Equivalence of DFAs and NFAs

Equivalence

- What does it mean for two automata to be equivalent?
  - Two finite automata M_1 and M_2 are equivalent if $L(M_1) = L(M_2)$.
  - If they accept the same language.
- NFA and DFA are equivalent if every language that can be accepted by an NFA can also be accepted by a DFA and vice versa

DFA / NFA Equivalence

- How we will show this:
  1. Given a DFA that accepts arbitrary language L_1, create an NFA that also accepts L_1
  2. Given an NFA that accepts arbitrary language L_2, create a DFA that also accepts L_2

- By proving 1, we show that the class of languages that DFAs represent is a subset of the class of languages that NFAs represent
- By proving 2, we show that the class of languages that NFAs represent is a subset of the class of languages that DFAs represent
- Put them together, and that shows that DFAs and NFAs represent the same class of languages!

Step 1: Given DFA find NFA

- Observe that a DFA can easily be converted to an equivalent NFA:
  - DFAs – all transitions lead to exactly one state
  - Define the transitions of the NFA to consist of sets of only 1 element. All ϵ–transitions are to $\emptyset$.
    - (When we think of state diagrams, the DFA state diagram is already an NFA state diagram)
    - (Formally, however, we have to complete the definition of the NFA by tweaking the transition function so that the output are sets, and the function is fully defined for all inputs (including ϵ–transitions).

Step 2: Given NFA find DFA

- Given NFA find equivalent DFA
  - Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA:
  - We need to show that there exists a DFA
    - $M = (Q', \Sigma, \delta', q_0', F')$
  - Such that $L(N) = L(M)$
  - For now we'll assume that N has no ϵ-transitions.

NFA -> DFA

- Basic idea
  - Recall that for an NFA, $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$
    - In other words, the transition function moves from one state to some set of states (0 or more).
    - P(Q) is the power set of Q – the set of all possible subsets of Q. The set of states we move to is some member of this power set.

NFA -> DFA

- Recall: we used the Cartesian product to track two machines simultaneously and build a machine that represents the union of two machines
  - We use a similar idea here to represent what the NFA is doing
  - Except the set of states for our new machine is not the set of all possible ordered pairs (as in Cartesian product)
  - Instead it's the set of all possible subsets of Q (i.e. the power set P(Q)). At every step, we'll be in some collection of states (some element of P(Q)), we'll read a symbol, and we'll move to some other set of states (some element of P(Q))

NFA -> DFA

- The same idea said slightly differently
  - Since the NFA can be in a set of states at any point, construct the DFA so that its states correspond to all the possible sets of states that the NFA could be in.

  - For Cartesian product, if we had two machines with $|Q_1|$ and $|Q_2|$ states, the union machine had
    - $|Q_1| * |Q_2|$ states

  - If the NFA has $|Q|$ states, how many will the constructed DFA have?
    - $|P(Q)| = 2^{|Q|}$

NFA -> DFA

but we're not worrying about ε yet

- Basic idea more formally
  - Recall that for an NFA, $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$
  - Use the states of M to represent subsets of Q.
  - If there is one state of M for every subset of Q, then the non-determinism of N can be eliminated.
  - This technique, called <u>subset construction</u>, is a primary means for removing non-determinism from an NFA.

NFA -> DFA

- Formal definition
  - Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA
  - We define a DFA, $M = (Q', \Sigma, \delta', q_0', F')$
    - $Q' = P(Q)$
    - $q_0' = \{q_0\}$
    - For $R \in Q'$ and $a \in \Sigma$,
    - $$\delta'(R,a) = \bigcup_{r \in R} \delta(r,a)$$
    - $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$

Notational abuse – although written as a set, this is a single state in the DFA

We use R to make clear that the individual states of M are themselves subsets of Q

NFA -> DFA

- Algorithm for building M (NFA-to-DFA)

1. Add $\{q_0\}$ to Q' -- Make it the initial state of M and mark it as unfinished

2. Repeat until all states of M are marked as finished
   1. Take any unfinished state V from M (i.e. $V \in Q'$) that has no outgoing edge for some symbol a.
   2. For all states q in V (recall $V \subseteq Q$) determine the set of states W that can be reached by following the transition from q on input a.

