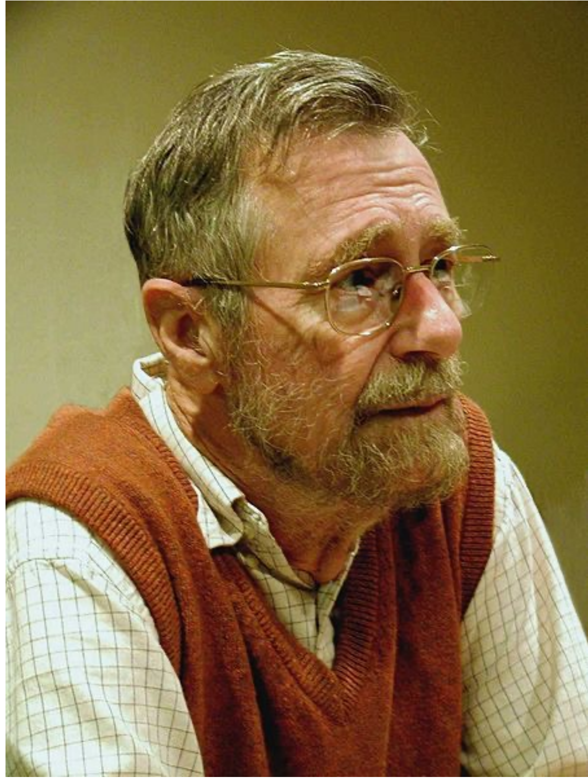
Theory of Computation
Chapter 1

Non-Deterministic Finite Automata

VCU

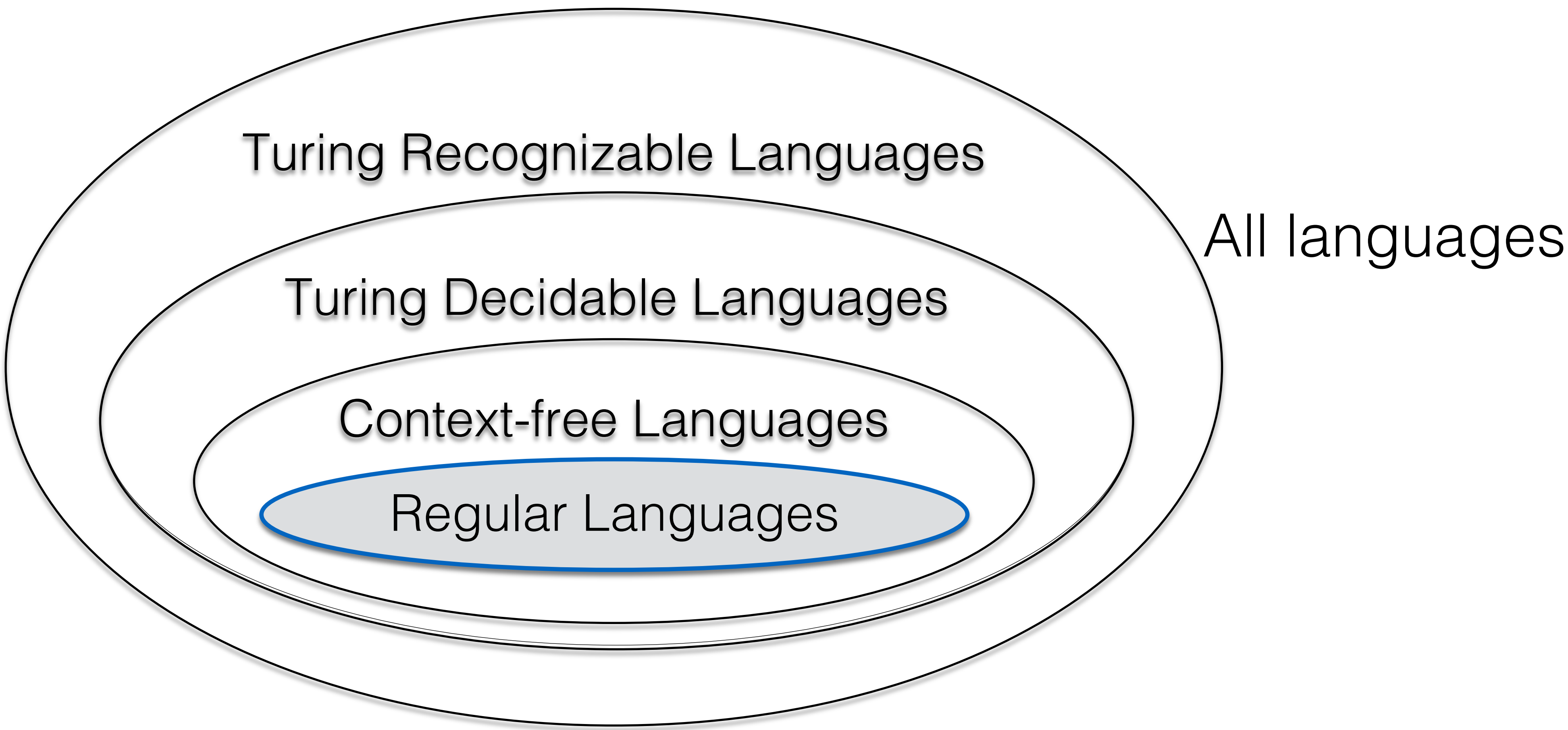
School of Engineering | Computer Science

Edsger Dijkstra 1930-2002



- Developed:
  - Shortest path algorithm (Dijkstra's algorithm)
  - Concept of a semaphore, Reverse Polish notation, the THE multiprogramming system, Banker's algorithm, CS sub-field of self stabilization….
- Wrote scathing commentary on (once) popular GOTO programming statement
- 1972 Turing award for programming language contributions
- "two or or more, use a for"
- "Computer science is no more about computers than astronomy is about telescopes."

Regular Languages



Regular Languages

DFA = NFA = RE

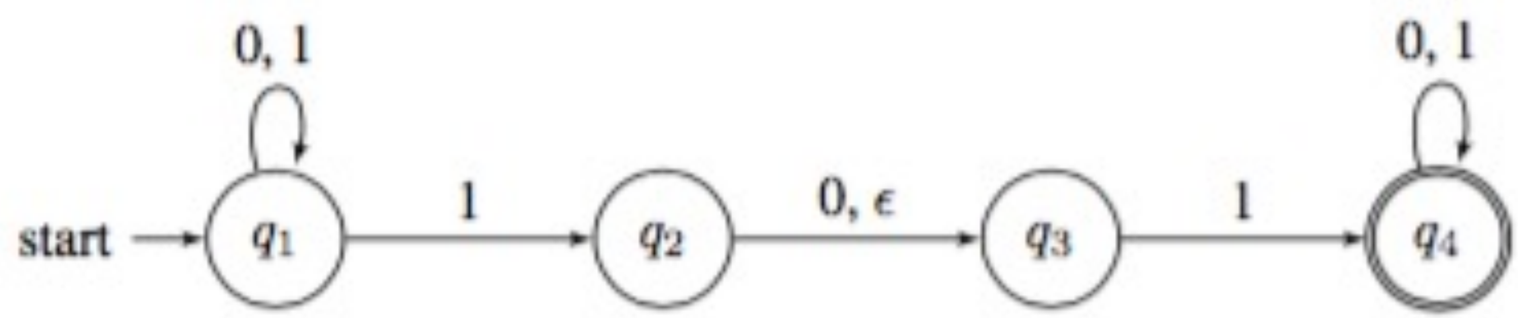
Closed under union, ∪, concatenation, ∘, and star, $*$.

