finite automata

chapter 2

preliminaries

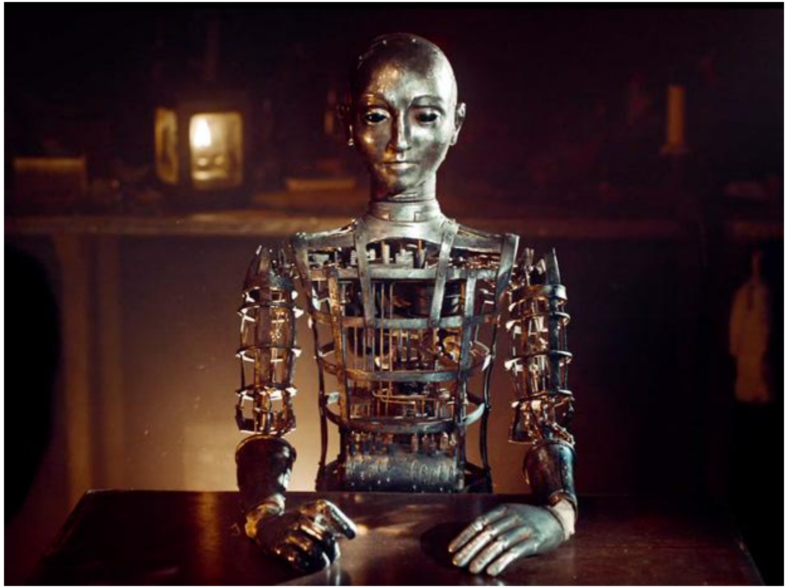
- What are the fundamental capabilities and limitations of computers?

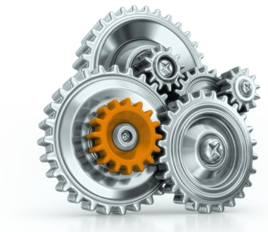
preliminaries

- What makes some problems computationally hard and others easy?

preliminaries

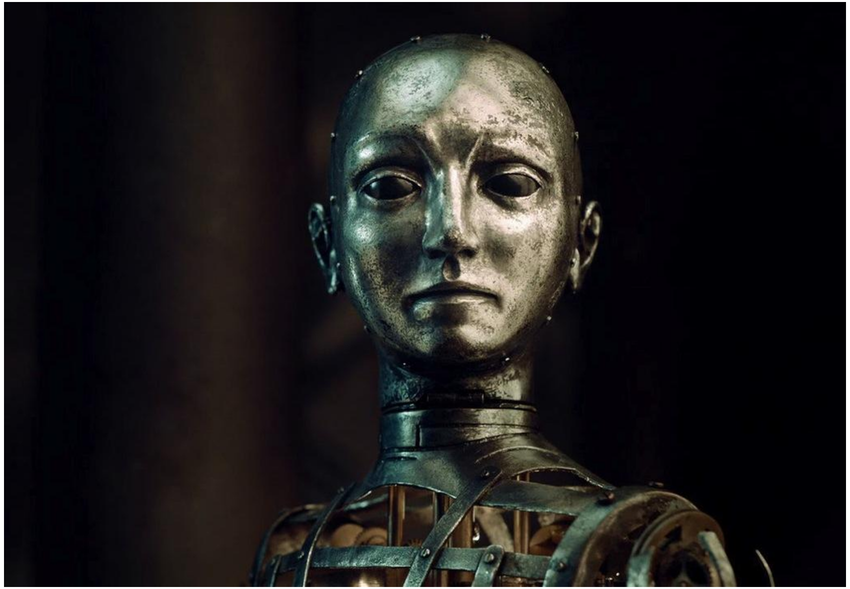
- What kinds of problems can the computer solve? Which problems can the computer not solve?





automata theory

- deals with definitions and properties of mathematical models of computation

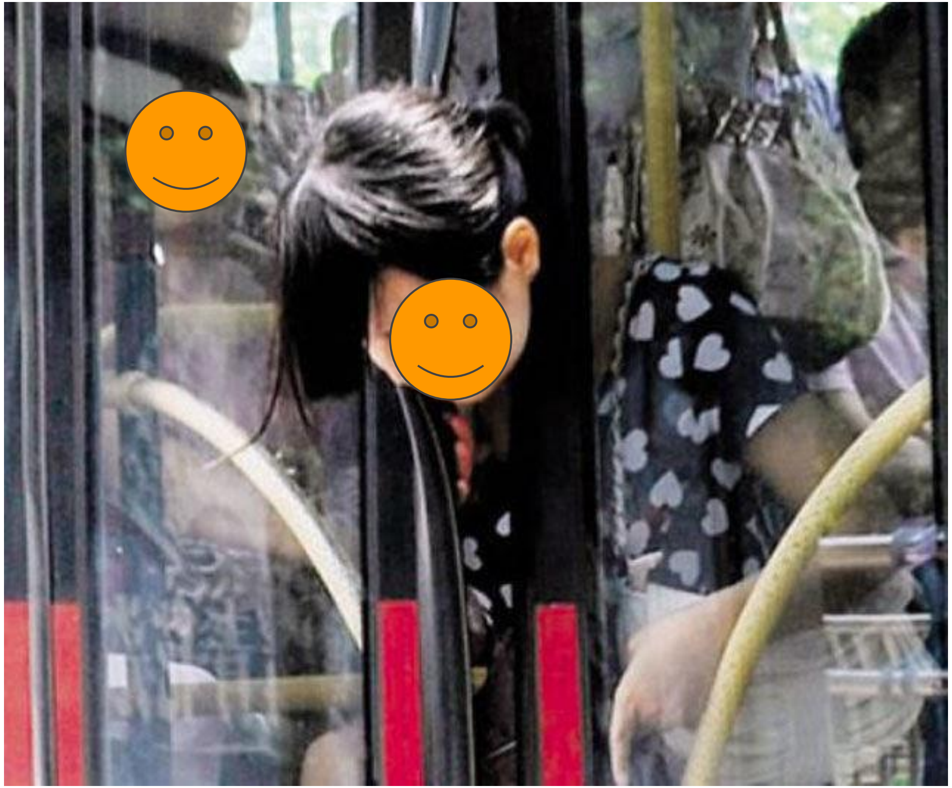


automata theory

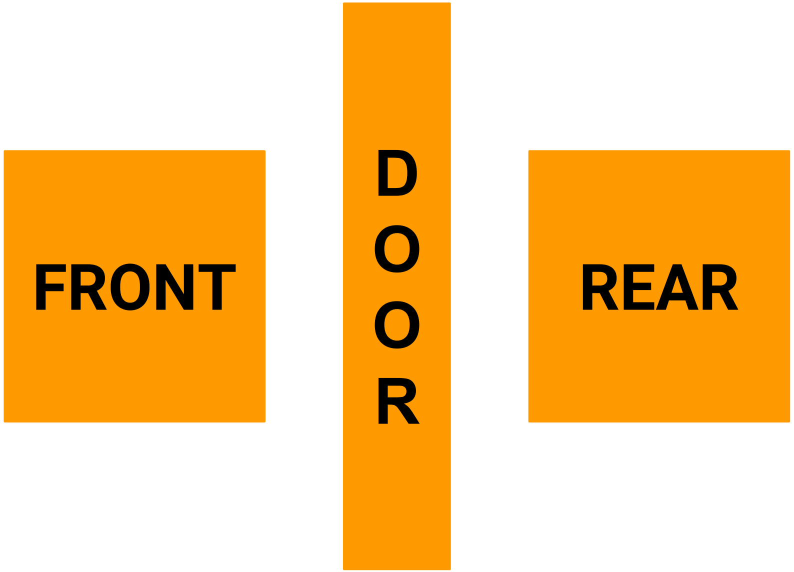
- what is a computer?
  - computational model

finite automata - the case of the automatic doors

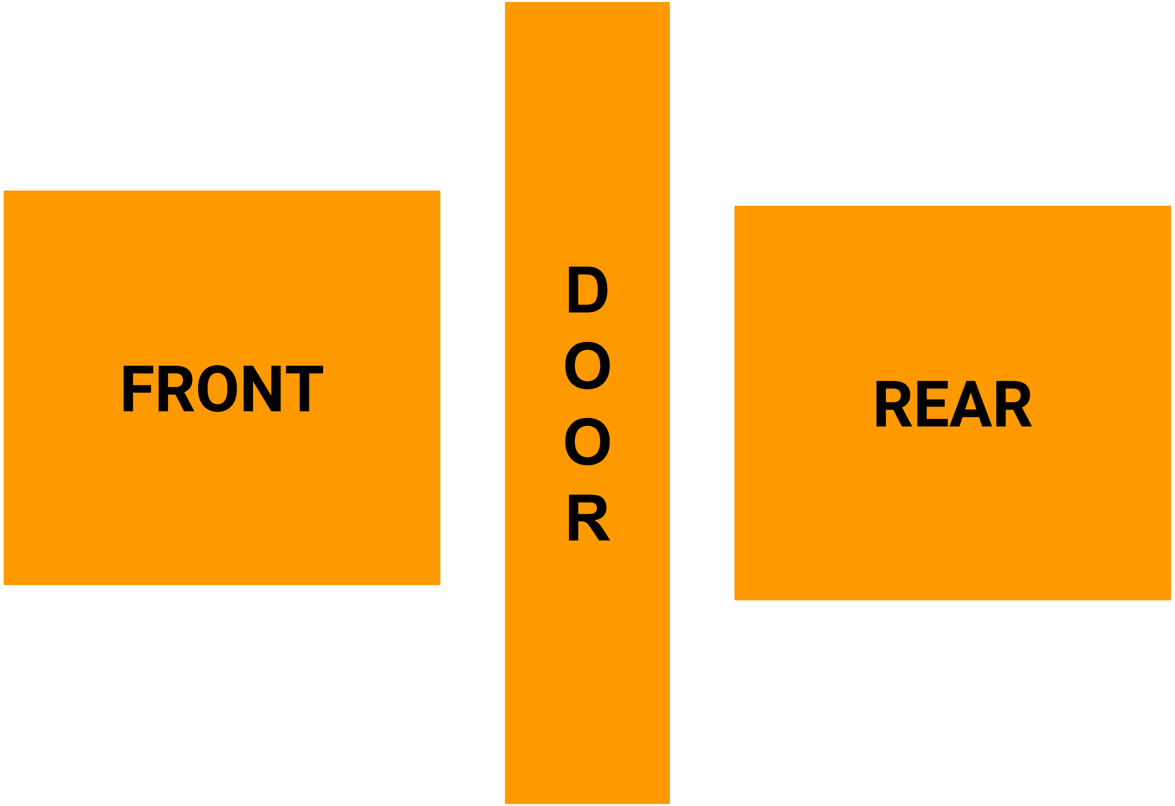
finite automata - the case of the automatic doors



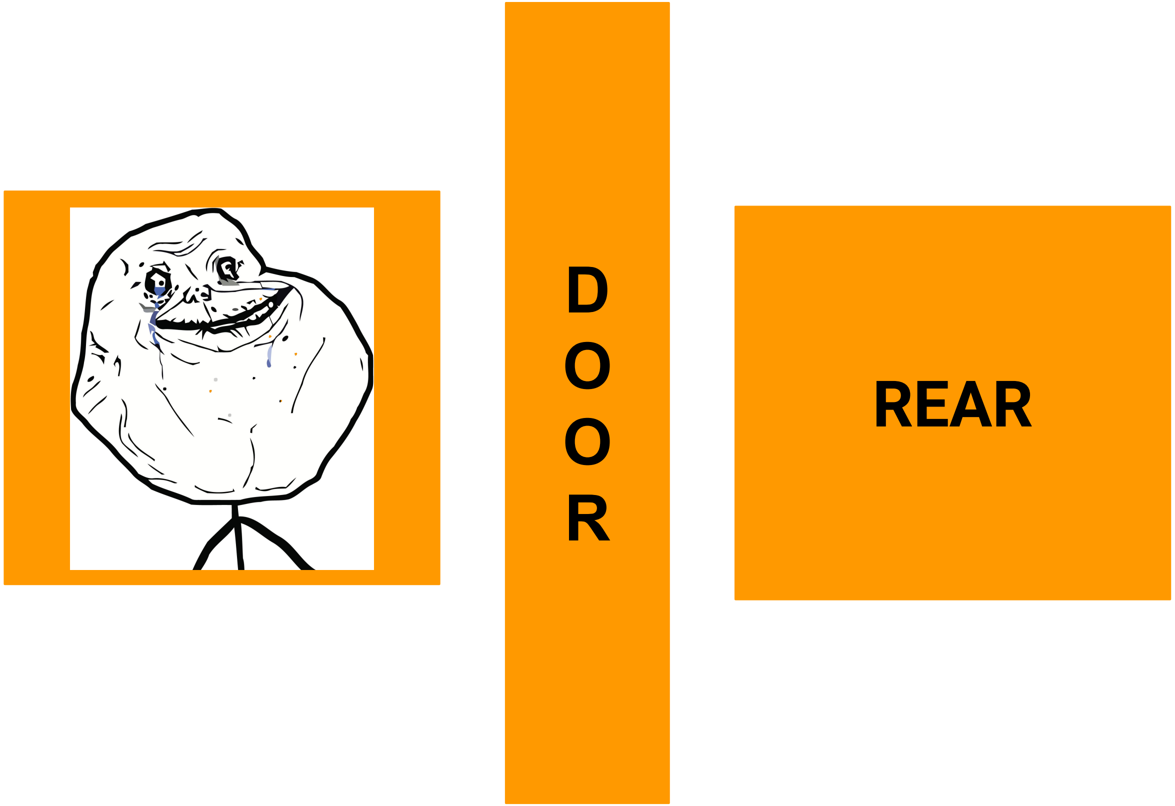
finite automata - the case of the automatic doors