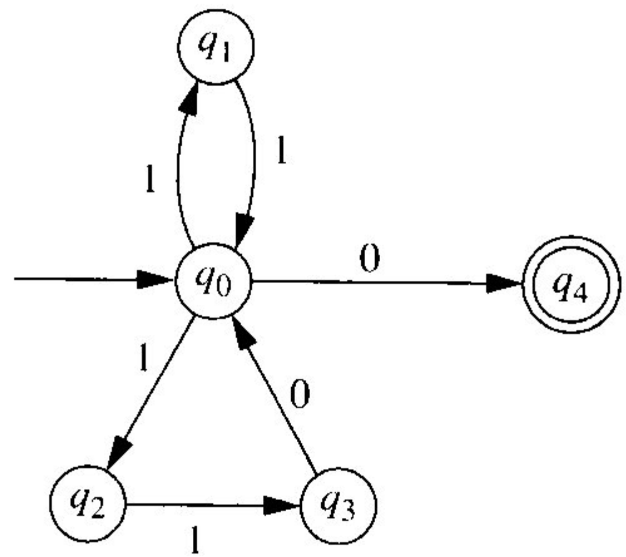
   $$W = \bigcup_{q \in V} \delta(q,a)$$

   3. Add W to states of M (add W to Q') if not already there.
   4. Add transition in M from V to W on input a.
   5. Mark V as finished if it has a transition arrow out for each symbol.

3. Mark every state in M that contains a final state from N as a final state in M.

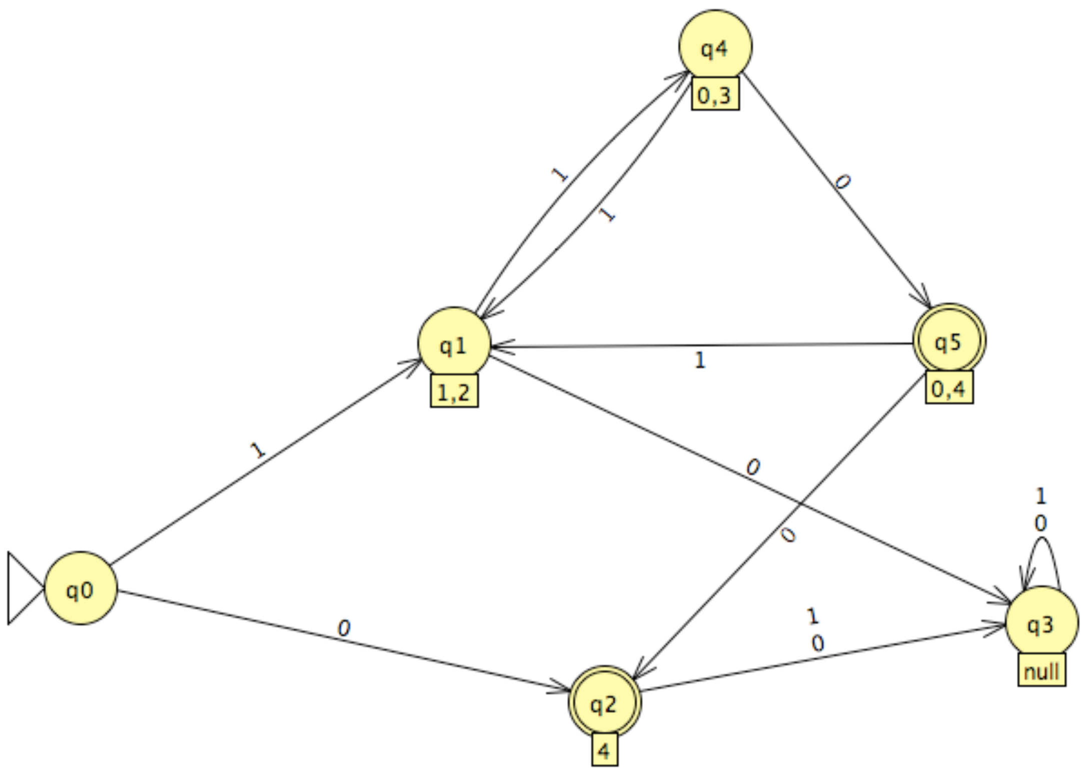
NFA -> DFA

- Example



(b)

NFA -> DFA



NFAs with ϵ–transitions

- What if the NFA has ϵ–transitions?
- Recall:
  - States in DFA correspond to set of states reachable in NFA on given input.
  - Must consider states that you can get to via ϵ–transitions.