Non-Deterministic Finite Automata (NFA)

- NFA = DFA where at each step you can follow multiple choices simultaneously
- Remember that a DFA has exactly one transition for each symbol at each state
  - This does not apply to the NFA
- The differences between an NFA and a DFA
  1. NFA's can label transitions with ε,the empty string
  2. Each symbol $a \in \Sigma$ can appear in 0, 1, or many transitions from a given state
     - What happens if the symbol a appears 0 times at a state?
       - Ans: There is no transition for a and the string is not accepted

Non-Deterministic Finite Automata, cont.

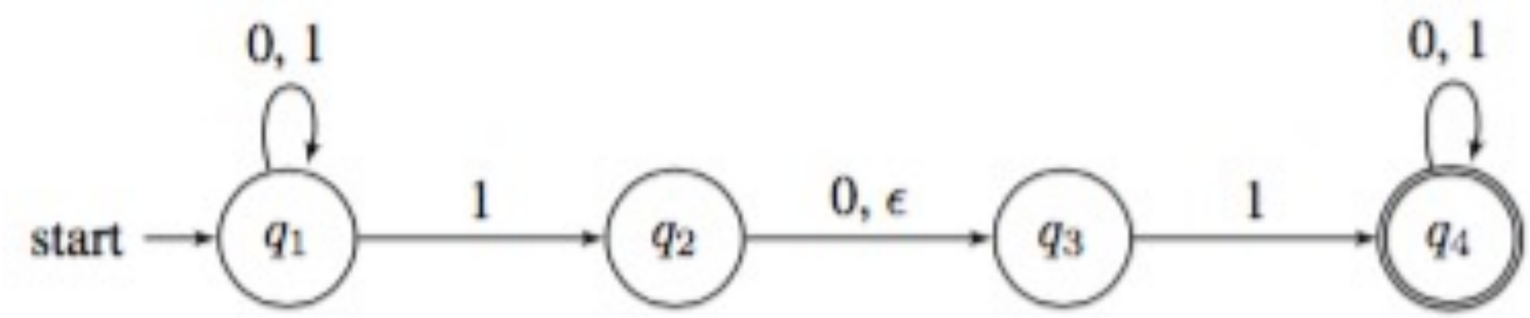
- Key Idea: An NFA will accept a string if there is any computational branch that leads to an accept state
- Ex: $L(N_1) = \{x \mid x \text{ contains } 11 \text{ or } 101 \text{ as a substring}\}$
  - NFA N_1:



  - What do you notice at q_1?
  - NFA accepts if any branch on the computational tree for an input string ends in an accept state.

Non-Deterministic Finite Automata, cont.

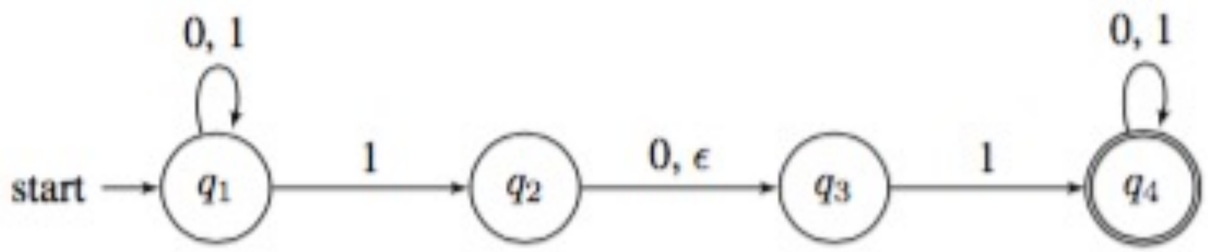
- Ex: $L(N_1) = \{x \mid x \text{ contains 11 or 101 as a substring}\}$
  - NFA N_1:



  - Creating the Computational Tree for string x = 010110 to determine if the string is accepted:
    - Follow each of the possible paths that the NFA can take for each symbol from the string x. Because it is an NFA, there is more than one possible path.
    - Like a maze where you hope that one path will lead to an accept state

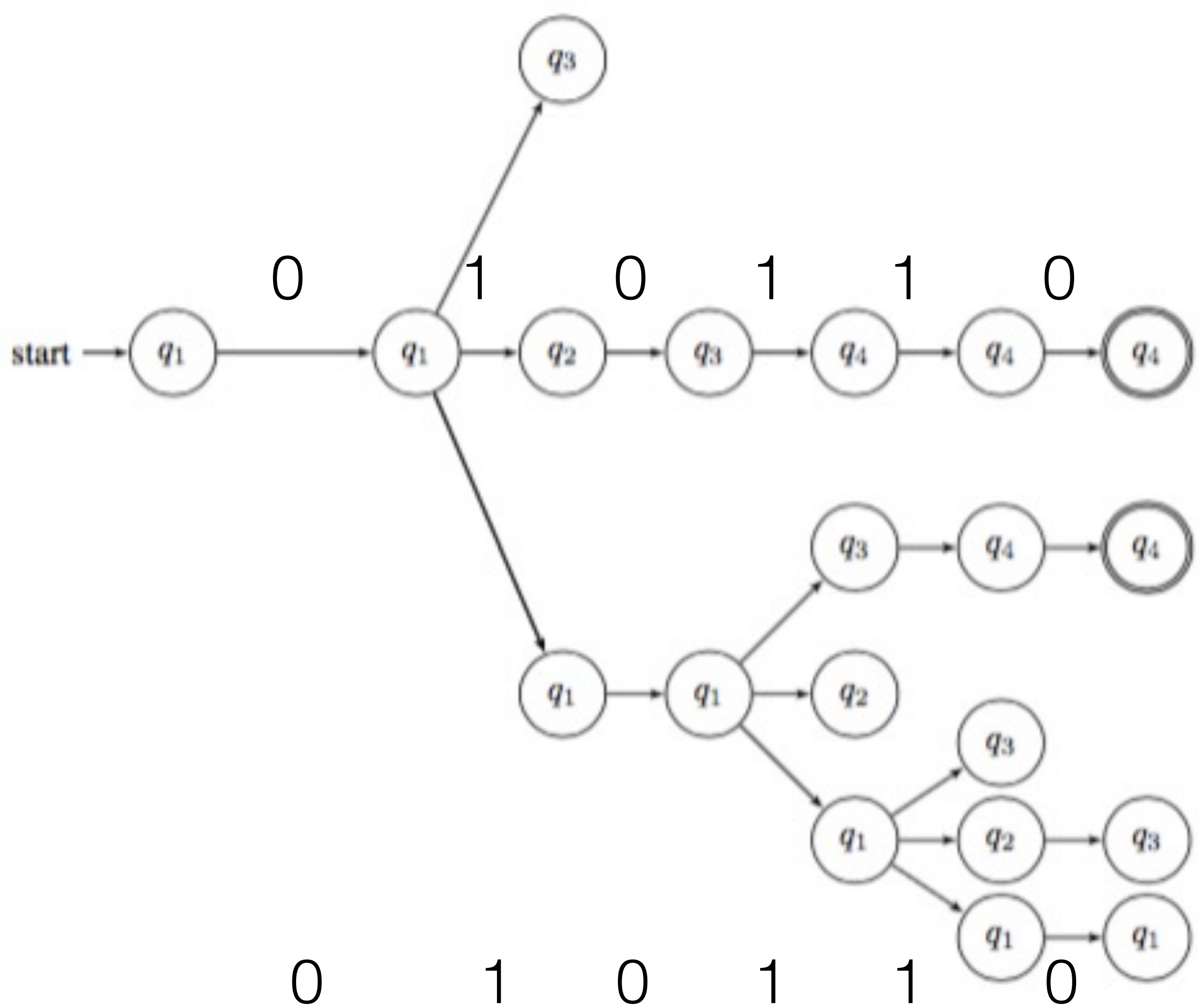
School of Engineering | Computer Science

Non-Deterministic Finite Automata, cont.