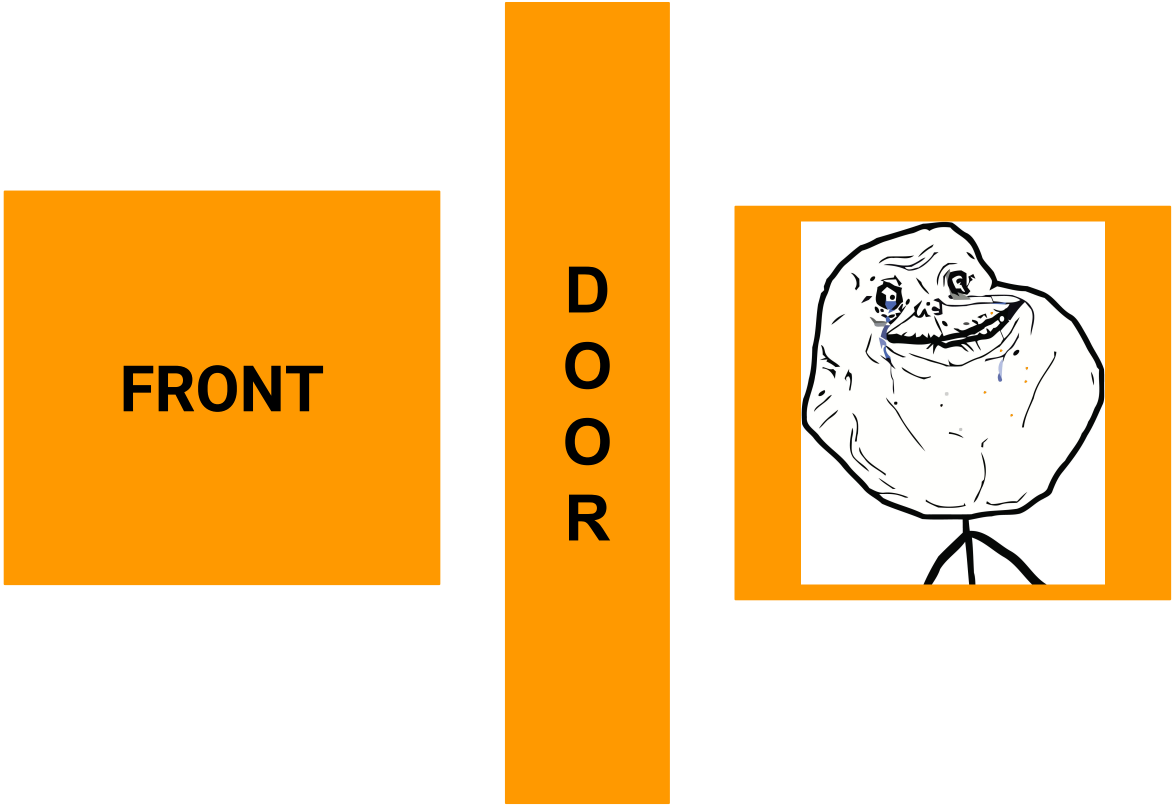
finite automata - the case of the automatic doors



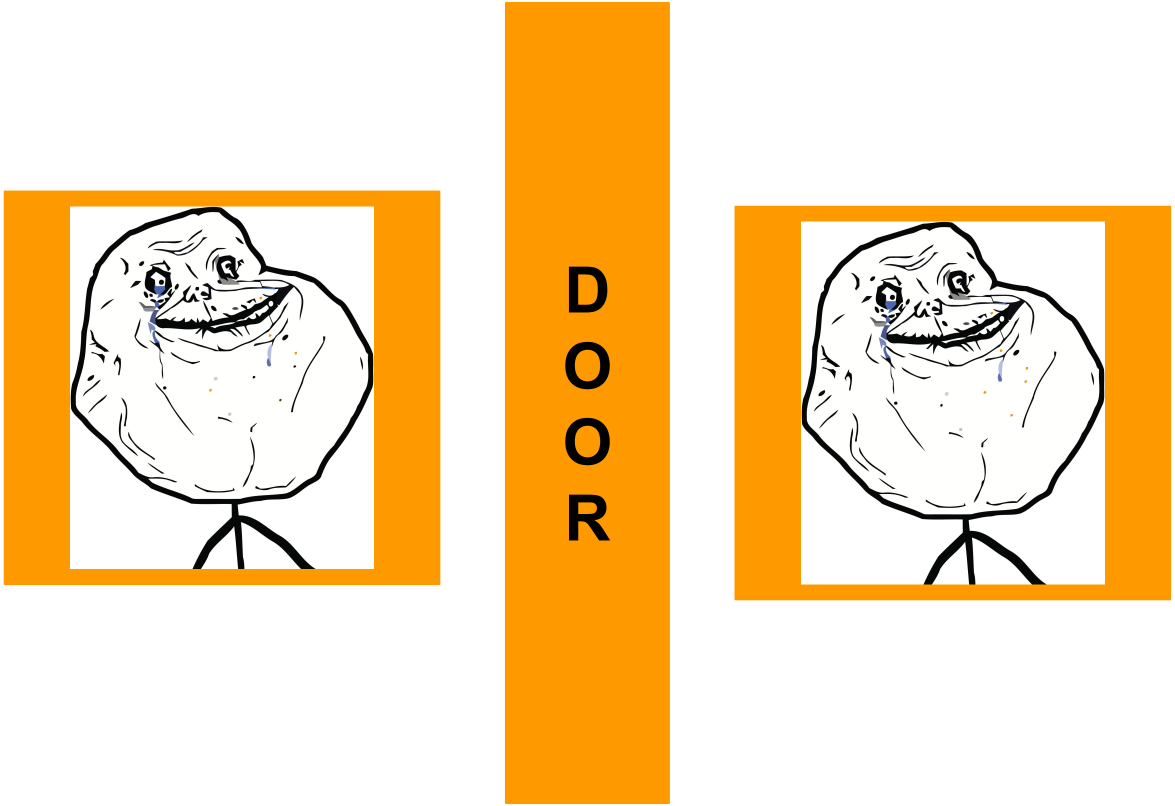
finite automata - the case of the automatic doors



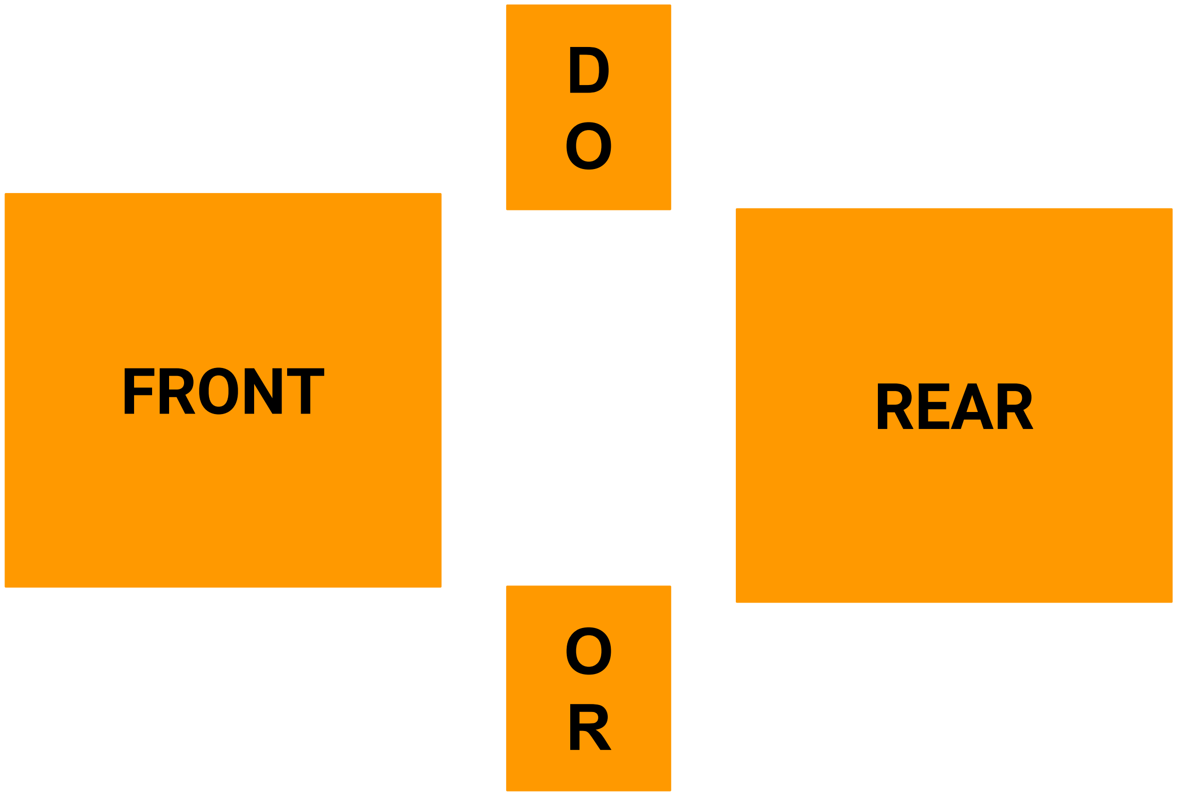
finite automata - the case of the automatic doors



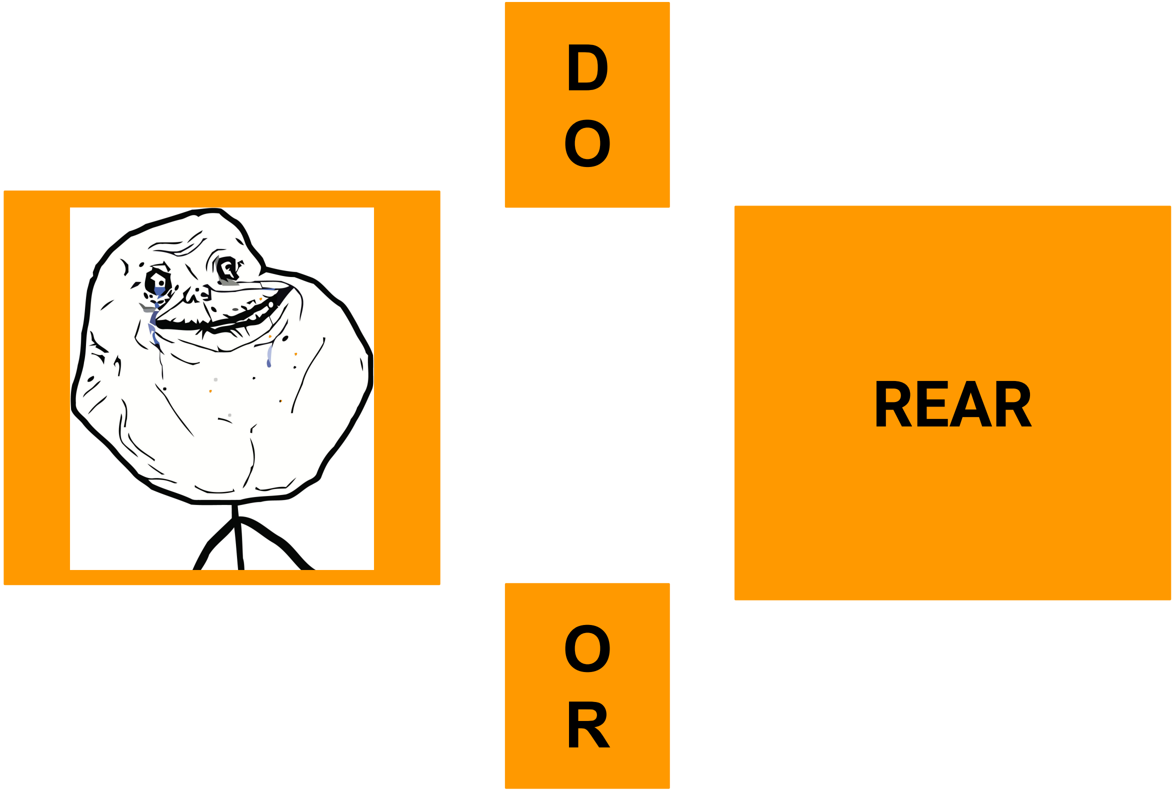
finite automata - the case of the automatic doors



finite automata - the case of the automatic doors



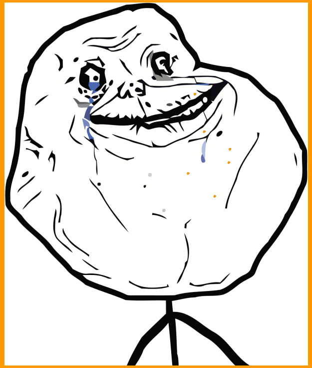
finite automata - the case of the automatic doors