ϵ–Closure

- Define
  - For a set of states R
  - E(R) = all states that can be reached from R by traveling along 0 or more ϵ–transitions.

  - Is $E(R) \supseteq R$?
    - Yes

NFA -> DFA (Take 2)

- Formal definition
  - Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA
  - We define a DFA, $M = (Q', \Sigma, \delta', q_0', F')$
    - $Q' = P(Q)$
    - $q_0' = E(\{q_0\})$
    - $F' = \{R \in Q' \mid R \cap F \neq \varnothing\}$

NFA -> DFA (Take 2)

- Computing δ'
  - δ' (S, a) for $S \in Q'$, $a \in \Sigma$
    - Let $S = \{ p_1, p_2, \ldots, p_n \}$
    - Compute the set of all states reachable from states in S on input a using transitions from N.

$$R = \{r_1, r_2, \cdots, r_m\} = \bigcup_{i=1}^{n} \delta\ (p_i, a)$$

    - δ' (S, a) will be the ϵ-closure of the set of states *R*

$$\delta'(S, a) = E(R)$$

NFA -> DFA

- Algorithm for building M (NFA-to-DFA modified)

1. Add $E(\{q_0\})$ to Q' -- Make it the initial state of M and mark it as unfinished

2. Repeat until all states of M are marked as finished
    1. Take any unfinished state V from M (i.e. V ∈ Q') that has no outgoing edge for some symbol a.
    2. For all states q in V (recall V ⊆ Q) determine the set of states W that can be reached by following the transition from q on input a, *followed by any number of ϵ-transitions.*

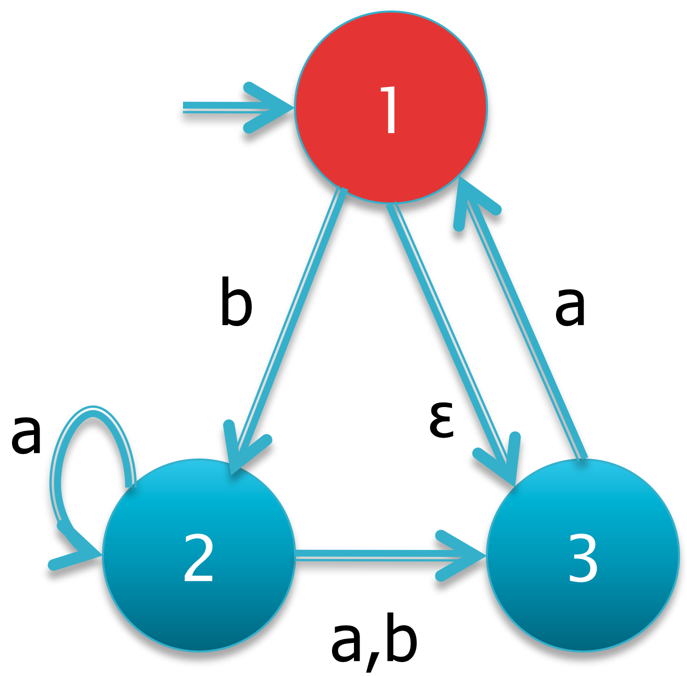
    $$W = E\left(\bigcup_{q \in V} \delta(q,a)\right)$$

    3. Add W to states of M (add W to Q') if not already there.
    4. Add transition in M from V to W on input a.
    5. Mark V as finished if it has a transition arrow out for each symbol.