These are all the possible ways 010110 can be derived.

- Computational Tree for string x = 010110:

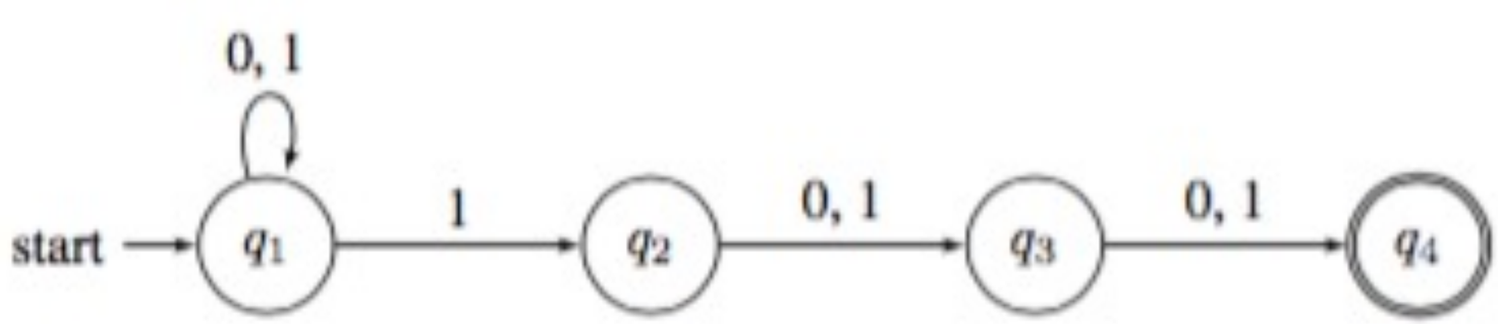


Note that the ε transition can move to the next state without reading a symbol from the string as in q_2 to q_3, which makes it look like q_1 goes directly to q_3.

The string is accepted since at least one branch ends at an accept state.

Designing NFA's

- Ex: NFA N_2:

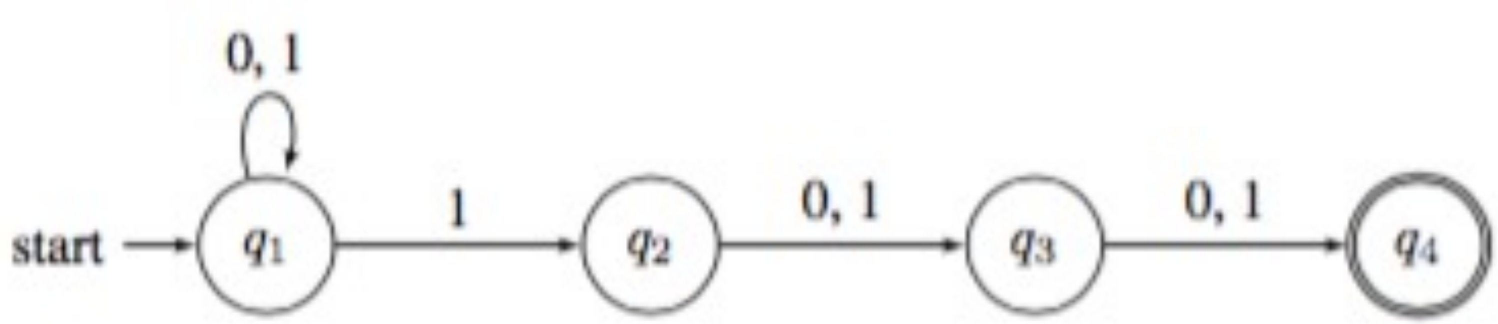


- What is the language that this NFA recognizes for strings with $\Sigma = \{0,1\}$?

School of Engineering | Computer Science

Designing NFA's

- Ex: NFA N_2:



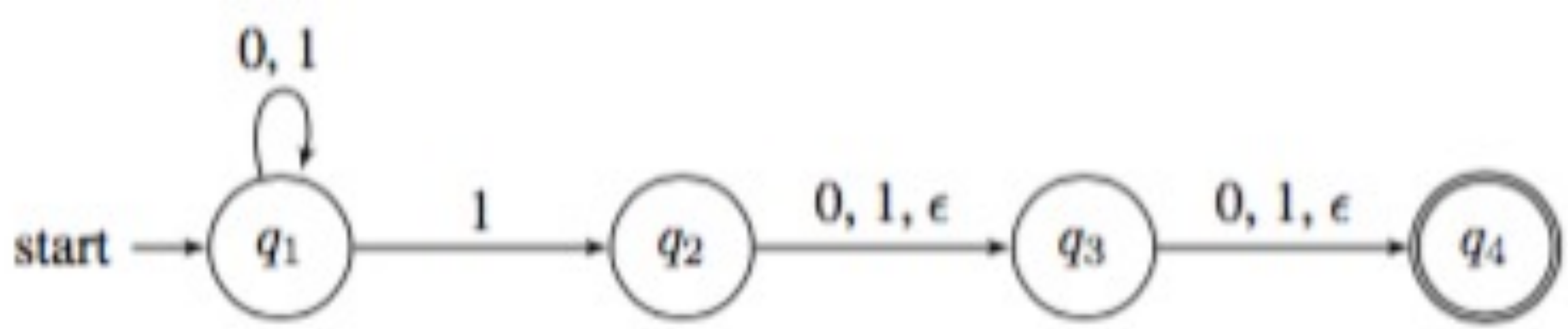
- What is the language that this NFA recognizes for strings with $\Sigma = \{0,1\}$?
  - $L(N_2) = \{x \mid x \in \Sigma^*1\Sigma^2\}$ (all strings where there is a 1 in the 3rd to last position from the end)

Designing NFA's

- Ex: $L(N_3)$ = {x | x has a 1 in at least one of the last three positions}
  - What is the NFA?