finite automata - the case of the automatic doors

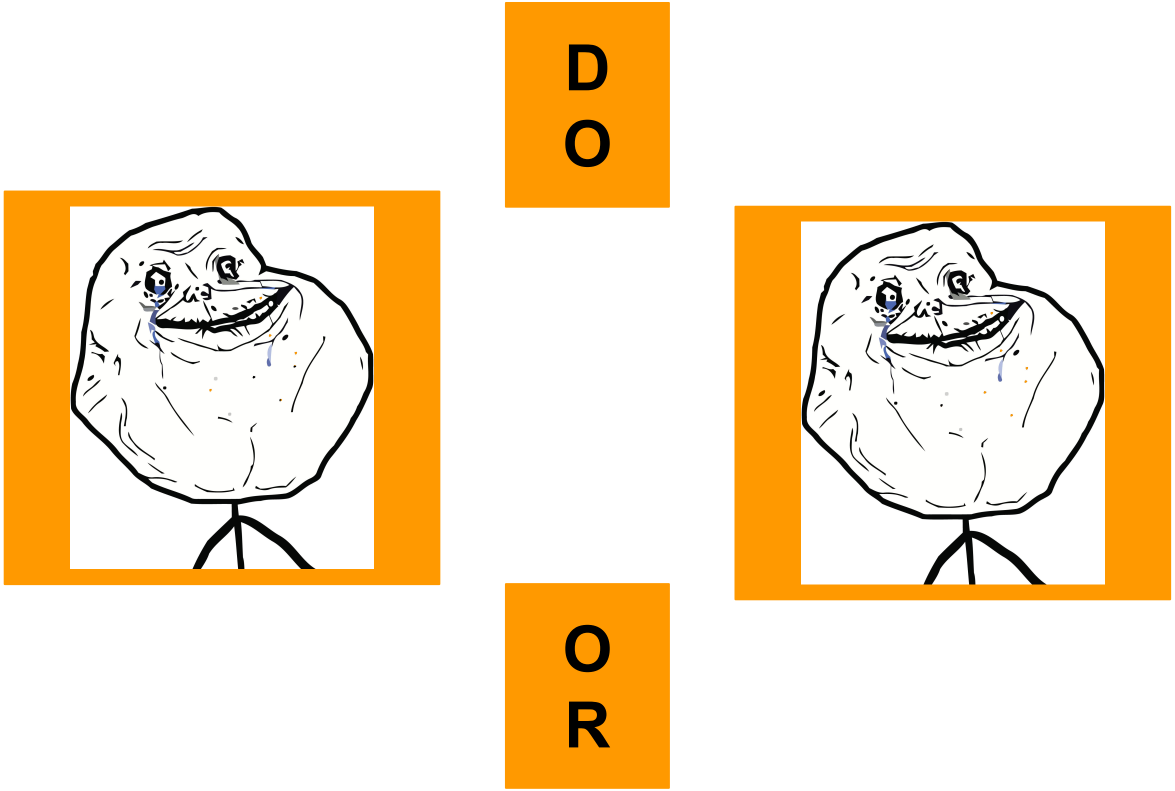
D
O

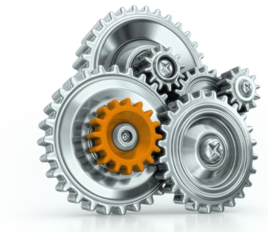
FRONT



O
R

finite automata - the case of the automatic doors

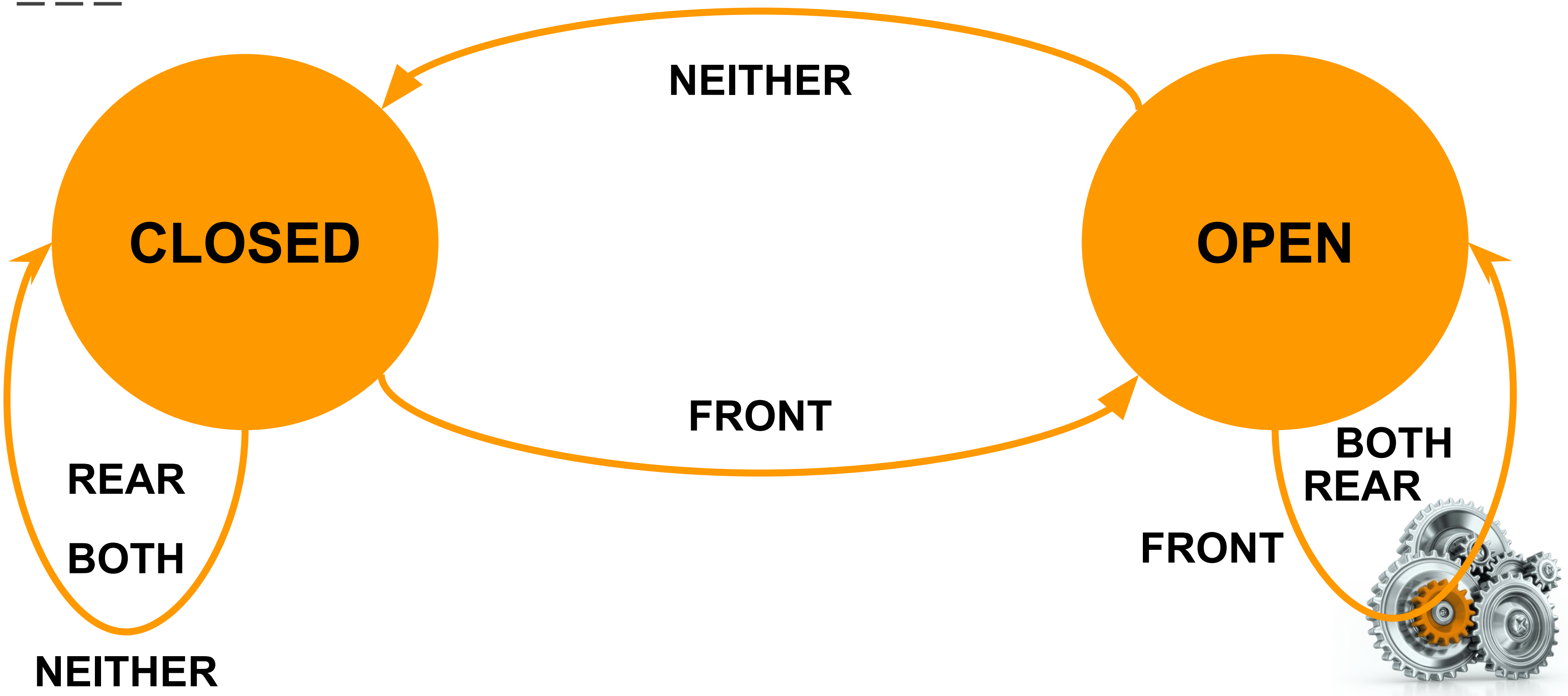




finite automata - the case of the automatic doors

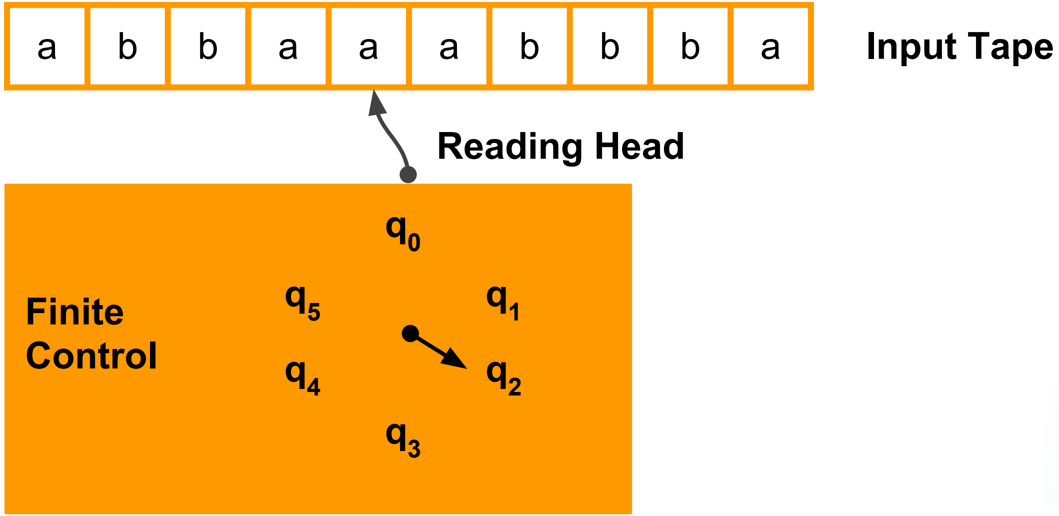
- states of the door
  - closed
  - open
- possible scenarios/inputs
  - neither
  - front presence only
  - rear presence only
  - both front and rear presence

finite automata - the case of the automatic doors



finite automata

- simplest model of computation
- machine designed to respond to encoded instructions
- has a “CPU” or controller with fixed, finite capacity



deterministic finite automaton

- A deterministic finite automaton is a quintuple $M = (K, \Sigma, \delta, s, F)$ where
  - K is a finite set of states
  - Σ is and alphabet
  - $S \in K$ is the initial state
  - $F \subseteq K$ is the of final states
  - δ, **the transition function**, is a function from $K \times \Sigma$ to K
    - If M is in state $q \in K$ and symbol read is $a \in \Sigma$, then $\delta(q,a) \in K$.

deterministic finite automaton

- computation
  - a sequence of **configurations** that represent the status of the machine
  - a **configuration** is determined by the current state and the unread part of the string
    - any element of $K \times \Sigma^*$
      - (q2, aabbba)

deterministic finite automaton

- the binary relation $\vdash_M$ holds between two configurations if and only if the machine can pass from one to the other as a result of a single move
  - if (q, w) and (q', w') are configurations of M, then (q, w) $\vdash$M (q', w') if and only if w=aw' for some symbol a $\in \Sigma$, and δ (q,a) = q'
    - (q, w) yields (q', w') in one step
    - $\vdash_M$ is a function from K x Σ^+ to K x Σ^*
    - A configuration of the form (q,e) means that M has consumed all its input and operation stops

deterministic finite automaton

- $(q, w) \vdash^*_M (q', w')$
  - (q, w) yields (q', w') after some number, possibly zero, steps
- a string $w \in \Sigma^*$ is said to be accepted by M if and only if there is a state $q \in F$ such that $(s, w) \vdash^*_M (q, e)$
- the language accepted by M, L(M), is the set of all strings accepted or recognized by M.

deterministic finite automaton

- Let M be the DFA $(K, \Sigma, \delta, s, F)$, where
  - $K = \{q_0, q_1\}$
  - $\Sigma = \{a, b\}$
  - $s = q_0$
  - $F = \{q_0\}$
- $(q_0, aabba)$
  - $\vdash_M (q_1, abba)$
  - $\vdash_M (q_0, bba)$
  - $\vdash_M (q_1, ba)$
  - $\vdash_M (q_0, a)$
  - $\vdash_M (q_1, e)$

q	σ	$\delta(q, \sigma)$
q_0	a	q_1
q_0	b	q_1
q_1	a	q_0
q_1	b	q_0

deterministic finite automaton

- Let M be the DFA $(K, \Sigma, \delta, s, F)$, where
  - $K = \{q_0, q_1, q_2, q_3\}$
  - $\Sigma = \{a, b\}$
  - $s = q_0$
  - $F = \{q_0, q_1, q_2\}$

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2

q	σ	$\delta(q, \sigma)$
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

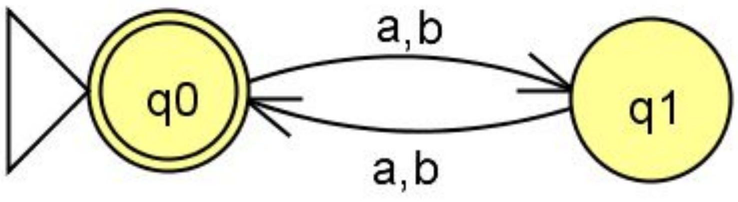
deterministic finite automaton

- Construct a DFA machine M that accepts the strings that have ($\Sigma = \{a, b\}$):
  - even b’s
  - “abba” as substring
  - at least three a's

deterministic finite automaton

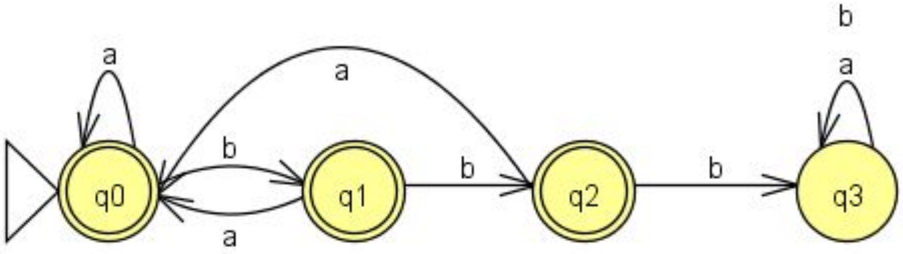
- state diagram
  - a graphical representation of a DFA

q	σ	$\delta(q, \sigma)$
q_0	a	q_1
q_0	b	q_1
q_1	a	q_0
q_1	b	q_0



deterministic finite automaton

- Let M be the DFA $(K, \Sigma, \delta, s, F)$
  - $K = \{q_0, q_1, q_2, q_3\}$
  - $\Sigma = \{a, b\}$
  - $s = q_0$
  - $F = \{q_0, q_1, q_2\}$



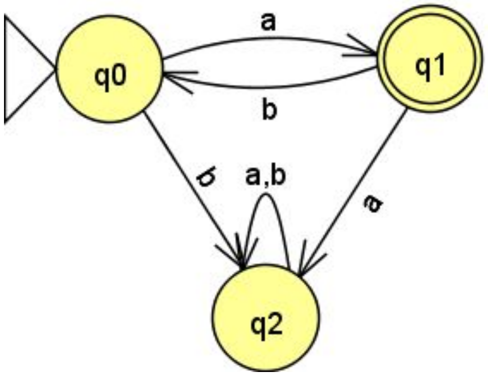
q	σ	δ(q, σ)
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2

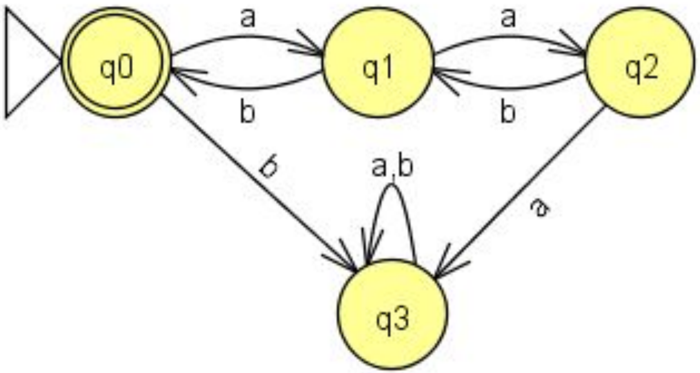
q	σ	δ(q, σ)
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

deterministic finite automaton

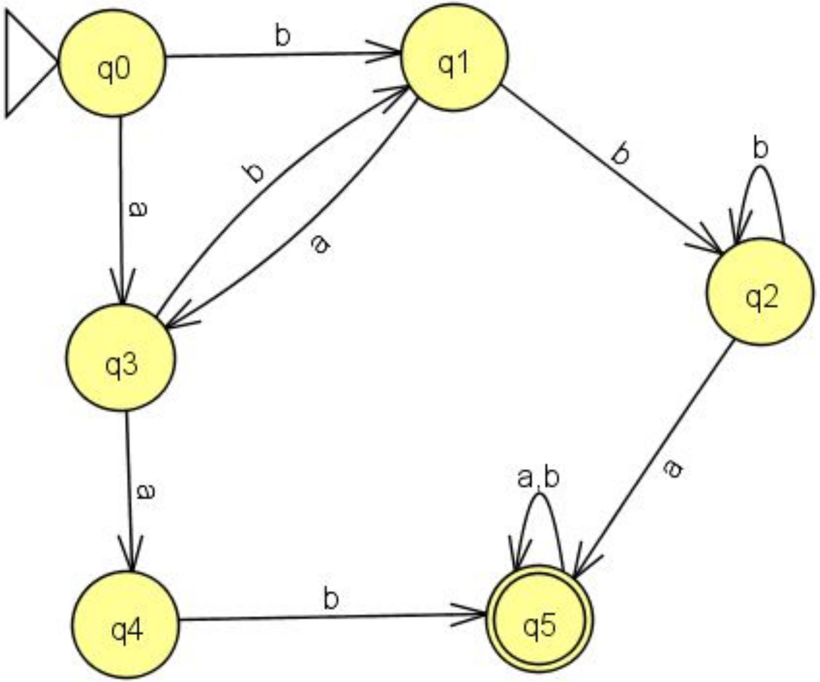
- Construct a state diagram that accepts the strings that have (Σ = {a, b}):
  - even b’s
  - “abba” as substring
  - at least three a's