3. Mark every state in M that contains a final state from N as a final state in M.

Example

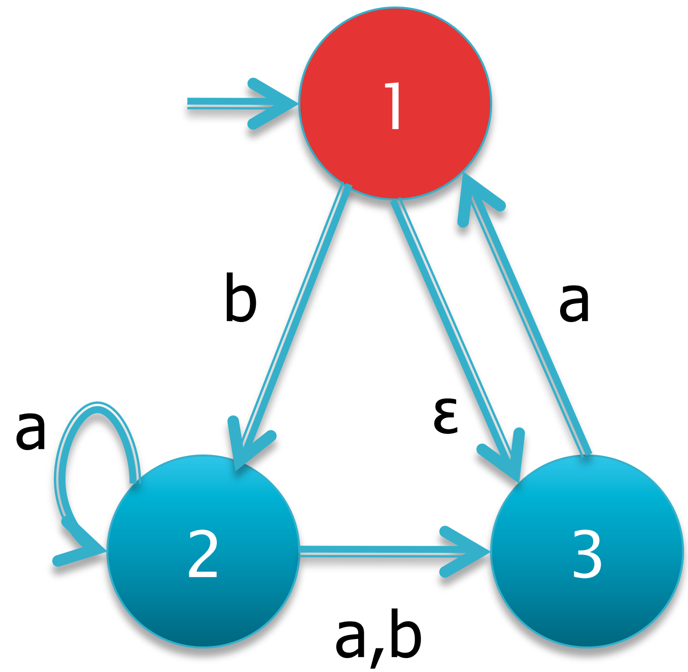
- Example 1.41 in Sipser



(red indicates accepting states)

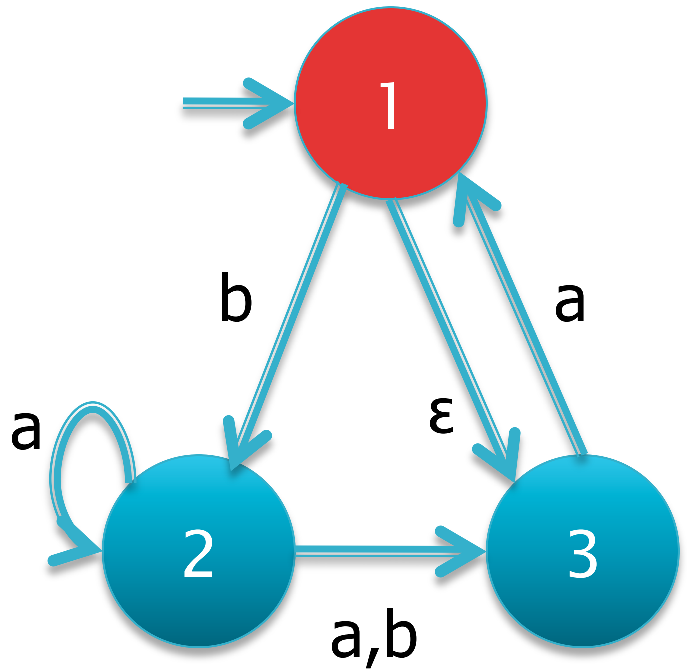
Example

- The NFA
  - $N = \{Q, \Sigma, \delta, q_0, F\}$
  - $Q = \{1, 2, 3\}$
  - $\Sigma = \{a, b\}$
  - $q_0 = 1$
  - $F = \{1\}$
  - $\delta(1,a) = \varnothing$ $\delta(1, b) = \{2\}$ $\delta(1, \epsilon) = \{3\}$
  - $\delta(2,a) = \{2,3\}$ $\delta(2, b) = \{3\}$ $\delta(2, \epsilon) = \varnothing$
  - $\delta(3,a) = \{1\}$ $\delta(3, b) = \varnothing$ $\delta(3, \epsilon) = \varnothing$



ϵ–Closure

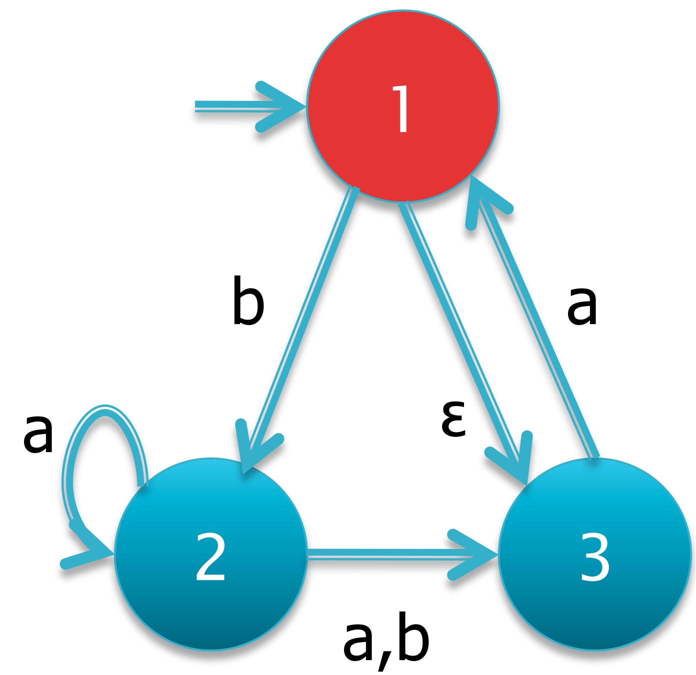
- E ({1}) = { 1, 3 }
- E ({2}) = { 2 }
- E ({3}) = { 3 }



