any string $x \in \Sigma^*$ and any $a \in \Sigma$ or $a = \varepsilon$,

$$(q, ax) \mapsto (r, x) \text{ if and only if } r \in \delta(q, a)$$

The Zero-Or-More-Move Relation

- We write

  $$I \vdash^* J$$

  if there is a sequence of zero or more moves that starts with *I* and ends with *J:*

  $$I \vdash \cdots \vdash J$$

- Because it allows zero moves, it is a *reflexive* relation: for all IDs *I*,

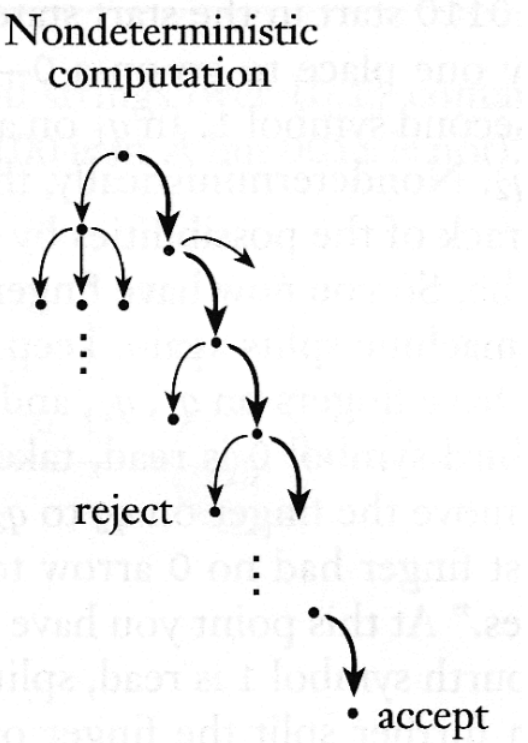
  $$I \vdash^* I$$

The δ* Function

- Now we can define the δ* function for NFAs:

$$\delta^*(q,x) = \left\{ r \,\middle|\, (q,x) \mapsto^* (r,\varepsilon) \right\}$$

- Intuitively, δ*(*q*,*x*) is the set of all states the NFA might be in after starting in state *q* and reading *x*



M Accepts *x*

- Now $\delta^*(q,x)$ is the set of states *M* may end in, starting from state *q* and reading all of string *x*
- So $\delta^*(q_0,x)$ tells us whether *M* accepts *x* by computing all possible states by executing all possible transitions in parallel on the string *x:*

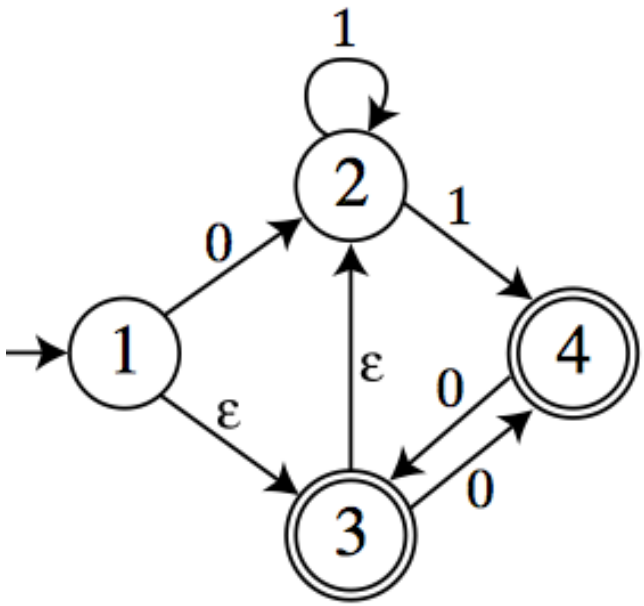
A string $x \in \Sigma^*$ is accepted by an NFA $M = (Q, \Sigma, \delta, q_0, F)$ if and only if the set $\delta^*(q_0, x)$ contains at least one element of *F*.

The Language An NFA Defines

For any NFA $M = (Q, \Sigma, \delta, q_0, F)$, $L(M)$ denotes the language accepted by M, which is

$$L(M) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \cap F \neq \{\}\}.$$

Exercise



- Compute the results of the following transitions:
  - $\delta^*(q1,\varepsilon)$
  - $\delta^*(q1,0110)$

Exercise

- **Theorem**: Let *M* be an NFA with a single accepting state, show how to construct the 5-tuple for a new NFA, say *N*, with

  $L(N) = \{ xy \mid x \in L(M) \text{ and } y \in L(M) \}$.

  Show that the language of construct NFA is indeed $L(N)$ as specified.
- **Proof Idea**: The idea here is to make two copies of the NFA, linking the accepting state of the first to the start state of the second. The accepting state of the second copy becomes the only accepting state in the new machine.

Assignment

- Assignment #3 – see website