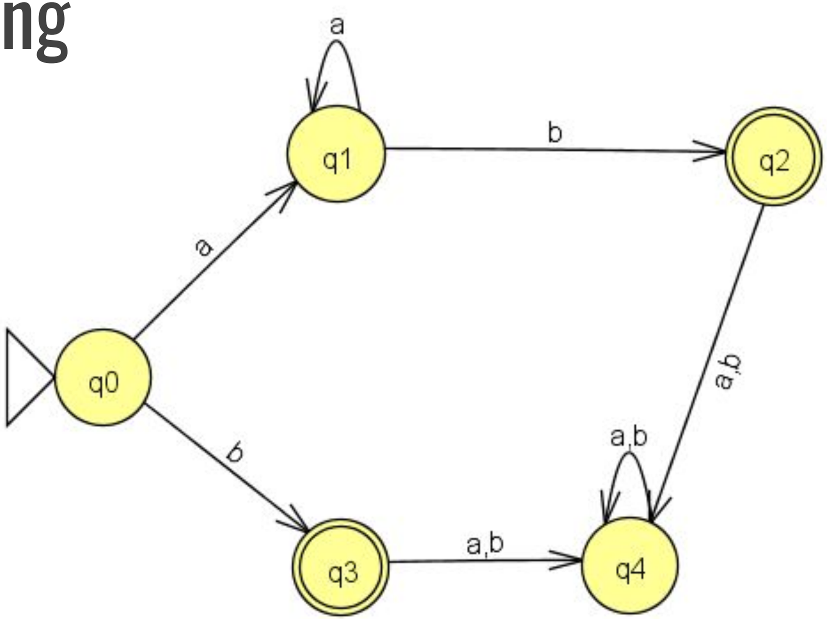
exercise - describe the following





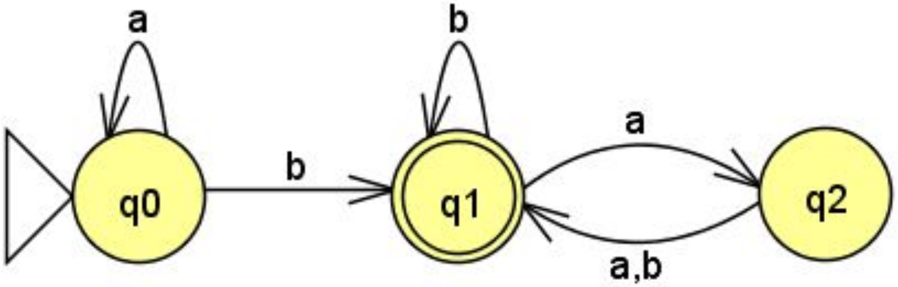
exercise - describe the following

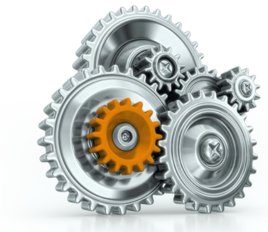




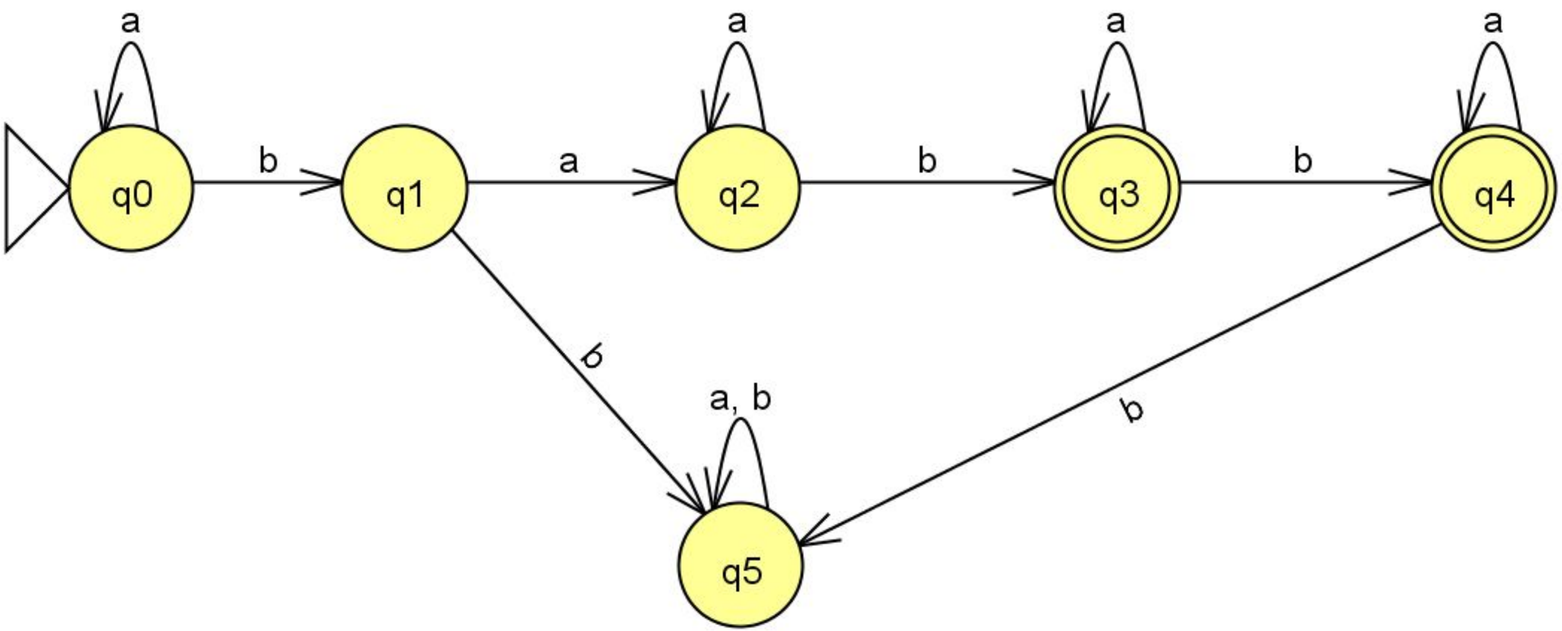
quiz - ¼ sheet of paper

- Convert the following FSM into its equivalent regular expression
- What types of strings does the FSM accept?

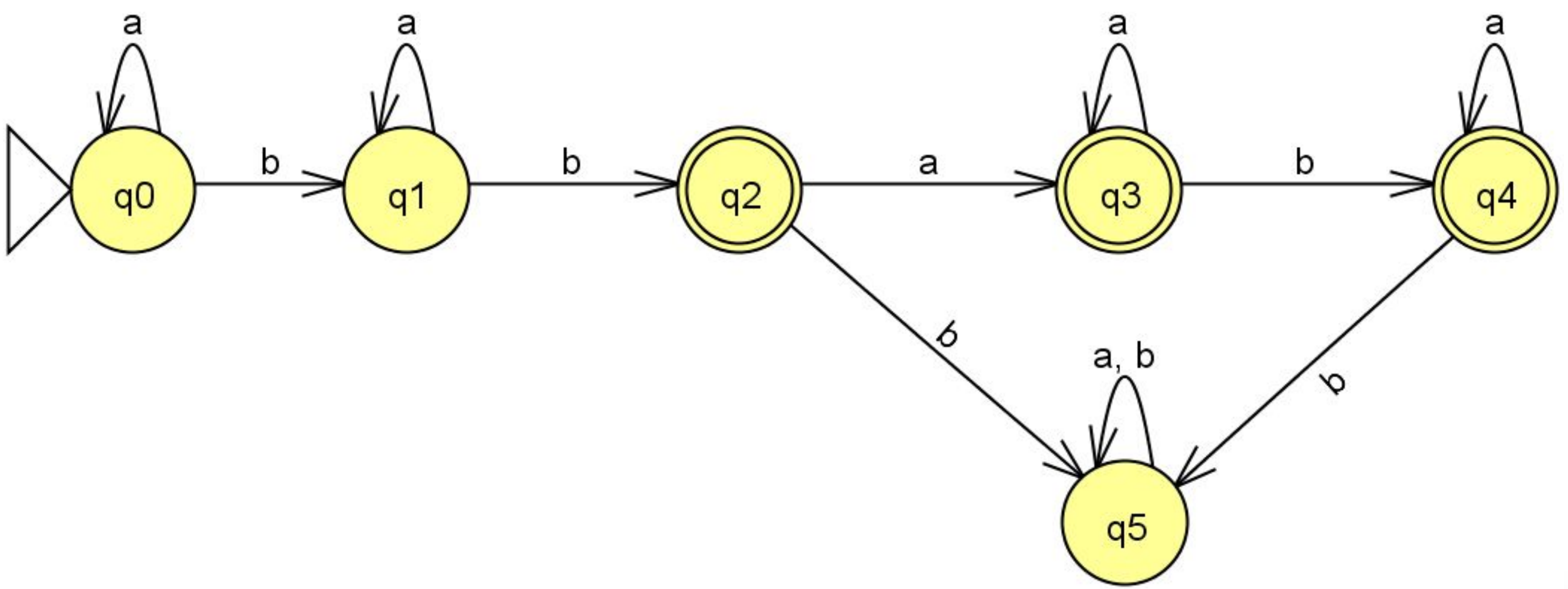




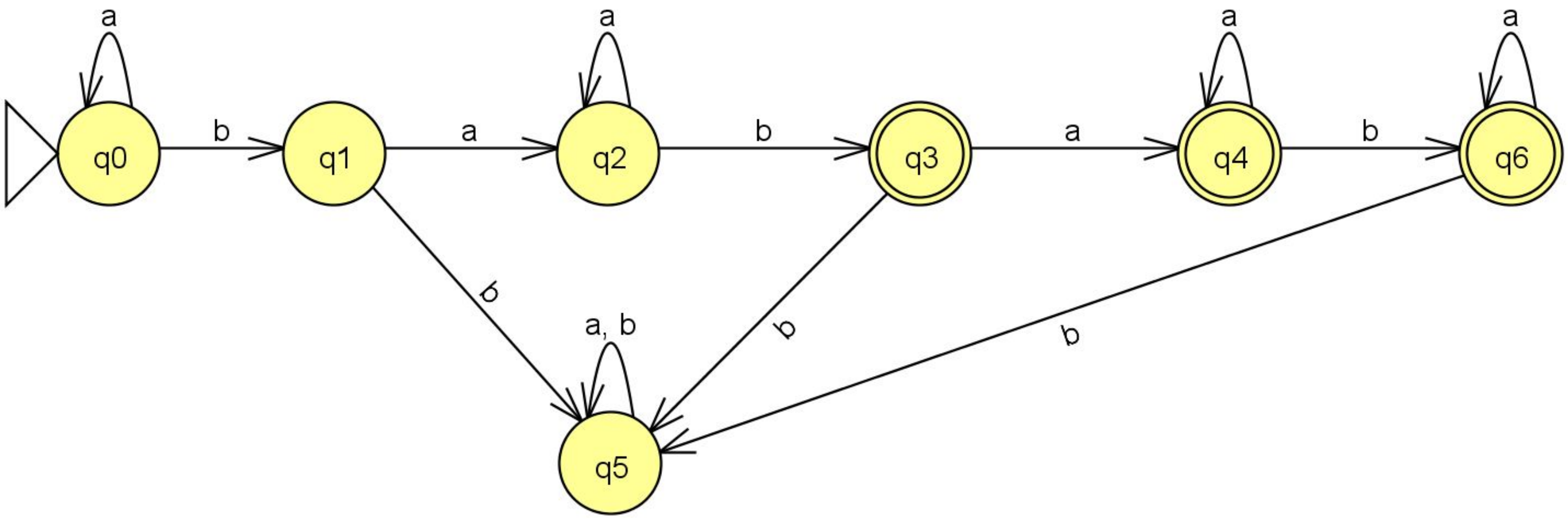
more on FSM via State Diagram



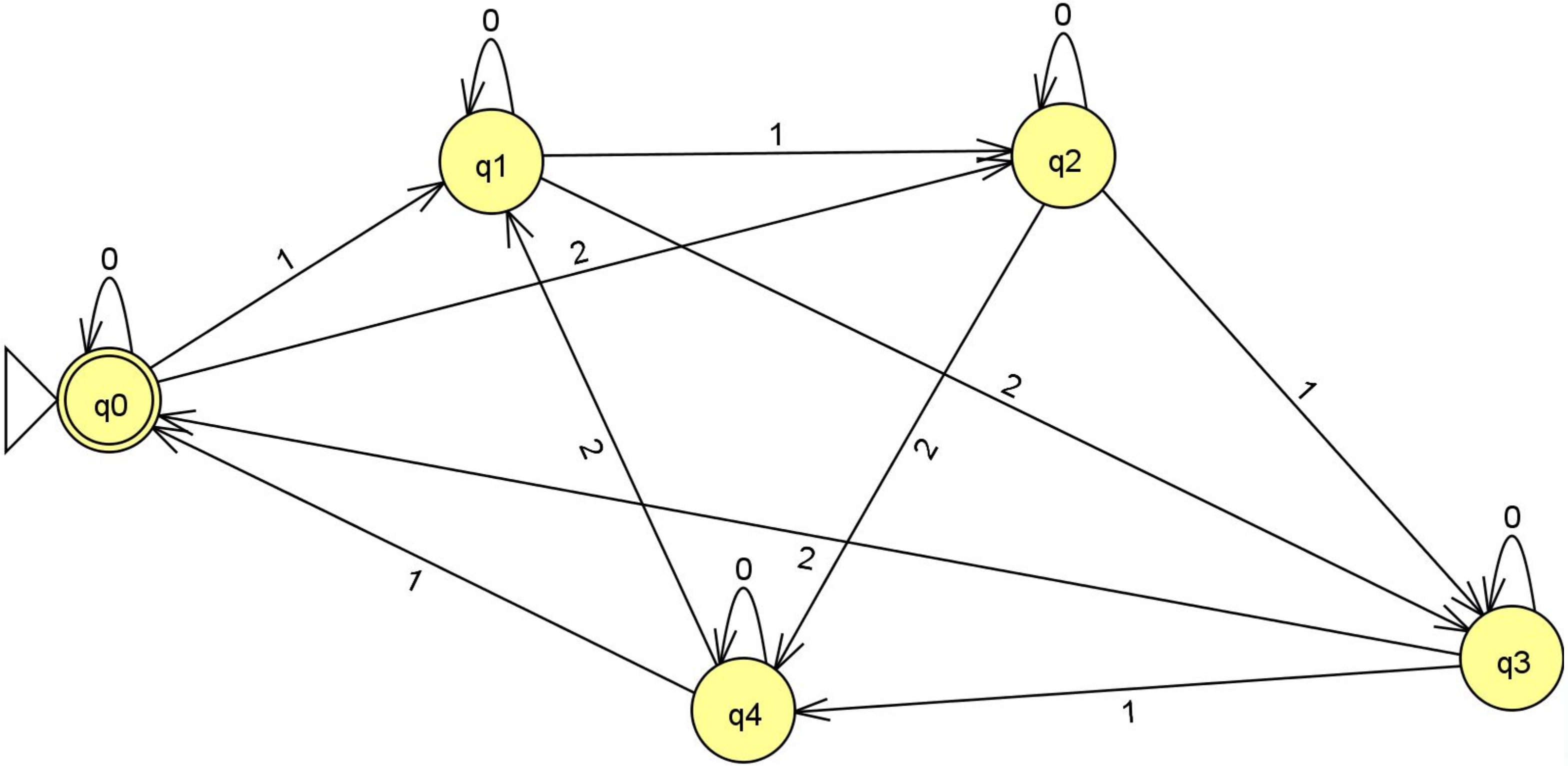
more on FSM via State Diagram



more on FSM via State Diagram



more on FSM via State Diagram



exercise

Construct a deterministic FSM to accept each of the following languages:

- {w ∈ {a, b}* : each ‘a’ in w is immediately preceded and followed by a ‘b’}
- {w ∈ {a, b}* : w has abab as a substring}
- {w ∈ {a, b}* : w has neither aa nor bb as a substring}
- {w ∈ {a, b}* : w has both ab and ba as substrings}
- {w ∈ {a, b}* : every odd position of w is a b}
- {w ∈ {a, b}* : w has an odd number of a's and an odd number of b's}
- {w ∈ {a, b}* : w has an odd number of a's and an even number of b's}

quiz - ¼ sheet of paper

- construct a state diagram accepting the language generated by each of the following regular expressions over $\Sigma = \{0, 1\}$
  - 0*(0(1 U e) U (1 U e)0)0*
  - 0* U (((0*(1 U (11)))((00*)(1 U (11)))*)0*)

non-determinism

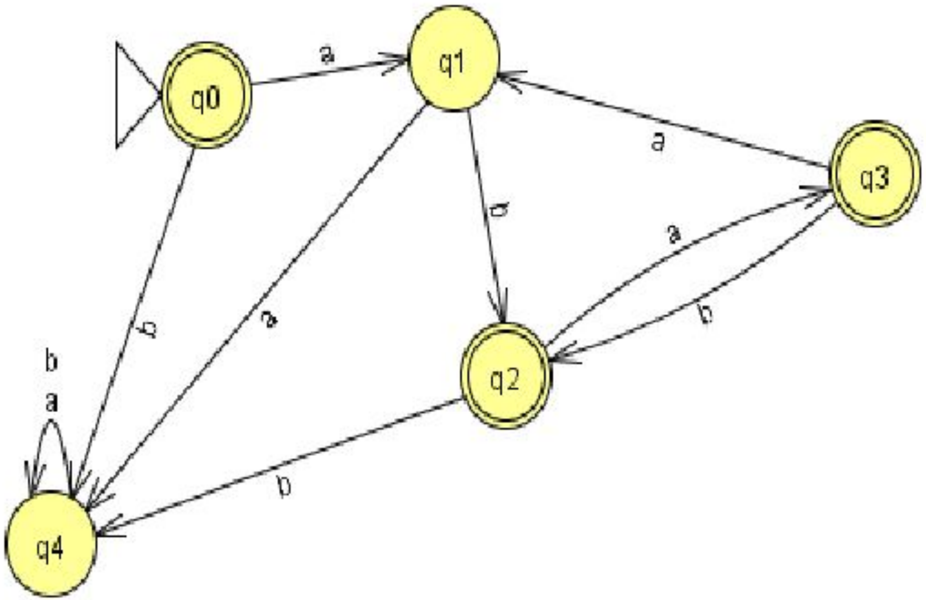
- Add a new feature to the automaton, nondeterministic property
  - Ability to change states in a way that is only partially determined by the current state and input symbol
  - Allow possible next states for a given combination of current state and input symbol
  - But allow next states that are legal from a given state with a given input

non-determinism

- as it reads the input string, may choose at each step to go into any of these legal next states
- the choice is not determined by anything in our model, and is therefore said to be **nondeterministic**
- the choice is not wholly unlimited
  - only those next states that are legal from a given state with a given input symbol can be chosen

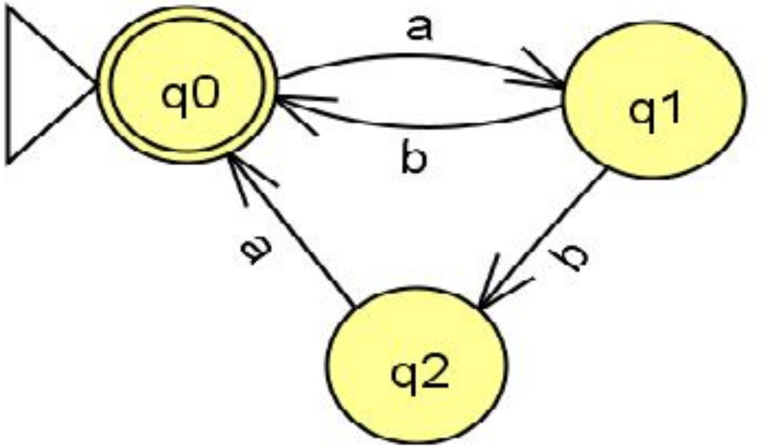
non-determinism

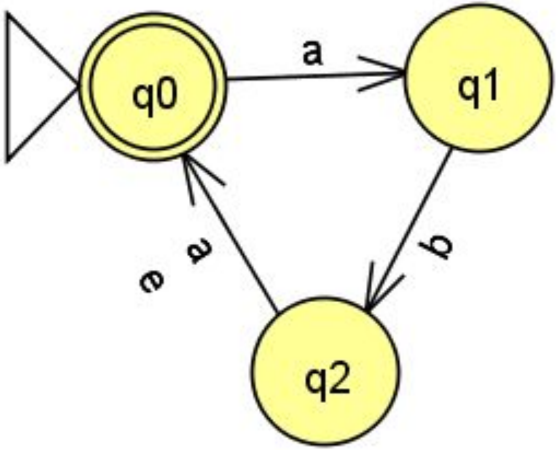
- not meant as realistic models of computers
  - notational generalizations
  - L = (ab U aba)*



non-determinism

- not meant as realistic models of computers
  - notational generalizations
  - L = (ab U aba)*





non-determinism

- A nondeterministic finite automaton is a quintuple M = (K, Σ, Δ, s, F) where
  - K is a finite set of states
  - Σ is and alphabet combination of current state and input symbol
  - s ∈ K is the initial state
  - F ⊆ K is the of final states
  - Δ, the transition function, is a function from K x (Σ U {e}) x K

non-determinism

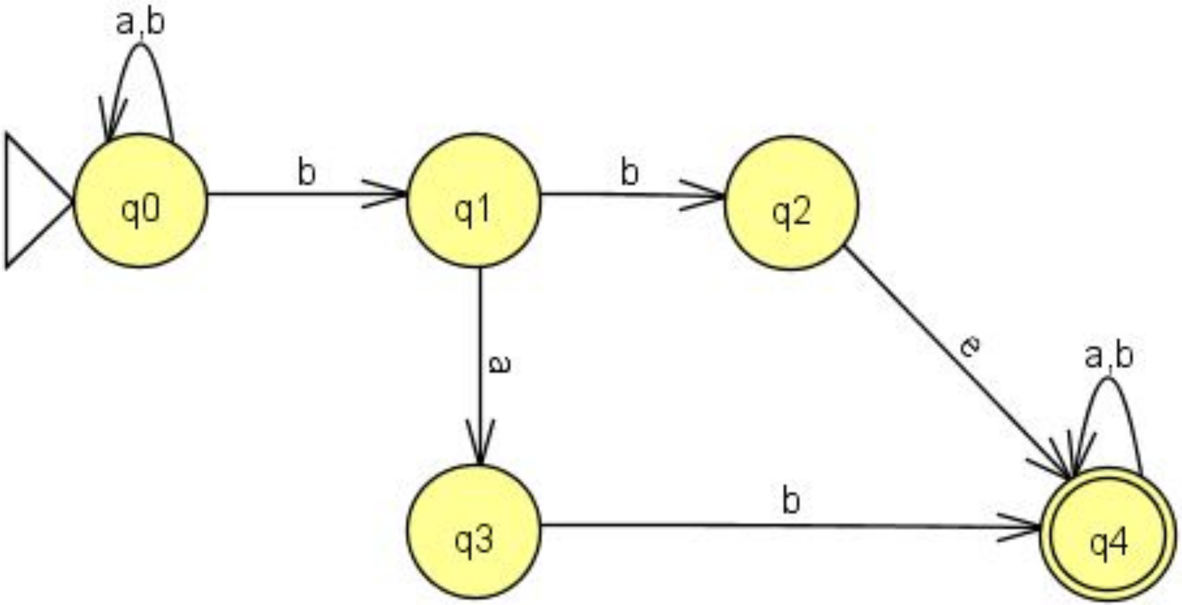
- Each triple (q,u,p) ∈ Δ is called a **transition** of M
- If M is in state q and the next input symbol is *a*
  - (q,a,p)
  - (q,e,p) – no input symbol is read

non-determinism

- $\vdash_M$
  - If (q, w) and (q', w') are configurations of M, then (q, w) $\vdash_M$ (q', w') if and only if w=aw' for some symbol a $\in \Sigma \cup \{e\}$, and (q,a,q') $\in \Delta$
  - A string w $\in \Sigma^*$ is said to be accepted by M if and only if there is a state q $\in$ F such that (s, w) $\vdash^*_M$ (q, e)
  - The language accepted by M, L(M), is the set of all strings accepted by M

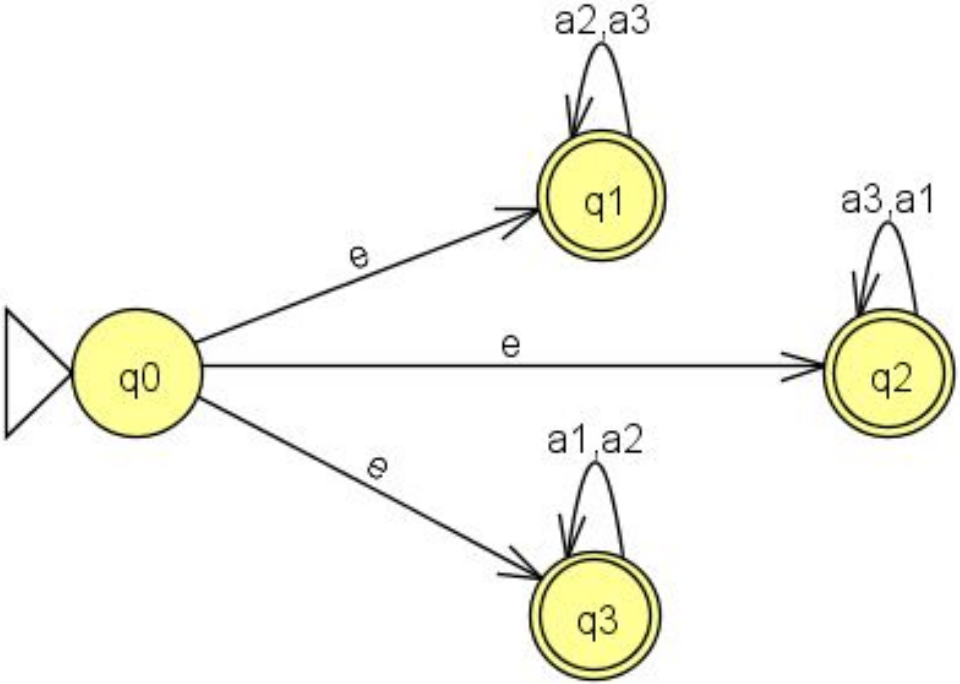
non-determinism

- set of all strings containing bb or bab



non-determinism

- $\Sigma = \{a_1, a_2, a_3\}$

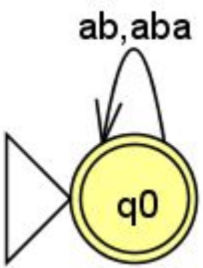


non-determinism

- A nondeterministic finite automaton is a quintuple M = (K, Σ, Δ, s, F) where
  - K is a finite set of states
  - Σ is and alphabet combination of current state and input symbol
  - s $\in$ K is the initial state
  - F $\subseteq$ K is the of final states
  - Δ, the transition function, is a function from K x Σ^* x K

non-determinism

- not meant as realistic models of computers
  - notational generalizations
  - L = (ab U aba)*



construct NFAs for the following regular expressions

- $\Sigma = \{a, b\}$
  - $(ab)^*(ba)^* \cup aa^*$
  - $((ab \cup aab)^*a^*)^*$
  - $((a^*b^*a^*)^*b)^*$
  - $(ba \cup b)^* \cup (bb \cup a)^*$

exercise

Construct a nondeterministic FSM to accept each of the following languages:

- {w ∈ {a, b}* : each ‘a’ in w is immediately preceded and followed by a ‘b’}
- {w ∈ {a, b}* : w has abab as a substring}
- {w ∈ {a, b}* : w has neither aa nor bb as a substring}
- {w ∈ {a, b}* : w has both ab and ba as substrings}

theorem: For each nondeterministic finite automaton, there is an equivalent deterministic finite automaton