Example

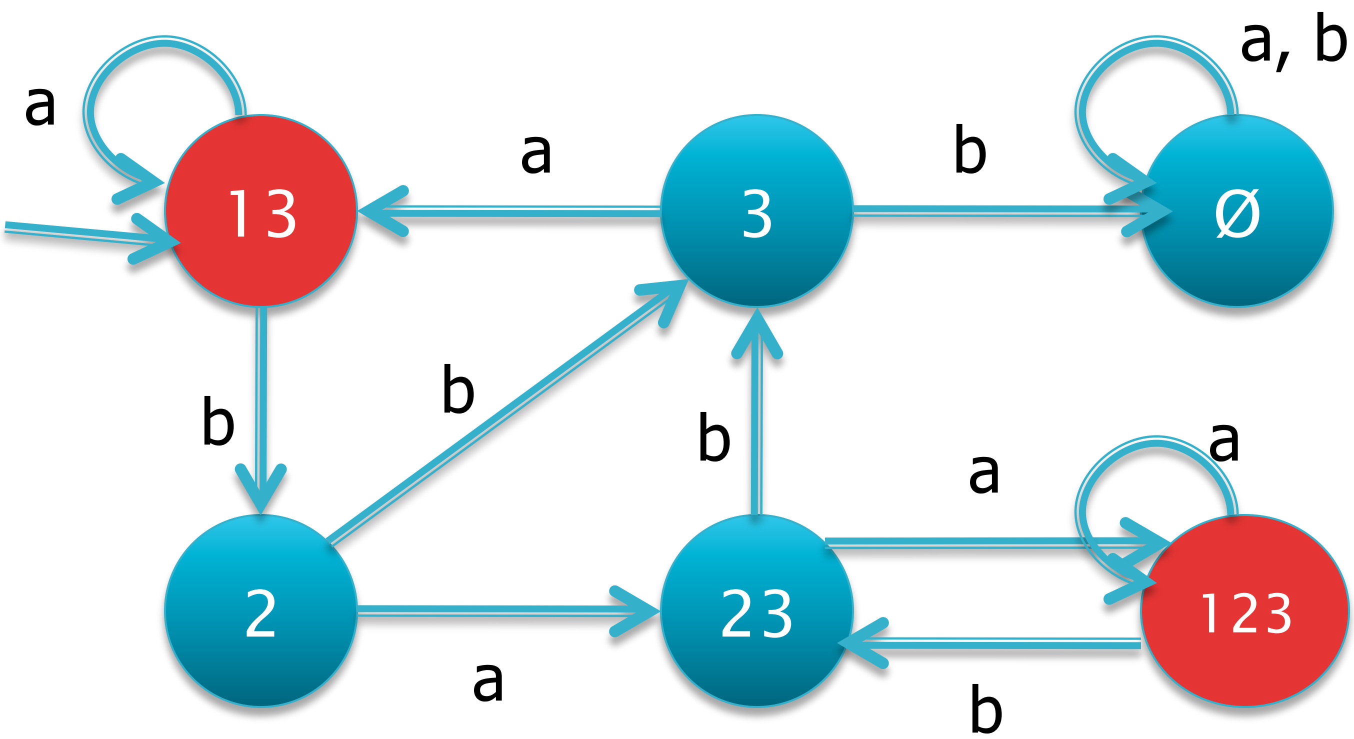
- The DFA, M
  - $M = (Q', \Sigma, \delta', q_0', F')$

  - $Q' = \{ \emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
  - $\Sigma = \{a, b\}$
  - $q_0' = \{1, 3\}$
  - $F = \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$