Designing NFA's

- Ex: $L(N_3)$ = {x | x has a 1 in at least one of the last three positions}
  - What is the NFA?
    - NFA N_3:



    - $L(N_3) = \{x \mid x \in \Sigma^*1\,\Sigma^2 \cup \Sigma^*1\,\Sigma^1 \cup \Sigma^*1\}$

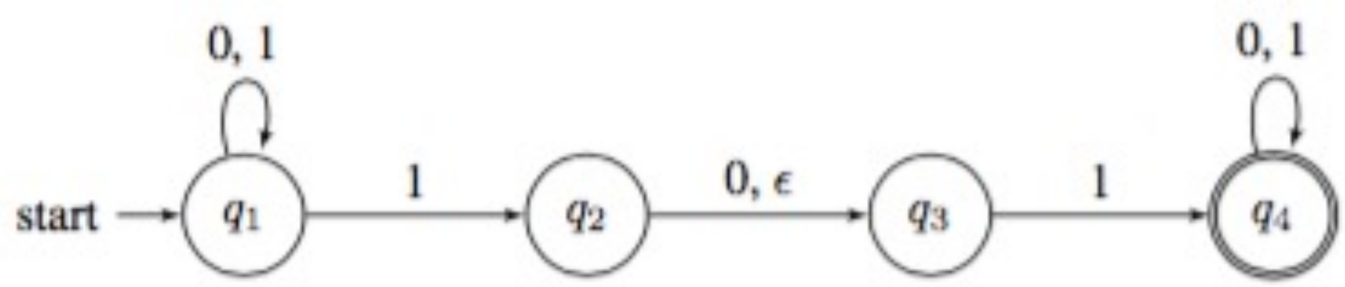
VCU School of Engineering | Computer Science

Definition of NFA

- Formal Definition of NFA:
  - A NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:
    1. Q is a finite set of states
    2. Σ is a finite set called the alphabet
    3. $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function
       - $\Sigma_\varepsilon = \Sigma \cup \varepsilon$
       - $P(Q)$ is the power set of Q (collection of all subsets of Q including the empty string)
    4. $q_0 \in Q$ is the start state
    5. $F \subseteq Q$ is a set of accept states

Definition of NFA, cont.

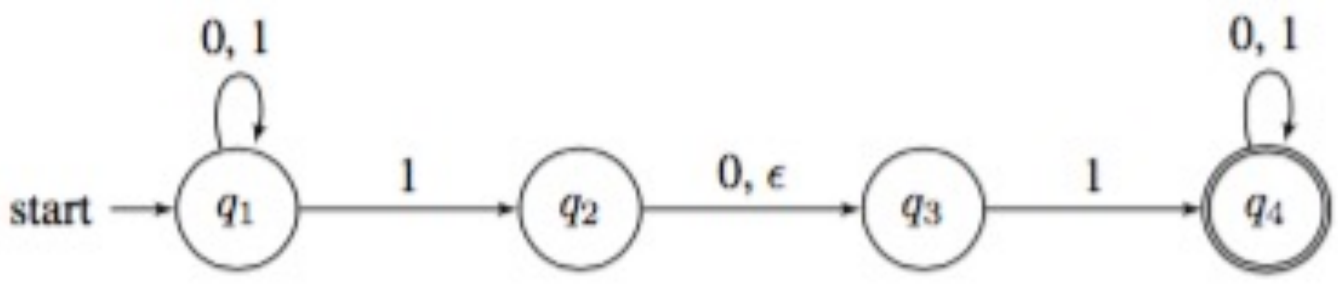
- Ex: N_1:



  - Give the formal definition of this NFA

Definition of NFA, cont.

- Ex: N_1:



  - Give the formal definition of this NFA
  - $Q = \{q_1, q_2, q_3, q_4\}$
  - $\Sigma = \{0, 1\}$
  - $\Sigma_\varepsilon = \{0, 1, \varepsilon\}$
  - $q_0 = q_1$
  - $F = \{q_4\}$

δ	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

School of Engineering | Computer Science

Formal Definition of Computation

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be a non-deterministic finite automata (NFA) and $w \in \Sigma^*$ (a string)
  - N <u>accepts</u> w if we can write $w = y_1 y_2 \ldots y_m$ for $y_i \in \Sigma_\varepsilon$ for all $1 \leq i \leq m$, and there exists a sequence of states $(r_0, r_1, \ldots, r_m) \in Q_{m+1}$ with 3 conditions:
    1. r_0 is the start state q_0 ($r_0 = q_0$)
    2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i \in \{0, \ldots, m\text{-}1\}$
    3. $r_m \in F$ (r_m is an accepting state)
- Do NFA's = DFA's?
  - Yes. For all languages L, there exists a NFA recognizing L if and only if there exists a DFA recognizing L.

Equivalence of NFAs and DFAs

- Theorem 1.39: Every NFA has an equivalent DFA
  - Proof: Let L be a language recognized by a NFA N = $(Q, \Sigma, \delta, q_0, F)$, we construct a DFA M = $(Q', \Sigma, \delta', q_0', F')$ recognizing L
  - Idea: At each point in computation, M keeps track of multiple states
    - $Q' = P(Q)$
    - $(P(Q)$ = set of subsets of $Q)$
    - $|Q'| = 2^{|Q|}$

Equivalence of NFAs and DFAs

- Formal Proof Every NFA has an equivalent DFA
  - Assume for now that the NFA N has no ε transitions:

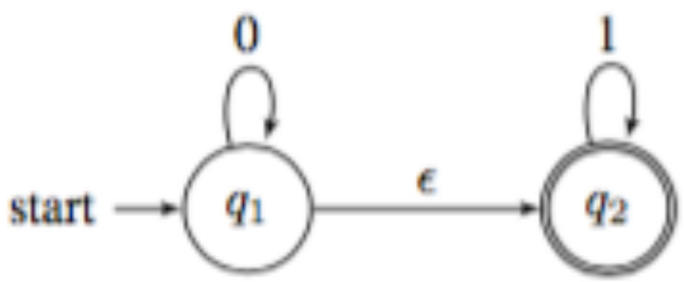
1. $Q' = P(Q)$
2. $\Sigma = \Sigma$
3. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a)$ for some $r \in R\}$
   - R is a state of M, the DFA, and is a set of states of N, the NFA
   - $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$
4. $q_0' = \{q_0\}$
5. $F' = \{R \in Q' \mid R$ contains an accept state of N$\}$

Equivalence of NFA's and DFA's

- <u>Formal Proof</u> Every NFA has an equivalent DFA
  - Now add in ε transitions:
    - For any state R of M, define E(R) = {q | q can be reached from R by traveling ε transitions}
    - Modify δ'(R, a) = {q ∈ E(δ(r, a)) for some r ∈ R}
    - Need to modify q_0' to be E({q_0}) so can handle ε transitions from q_0

Equivalence of NFA's and DFA's

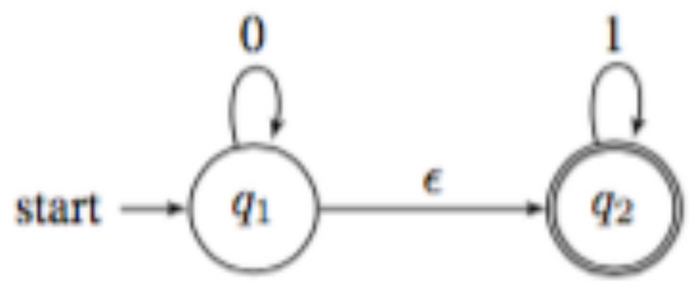
- Ex: NFA → DFA
  - NFA N
    - $Q = \{q_1, q_2\}$
    - $\Sigma = \{0, 1\}$
    - $q_0 = q_1$
    - $F = \{q_2\}$



δ	0	1	ε
$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\{q_2\}$	$\emptyset$	$\{q_2\}$	$\emptyset$