- Let $M = (K, \Sigma, \Delta, s, F)$ be a nondeterministic finite automaton.
- To prove the theorem, we construct $M' = (K', \Sigma, \delta, s', F')$ equivalent to M
  - $K' = 2^K$
  - $s' = E(s)$
  - $F' = \{Q \subseteq K: Q \cap F \neq \varnothing\}$
  - δ

nfa and dfa equivalence

- An NFA is a DFA provided that
  - If$(q, u, q') \in \Delta$ then $|u| = 1$
  - (q, e, q') transitions are absent in Δ
  - For each $q \in K$ and $\sigma \in \Sigma$, there is a unique $q' \in K$ s.t. $(q, \sigma, q') \in \Delta$
- Finite automata M1 and M2 are equivalent iff $L(M1) = L(M2)$
  - $w \in L(M1)$ then $w \in L(M2)$

nfa and dfa equivalence

- How to convert an NFA to DFA
  - Transitions of the form (q, u, q') s.t. |u| > 1
  - Transitions of the form (q, e, q')
  - Missing transitions
  - Multiple transitions

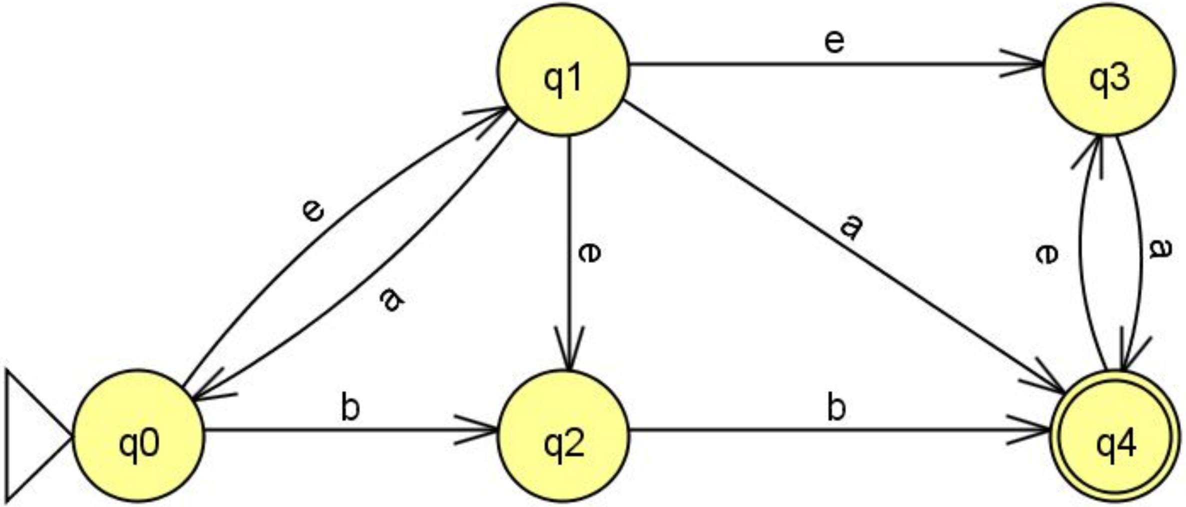
nfa and dfa equivalence

- Let $M = (K, \Sigma, \Delta, s, F)$ be an NFA
- Step 1: Eliminate "string" transitions
  - Formally, if $(q, \sigma_1\sigma_2\ldots\sigma_k, q') \in \Delta$ and $\sigma_1, \sigma_2, \ldots \sigma_k, \in \Sigma$, $k>1$, add new non-final states $q_1, \ldots q_k$-1 to K and new transitions $(q,\sigma_1,q_1), (q_1,\sigma_2,q_2), \ldots (q_{k-1}, \sigma_k, q')$ to Δ
  - Let $M' = (K', \Sigma, \Delta', s', F')$ be the NFA that resulted when the transformation was applied to each transition (q,u,q'), $|u| > 1$
  - Obviously, $L(M) = L(M')$

nfa and dfa equivalence

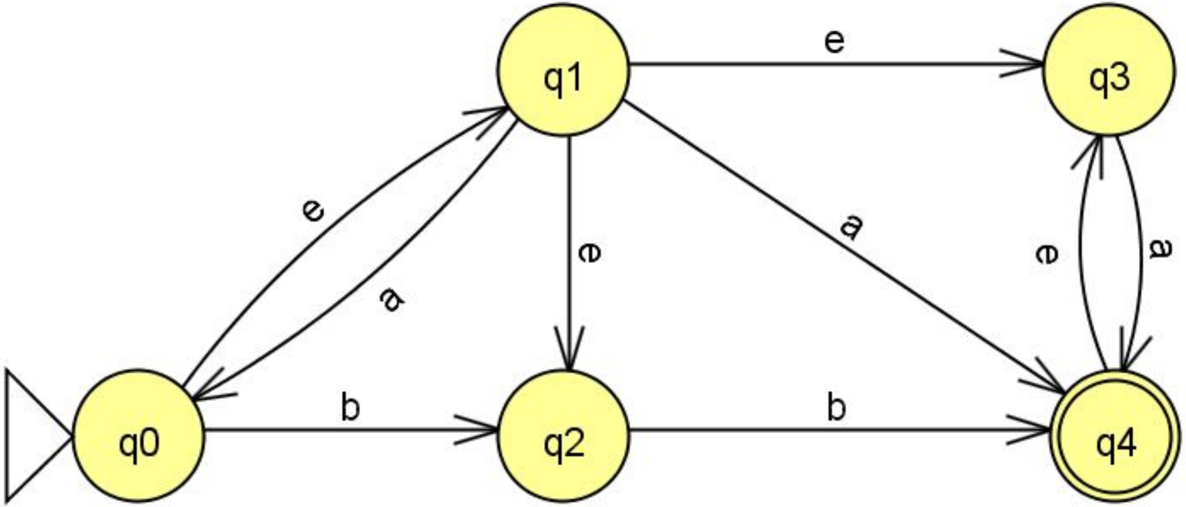
- Construct M'' = (K'', Σ, Δ'', s'', F'') equivalent to M'
- Step 2: Eliminate e-transitions
  - For any state q ∈ K', let E(q) be the set of all states of M' that are reachable from state q without reading any input.
    - $E(q) = \{ p \in K' \mid (q,e) \vdash^*_M (p,e) \}$
    - If M' moves without reading any input then its operation clearly does not depend on what that input is.

nfa and dfa equivalence



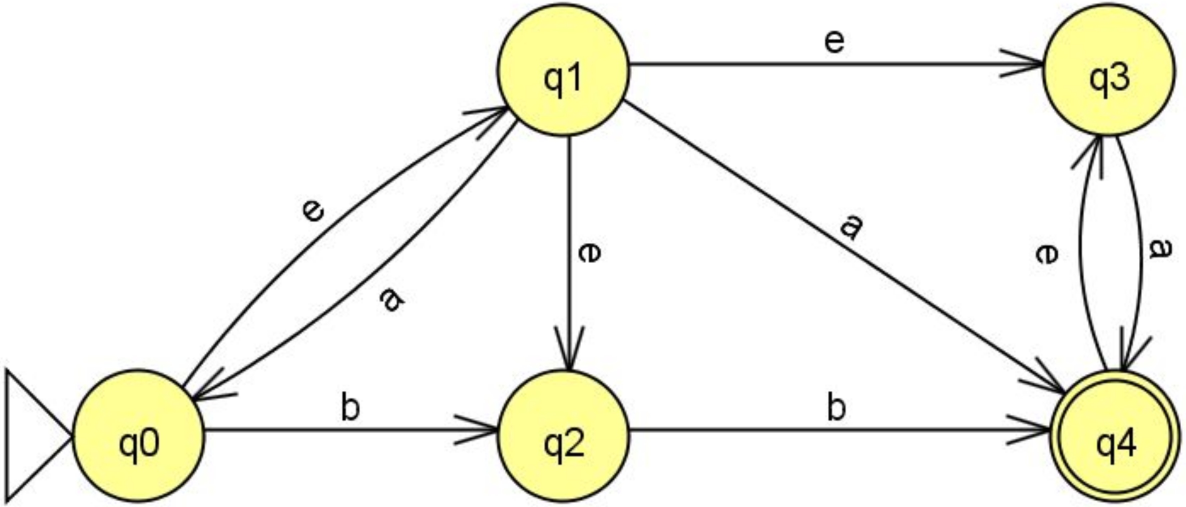
$E(q_0) = \{q_0, q_1, q_2, q_3\}$

nfa and dfa equivalence



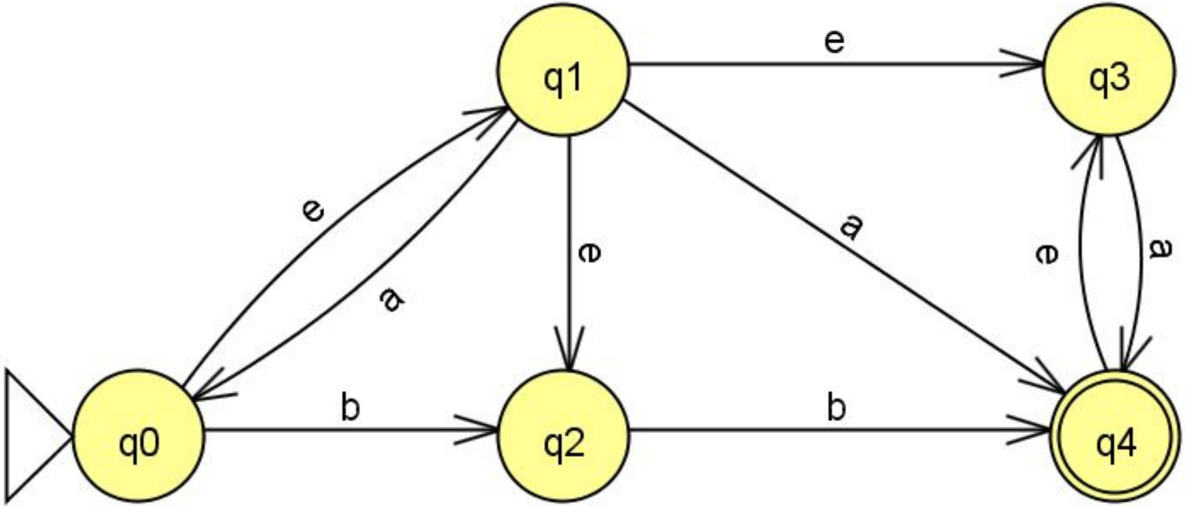
$E(q_1) = \{q_1, q_2, q_3\}$

nfa and dfa equivalence



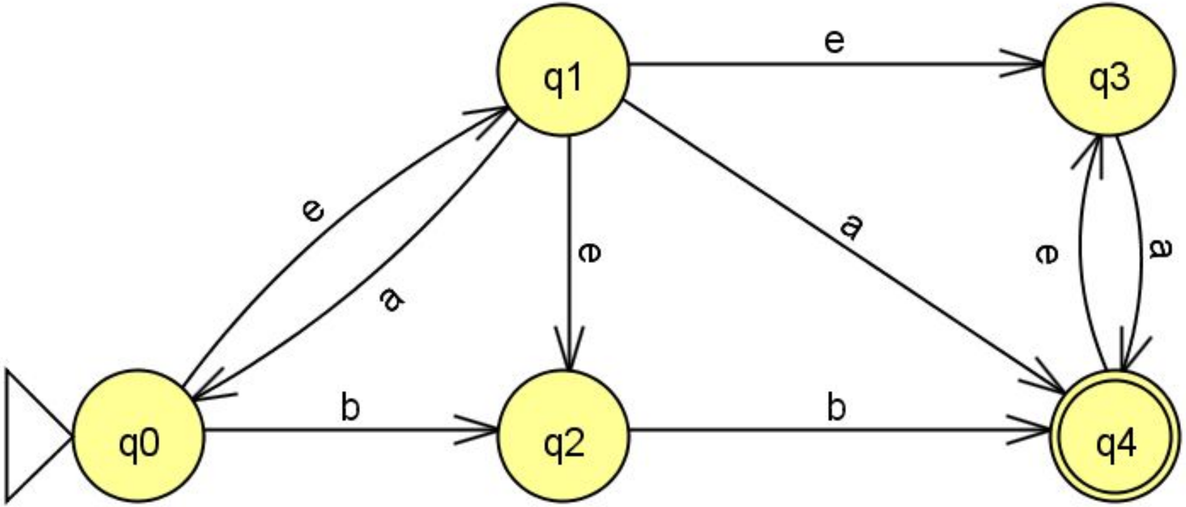
$E(q_2) = \{q_2\}$

nfa and dfa equivalence



$E(q_3) = \{q_3\}$

nfa and dfa equivalence

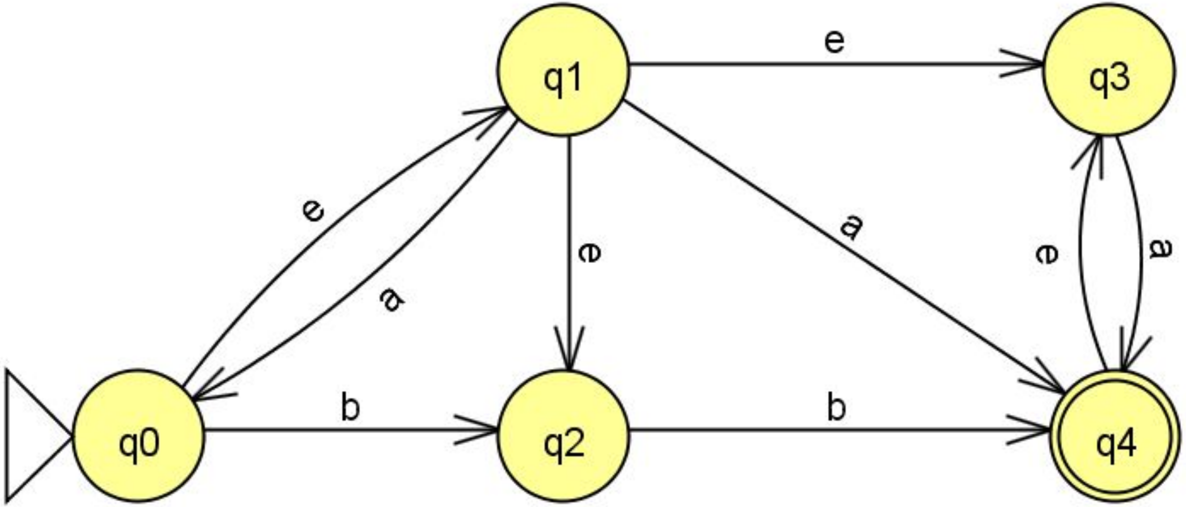


$E(q_4) = \{q_3, q_4\}$

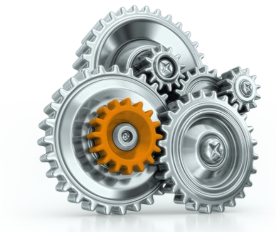
nfa and dfa equivalence

- Define $M'' = (K'', \Sigma, \Delta'', s'', F'')$
  - $K'' = 2^{K'}$
  - $s'' = E(s')$
  - $F'' = \{Q \subseteq K' \mid Q \cap F' \neq \emptyset\}$
  - and for each $Q \subseteq K$ and each symbol $a \in \Sigma$, define $\Delta''(Q, a) = \bigcup \{E(p) \mid p \in K' \text{ and } (q,a,p) \in \Delta' \text{ for some } q \in Q\}$