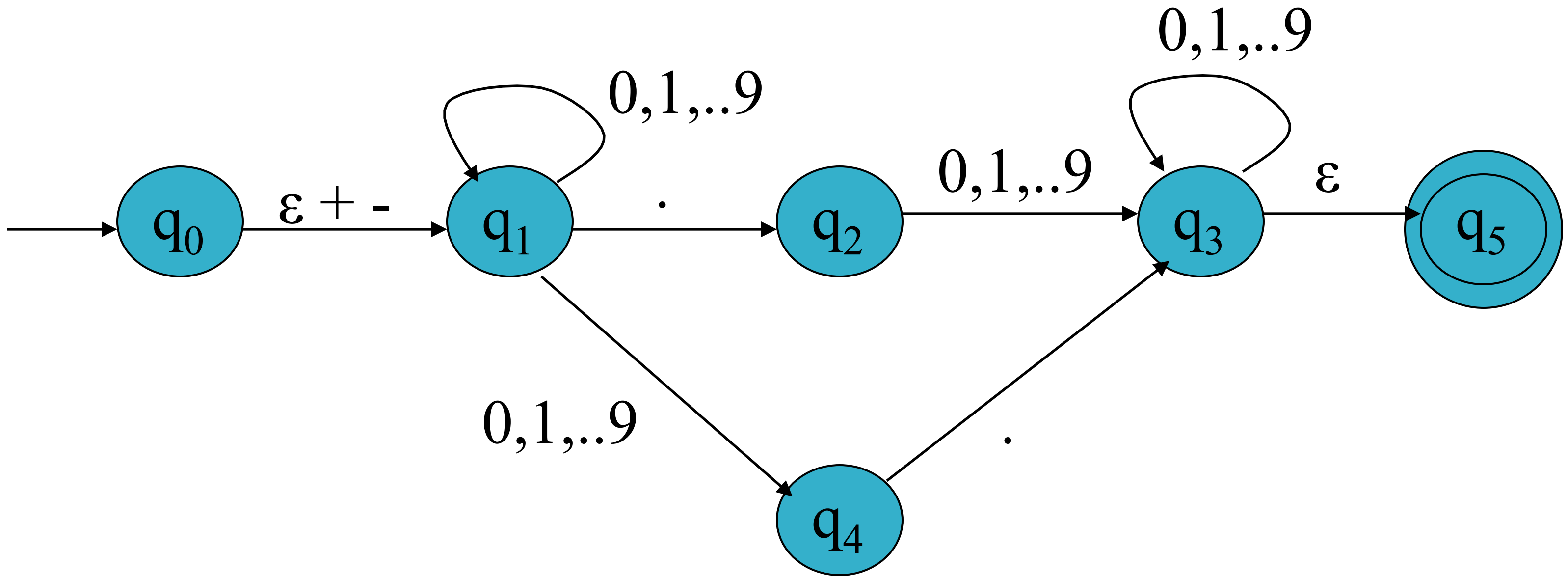


Example

- The DFA



NFA -> DFA

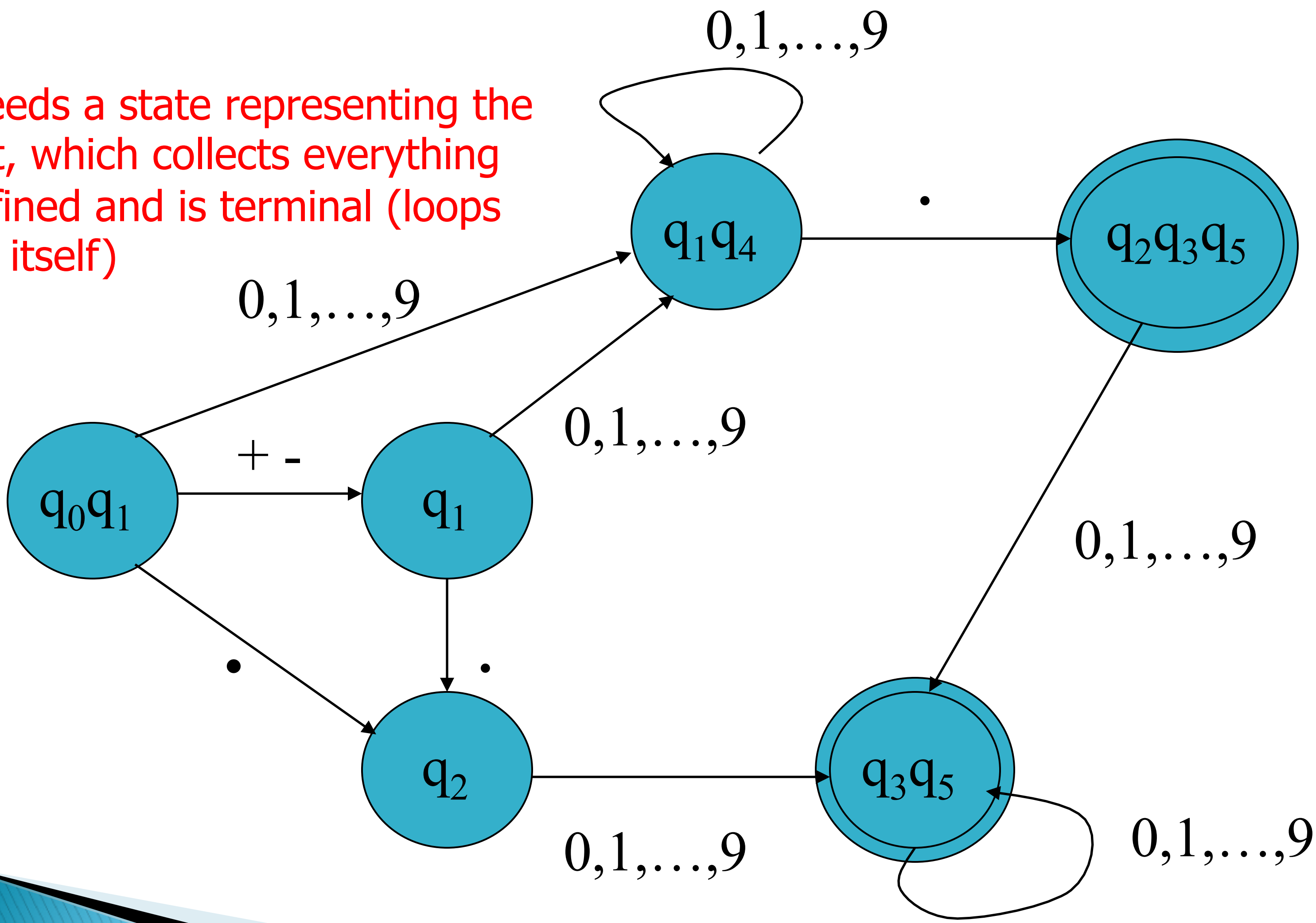


NFA -> DFA

State	ε closure
q_0	$\{q_0, q_1\}$
q_1	$\{q_1\}$
q_2	$\{q_2\}$
q_3	$\{q_3, q_5\}$
q_4	$\{q_4\}$
q_5	$\{q_5\}$

NFA -> DFA

This needs a state representing the null set, which collects everything not defined and is terminal (loops only to itself)

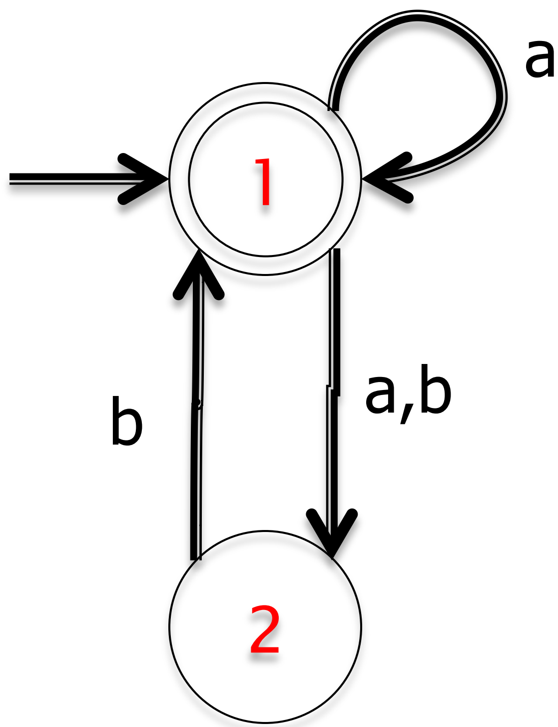


What We Have Shown

- For every DFA, there is an NFA that accepts the same language
- For every NFA, there is a DFA that accepts the same language

- DFAs, NFAs are equivalent!

Practice Problem (Sipser 1.16a)



Construct an equivalent DFA for the given NFA

Practice Problem (Sipser 1.17)

- Give an NFA recognizing the language over {0,1} given by
  - $L(N) = \{01, 001, 010\}^*$
- Convert this NFA to an equivalent DFA. Include only reachable states.