Equivalence of NFA's and DFA's

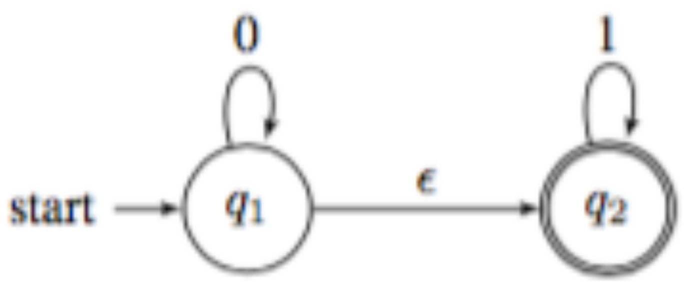
- Ex: NFA → DFA
  - NFA N:
  - 2 states, so $2^Q = 2^2 = 4$
  - DFA M:
    - Q' = power set which includes the empty set and any combination of states from Q
      - $Q' = \{\emptyset, q_1, q_2, q_{12}\}$
    - $\Sigma = \{0, 1\}$ (the alphabet does not change)
    - $q_0' = E(\{q_0\})$
      - q_0' would equal q_1 but there is an ε transition from q_1 to q_2,
      - $q_0' = q_{12}$ since can slide to state q_2 at the start or stay at state q_1
    - F' = any state containing an original accept state
      - $F' = \{q_2, q_{12}\}$



School of Engineering | Computer Science

Equivalence of NFA's and DFA's

- Ex: NFA → DFA
  - NFA N:

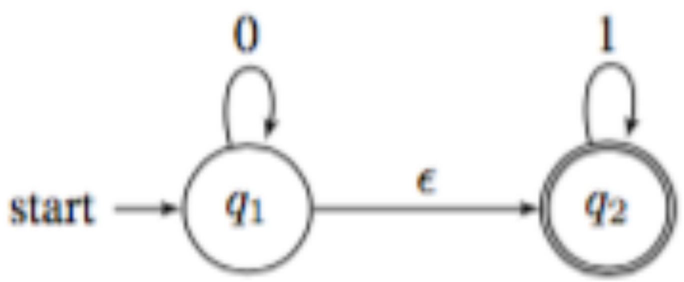


  - DFA M:
    - $Q' = \{\emptyset, q_1, q_2, q_{12}\}$ $\Sigma = \{0, 1\}$ $q_0' = q_{12}$ $F' = \{q_2, q_{12}\}$
    - δ transitions:

School of Engineering | Computer Science

Equivalence of NFA's and DFA's

- Ex: NFA → DFA
  - NFA N:

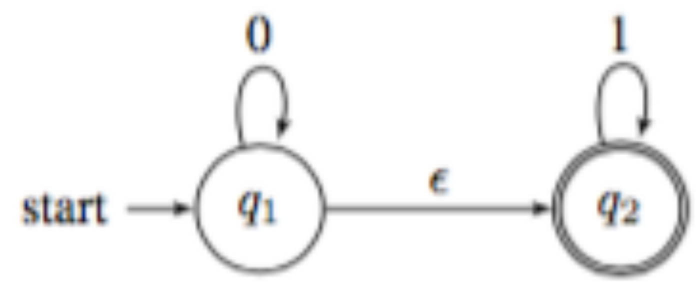


  - DFA M:
    - $Q' = \{\emptyset, q_1, q_2, q_{12}\}$ $\Sigma = \{0, 1\}$ $q_0' = q_{12}$ $F' = \{q_2, q_{12}\}$
    - δ transitions:
      - At state q_1 with a 0 input, can stay at q_1 or slide to q_2 (union of both = q_{12})
      - At state q_1 with a 1 input, there is no transition, so go to $\emptyset$
      - At state q_2 with a 0 input, there is no transition, so go to $\emptyset$
      - At state q_2 with a 1 input, stay at q_2

School of Engineering | Computer Science

Equivalence of NFA's and DFA's

- Ex: NFA → DFA
  - NFA N:

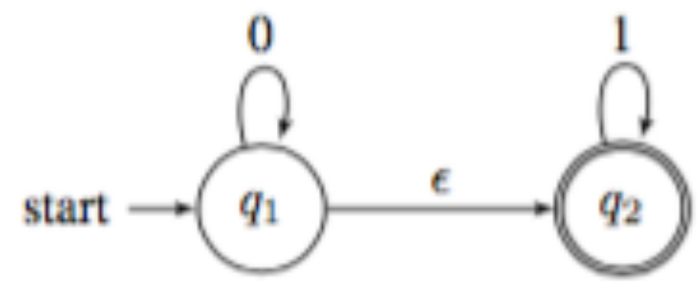


  - DFA M:
    - $Q' = \{\emptyset, q_1, q_2, q_{12}\}$ $\Sigma = \{0, 1\}$ $q_0' = q_{12}$ $F' = \{q_2, q_{12}\}$
    - δ transitions:
      - At state ∅ with an input of 0 or 1 stay at state ∅
      - At state q_{12} (look at state q_1 and state q_2) with an input of 1, q_1 goes to ∅ and q_2 stays at q_2 (union of both = q_2)
      - At state q_{12}, (look at state q_1 and state q_2) with an input of 0, q_1 stays at q_1 or could slide to q_2 and q_2 goes to ∅ (union of both = q_{12})

School of Engineering | Computer Science

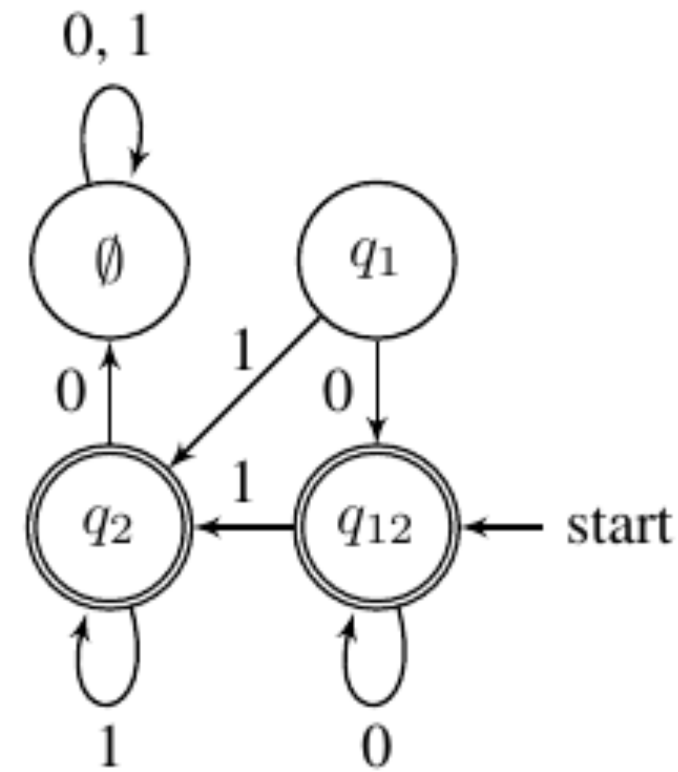
Equivalence of NFA's and DFA's

- Ex: NFA → DFA, cont.
  - NFA N:



  - 2 states, so $2^2 = 4$
  - DFA M
    - $Q' = \{\emptyset, q_1, q_2, q_{12}\}$
    - $\Sigma = \{0, 1\}$
    - $q_0 = q_{12}$
    - $F' = \{q_2, q_{12}\}$