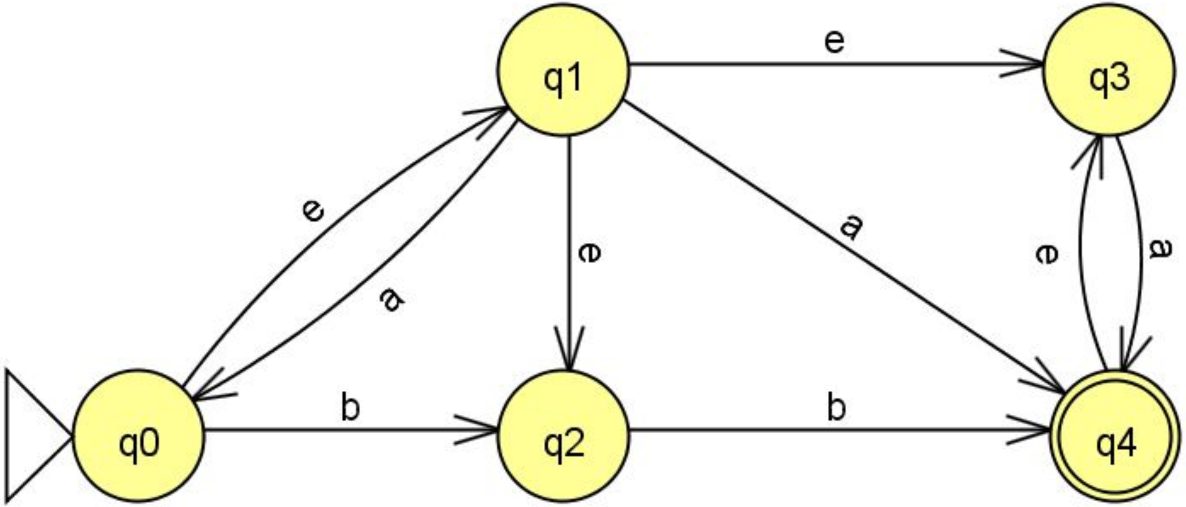
nfa and dfa equivalence



$s'' = E(s') = \{q_0, q_1, q_2, q_3\}$

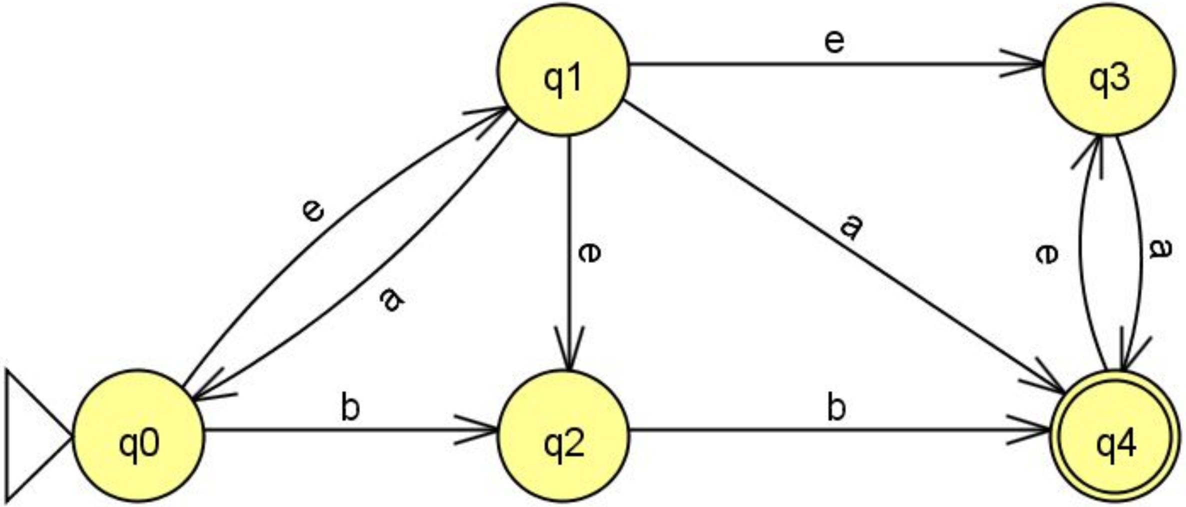


nfa and dfa equivalence



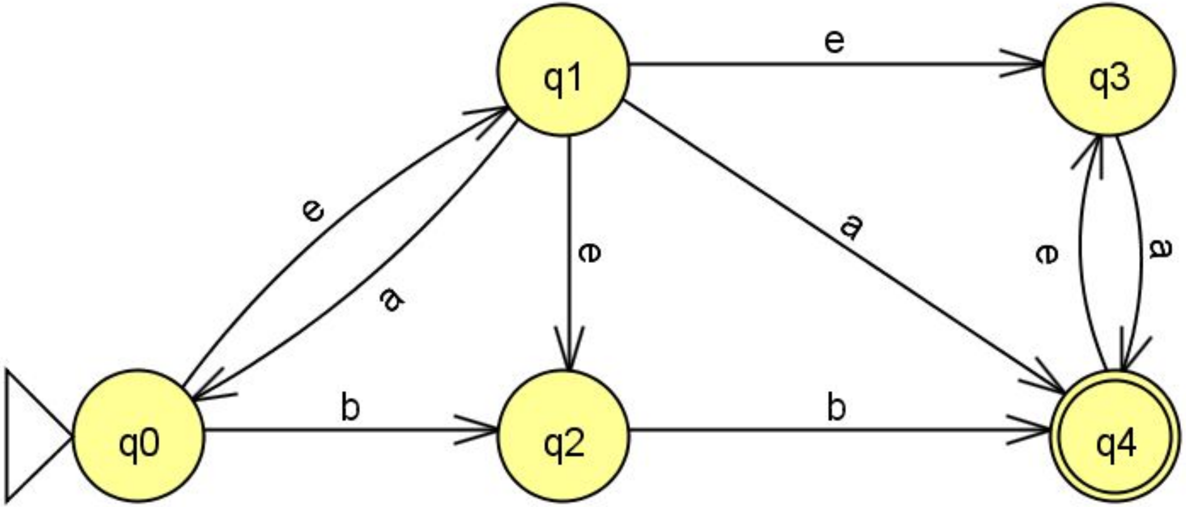
$\Delta'' (\{q_0,q_1,q_2,q_3\}, a) = E(q_0) \cup E(q_4) = \{q_0,q_1,q_2,q_3,q_4\}$

nfa and dfa equivalence



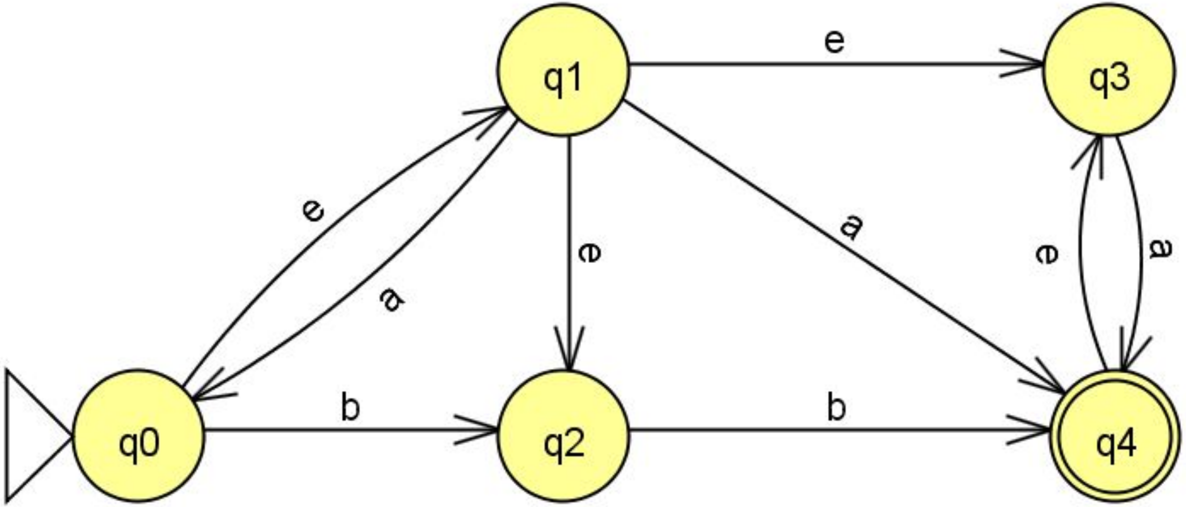
$\Delta''(\{q_0, q_1, q_2, q_3\}, b) = E(q_2) \cup E(q_4) = \{q_2, q_3, q_4\}$

nfa and dfa equivalence



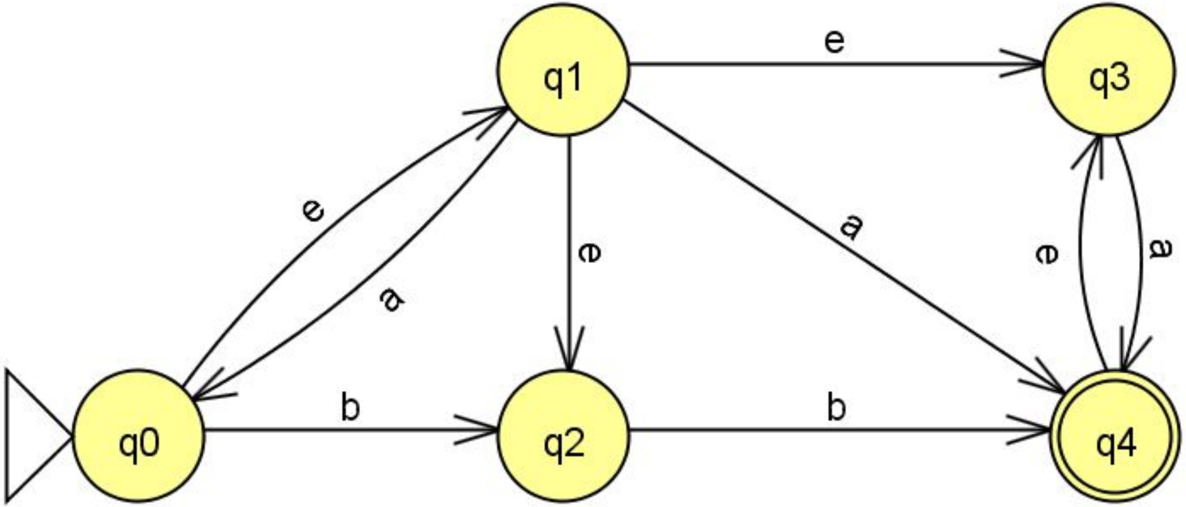
$\Delta''(\{q_0, q_1, q_2, q_3, q_4\}, a) = E(q_0) \cup E(q_4) = \{q_0, q_1, q_2, q_3, q_4\}$

nfa and dfa equivalence



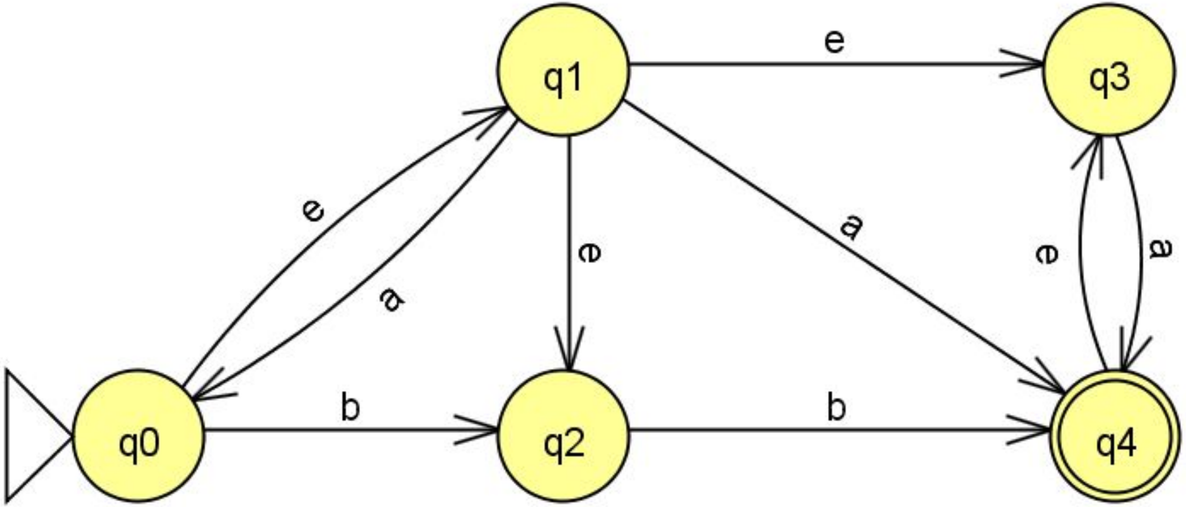
$\Delta''(\{q_0, q_1, q_2, q_3, q_4\}, b) = E(q_2) \cup E(q_4) = \{q_2, q_3, q_4\}$