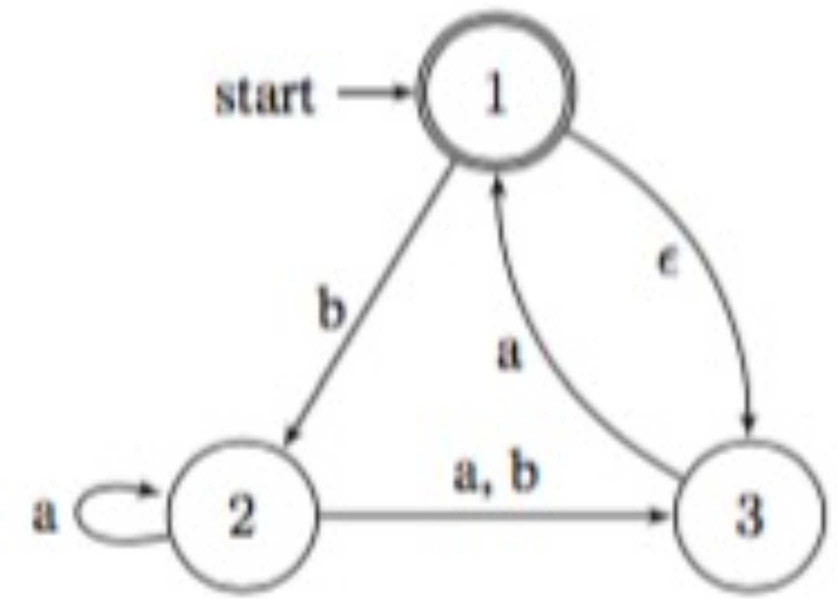
δ'	0	1
q_1	q_{12}	$\emptyset$
q_2	$\emptyset$	q_2
q_{12}	q_{12}	q_2
$\emptyset$	$\emptyset$	$\emptyset$



- Thus, DFA's and NFA's are equivalent in power

Equivalence of NFA's and DFA's

- Ex 2: NFA → DFA
  - NFA N:
    - $Q = \{1, 2, 3\}$
    - $\Sigma = \{a, b\}$
    - $q_0 = 1$
    - $F = \{1\}$

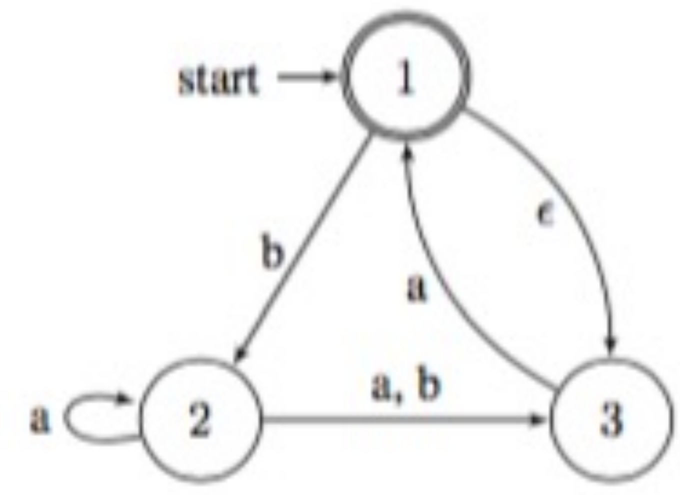


δ	a	b	ε
1	∅	{2}	{3}
2	{2,3}	{3}	∅
3	{1,3}	∅	∅

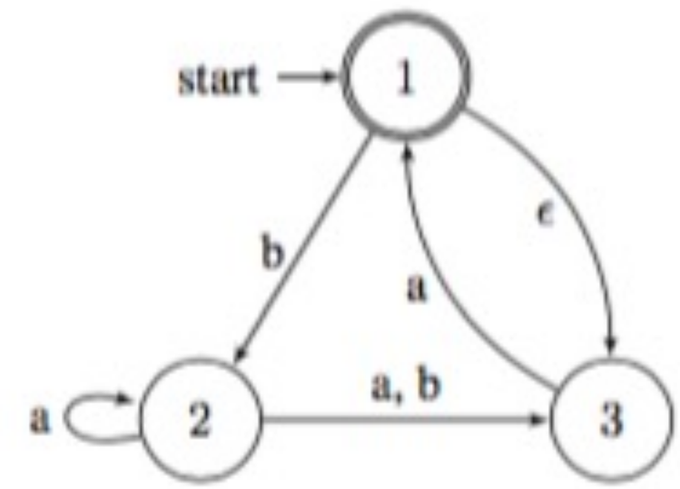
School of Engineering | Computer Science

Equivalence of NFA's and DFA's

- Ex 2: NFA → DFA
  - NFA N:
  - 3 states, so 2^Q =
  - DFA M:
    - Q' =
    - Σ =
    - q_0' =
    - F' =

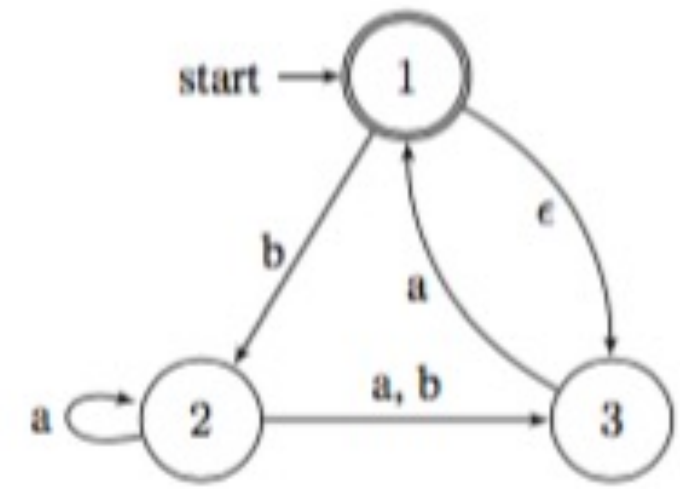


Equivalence of NFA's and DFA's



- Ex 2: NFA → DFA
  - NFA N:
  - 3 states, so $2^Q = 2^3 = 8$
  - DFA M:
    - Q′ = power set which includes the empty set and any combination of states from Q
      - Q′ = {∅, 1, 2, 3, 12, 13, 23, 123}
    - Σ = {a, b} (the alphabet does not change)
    - $q_0{}' = E(\{q_0\})$
      - $q_0{}'$ would equal 1 but there is an ε transition from 1 to 3,
      - $q_0{}'$ = 13 since can slide to state 3 at the start or stay at state 1
    - F′ = any state containing an original accept state
      - F′ = {1, 12, 13, 123}

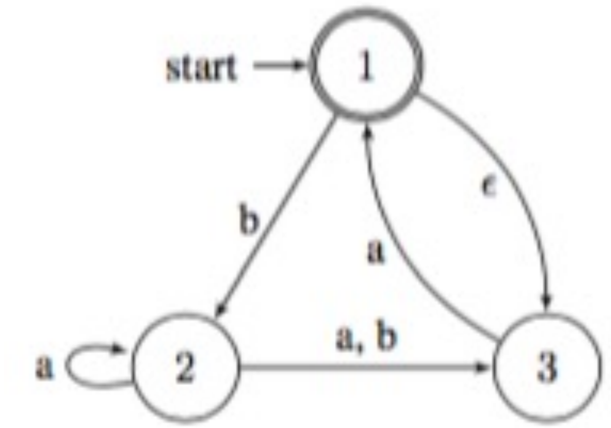
Equivalence of NFA's and DFA's



- Ex 2: NFA → DFA
  - DFA M δ transitions: {Ø, 1, 2, 3, 12, 13, 23, 123}

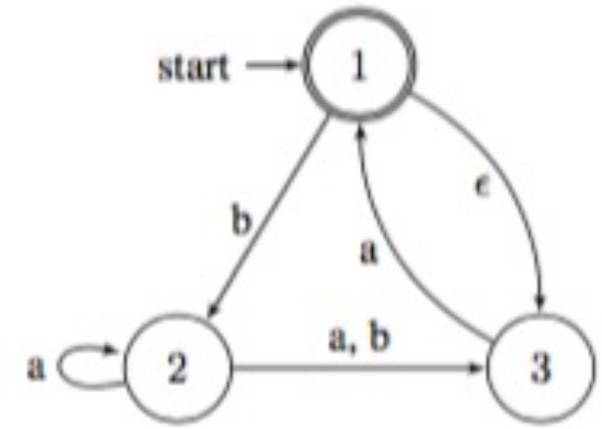
School of Engineering | Computer Science

Equivalence of NFA's and DFA's



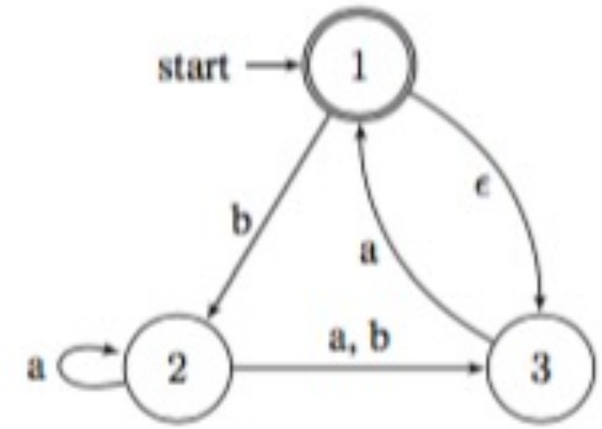
- Ex 2: NFA → DFA
  - DFA M δ transitions: {Ø, 1, 2, 3, 12, 13, 23, 123}
    - At state 1 with an input of a, there is no transition, go to Ø
    - At state 1 with an input of b, move to state 2
    - At state 2 with an input of a, stay at state 2 or move to state 3 (union of both = 23)
    - At state 2 with an input of b, move to state 3
    - At state 3 with an input of a, move to state 1 or slide back to state 3 (union of both = 13)
    - At state 3 with an input of b, there is no transition, go to Ø
    - At state Ø with an input of a or b stay at state Ø

Equivalence of NFA's and DFA's



- Ex 2: NFA → DFA
  - DFA M δ transitions: {∅, 1, 2, 3, 12, 13, 23, 123}
    - At state 12 (look at state 1 and state 2) with an input of a, 1 goes to ∅ and 2 moves to 23 (union of both = 23)
    - At state 12 (look at state 1 and state 2) with an input of b, 1 goes to 2 and 2 moves to 3 (union of both = 23)
    - At state 13 (look at state 1 and state 3) with an input of a, 1 goes to ∅ and 3 moves to 1 or slides back to 3 (union of both = 13)
    - At state 13 (look at state 1 and state 3) with an input of b, 1 goes to 2 and 3 goes to ∅ (union of both = 2)

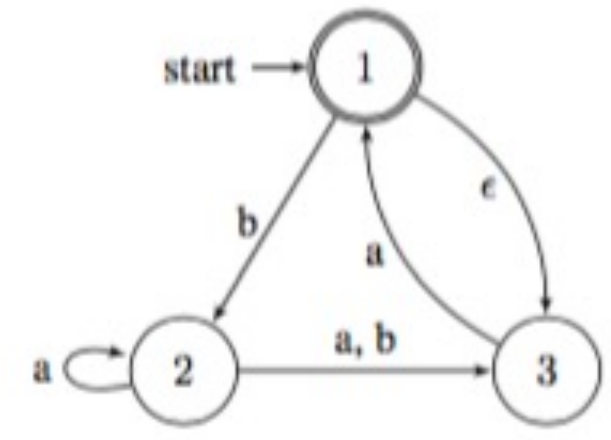
Equivalence of NFA's and DFA's



- Ex 2: NFA → DFA
  - DFA M δ transitions: {Ø, 1, 2, 3, 12, 13, 23, 123}
    - At state 23 (look at state 2 and state 3) with an input of a, 2 goes to 23 and 3 moves to 1 or slides back to 3 (union of both = 123)
    - At state 23 (look at state 2 and state 3) with an input of b, 2 goes to 3 and 3 goes to Ø (union of both = 3)
    - At state 123 (look at state 1, state 2 and state 3) with an input of a, 1 goes to Ø, 2 goes to 23 and 3 moves to 1 or slides back to 3 (union of all = 123)
    - At state 123 (look at state 1, state 2 and state 3) with an input of b, 1 goes to 2, 2 goes to 3 and 3 goes to Ø (union of all = 23)

Equivalence of NFA's and DFA's

- Ex 2: NFA → DFA, cont.
  - NFA N:
  - 2 states, so $2^3 = 8$
  - DFA M:
    - $Q' = \{\emptyset, 1, 2, 3, 12, 13, 23, 123\}$
    - $\Sigma = \{a, b\}$
    - $q_0 = 13$
    - $F' = \{1, 12, 13, 123\}$

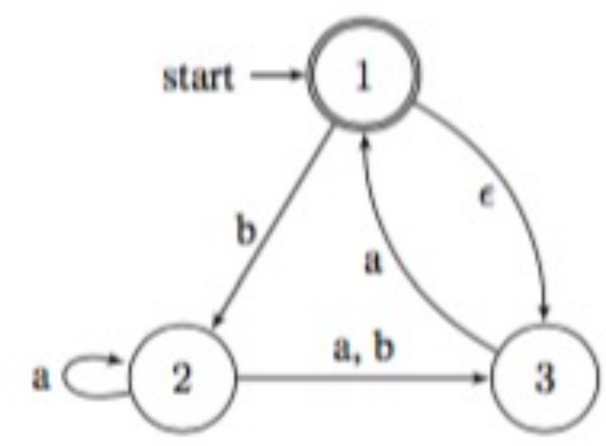


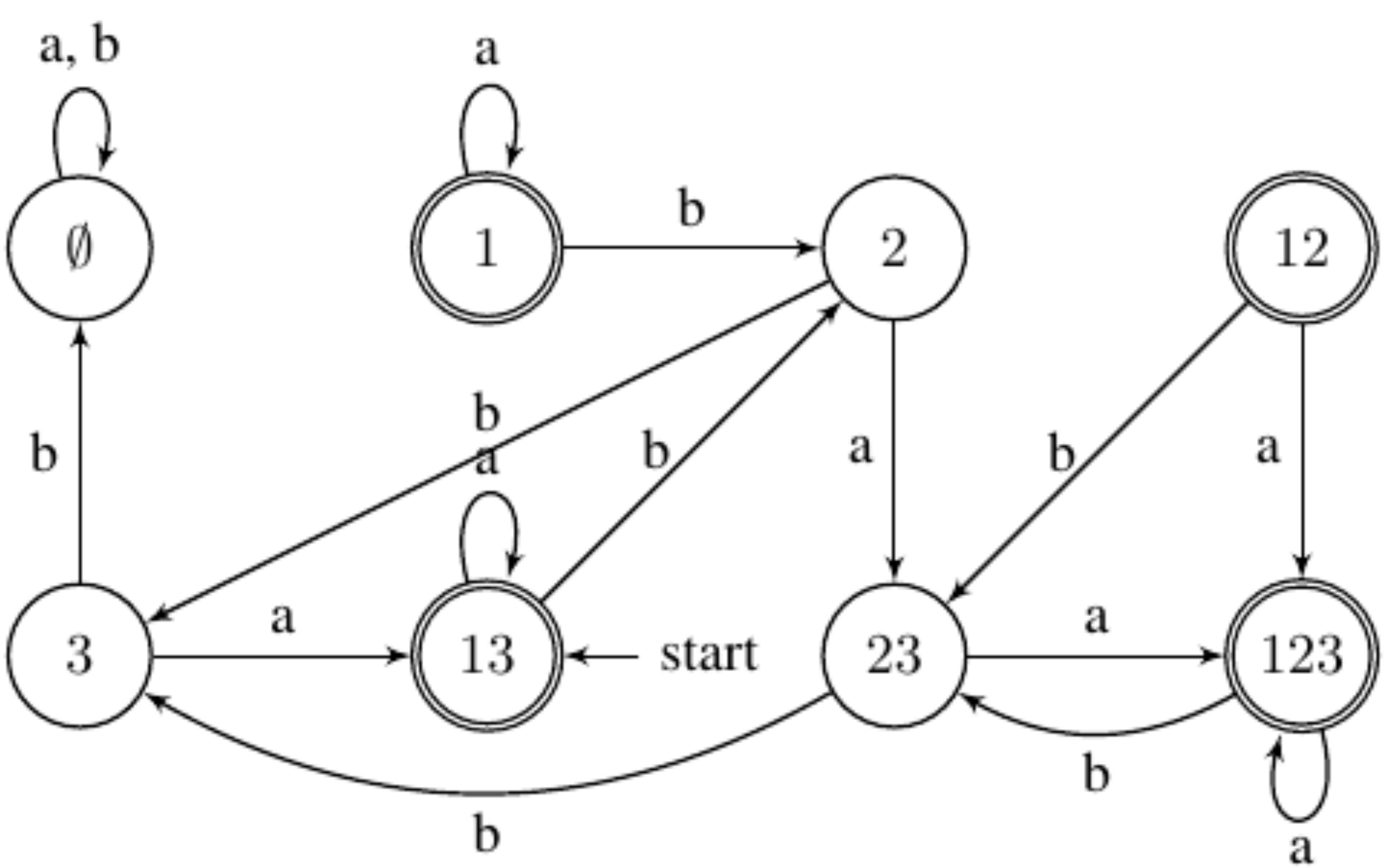
δ'	a	b
1	∅	2
2	23	3
3	13	∅
12	23	23
13	13	2
23	123	3
123	123	23
∅	∅	∅

School of Engineering | Computer Science

Equivalence of NFA's and DFA's

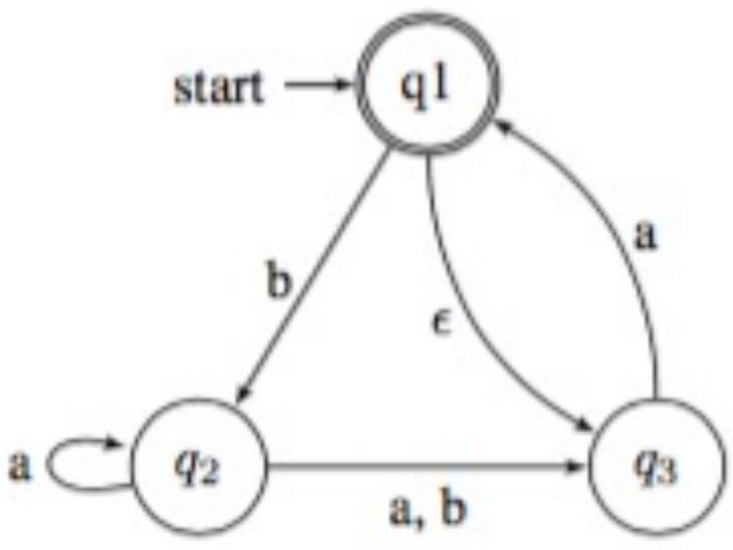
- Ex 2: NFA → DFA, cont.
  - NFA N:
  - DFA M:





Try It

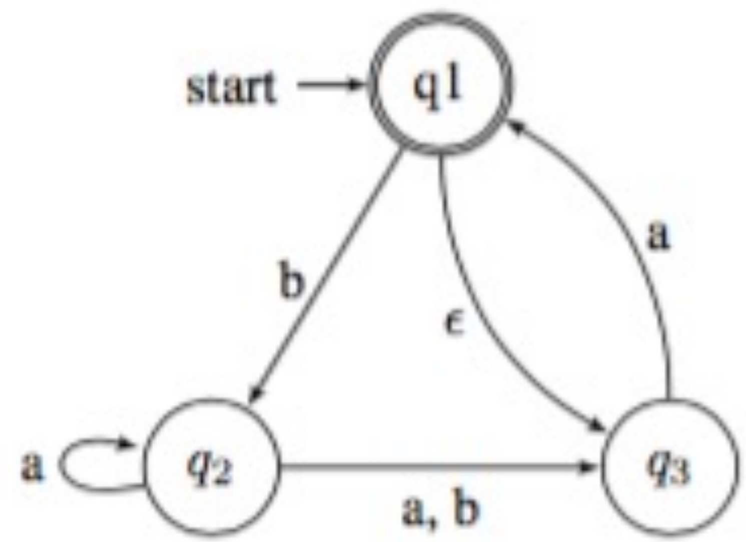
- Formally describe this NFA:



- What are some strings it will accept?

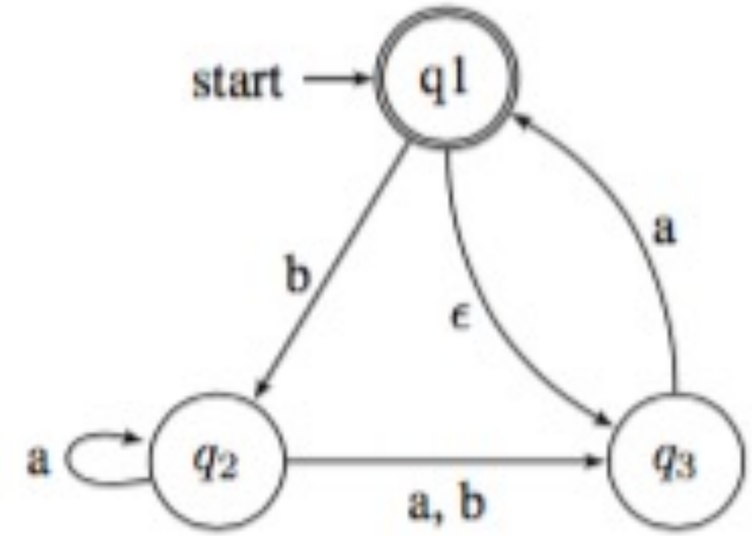
Try It

- Formally describe this NFA:
  - $Q = \{q_1, q_2, q_3\}$
  - $\Sigma = \{a, b\}$
  - $\Sigma_\varepsilon = \{a, b, \varepsilon\}$
  - $q_0 = q_1$
  - $F = \{q_1\}$



δ	a	b	ε
q_1	$\emptyset$	$\{q_2\}$	$\{q_3\}$
q_2	$\{q_2, q_3\}$	$\{q_3\}$	$\emptyset$
q_3	$\{q_1\}$	$\emptyset$	$\emptyset$

Try It



- What are some strings it will accept?
  - ε, a, bba, baa
  - Officially it accepts: $((ba^*(a \cup b) \cup \varepsilon)(a^*a)) \cup \varepsilon$