nfa and dfa equivalence



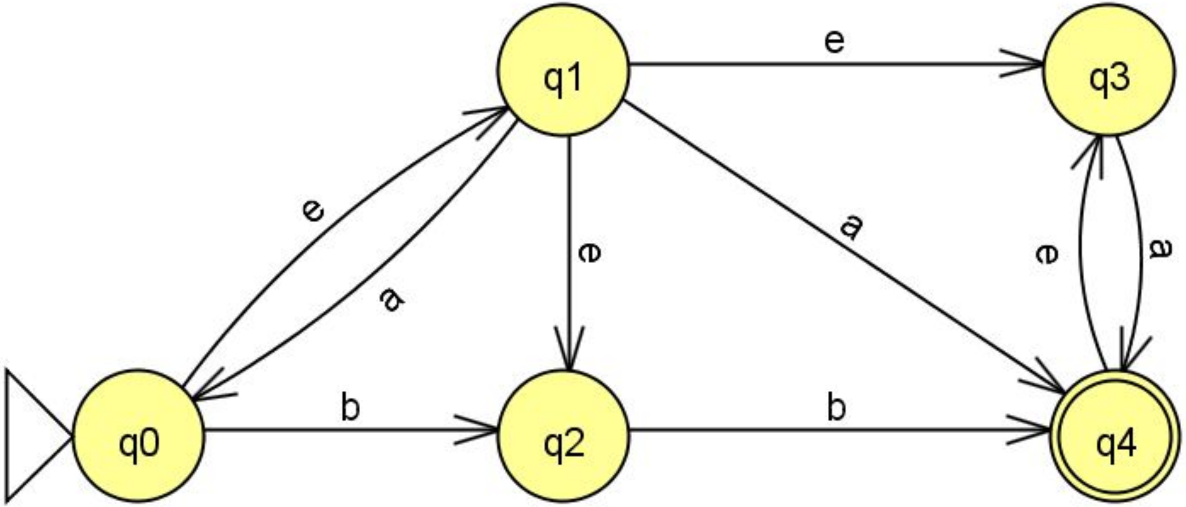
$$\Delta'' (\{q_2, q_3, q_4\}, a) = E(q_4) = \{q_3, q_4\}$$

nfa and dfa equivalence



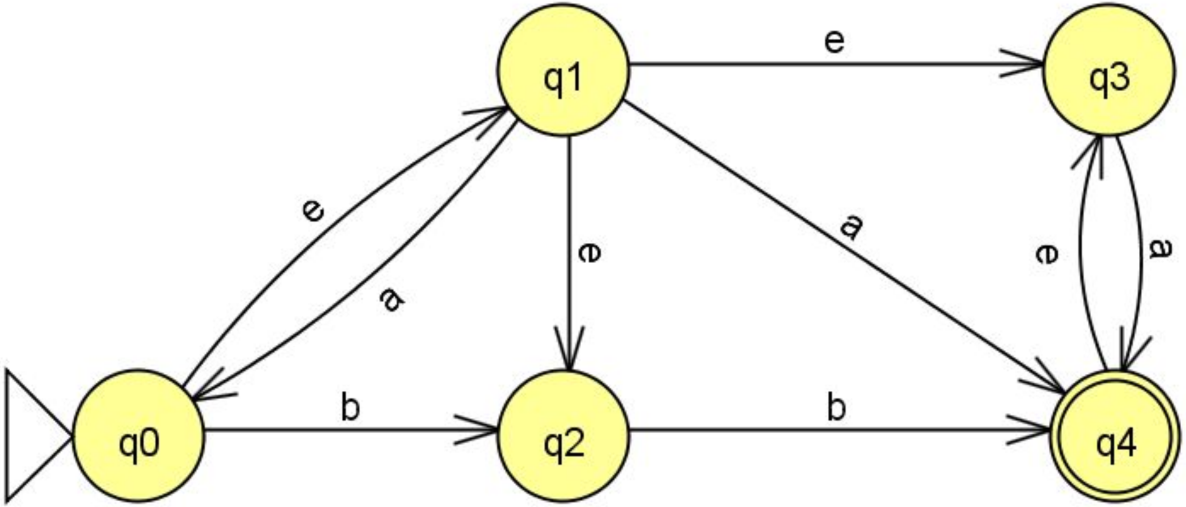
$$\Delta'' (\{q_2, q_3, q_4\}, b) = E(q_4) = \{q_3, q_4\}$$

nfa and dfa equivalence



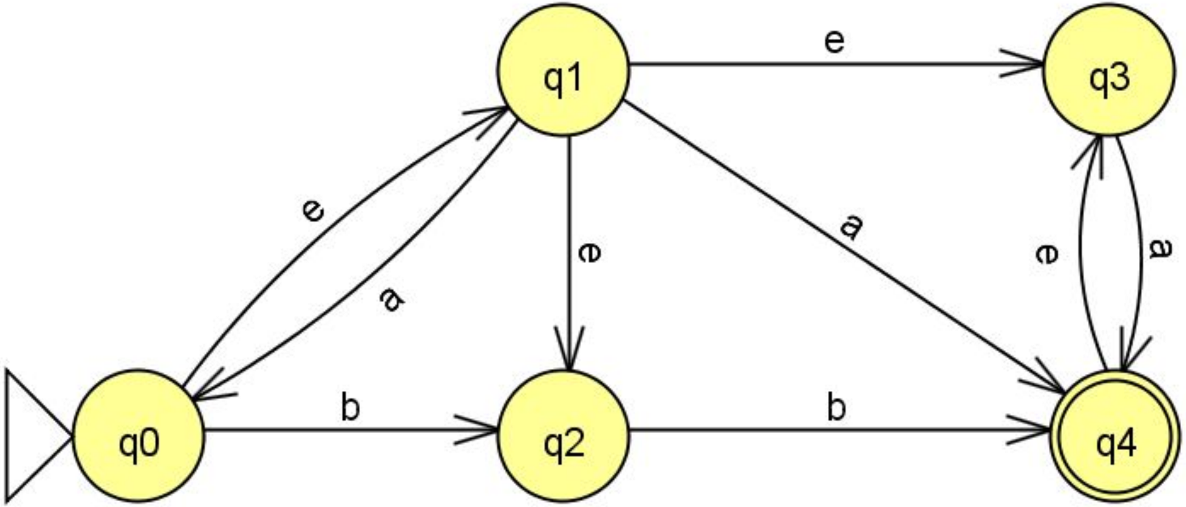
$$\Delta''(\{q_3, q_4\}, a) = E(q_4) = \{q_3, q_4\}$$

nfa and dfa equivalence



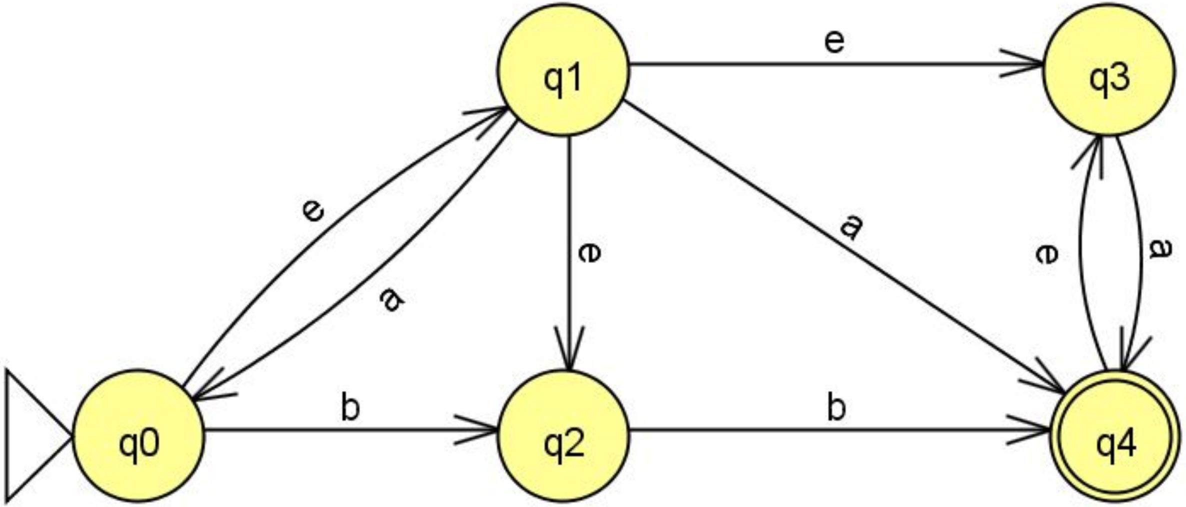
$\Delta'' (\{q_3, q_4\}, b) = \emptyset$

nfa and dfa equivalence



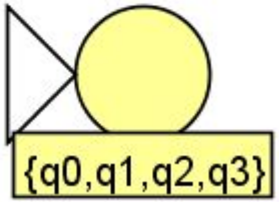
$\Delta''(\emptyset, a) = \emptyset$

nfa and dfa equivalence



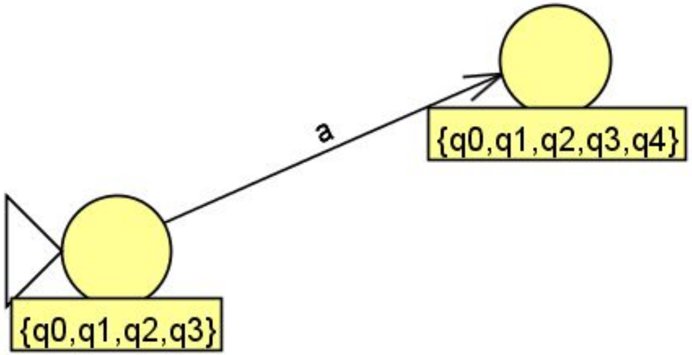
$\Delta''(\varnothing, b) = \varnothing$

nfa and dfa equivalence



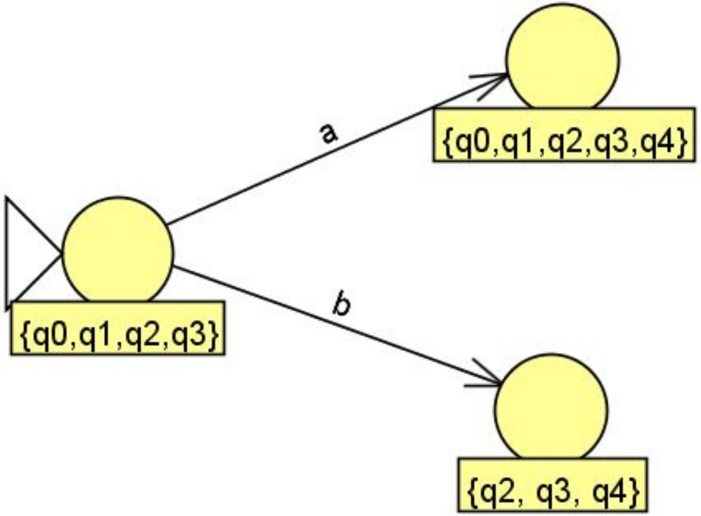
$s'' = E(s') = \{q_0, q_1, q_2, q_3\}$

nfa and dfa equivalence



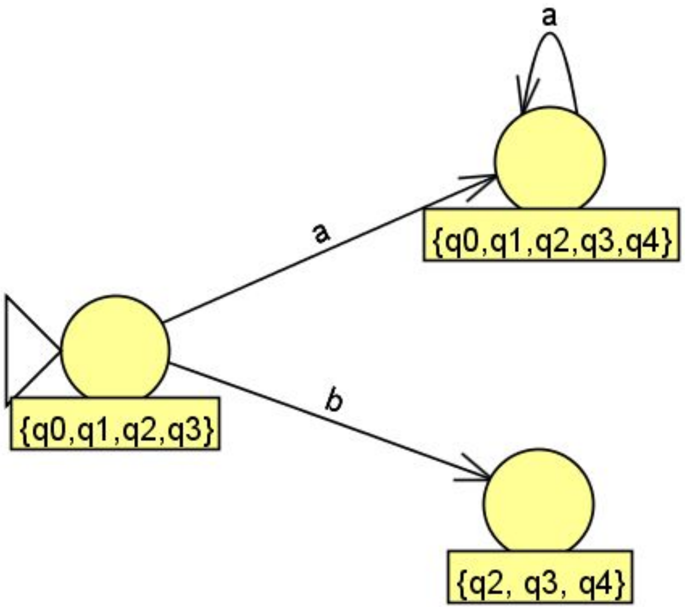
$\Delta''(\{q_0, q_1, q_2, q_3\}, a) = \{q_0,q_1,q_2,q_3,q_4\}$

nfa and dfa equivalence

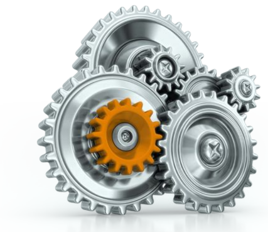


$\Delta''(\{q_0, q_1, q_2, q_3\}, b) = \{q_2, q_3, q_4\}$

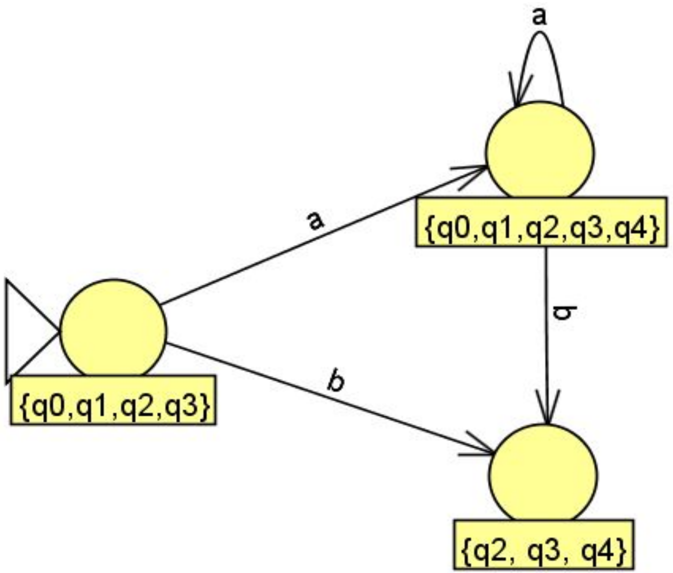
nfa and dfa equivalence



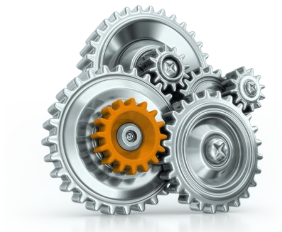
$\Delta'' (\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$



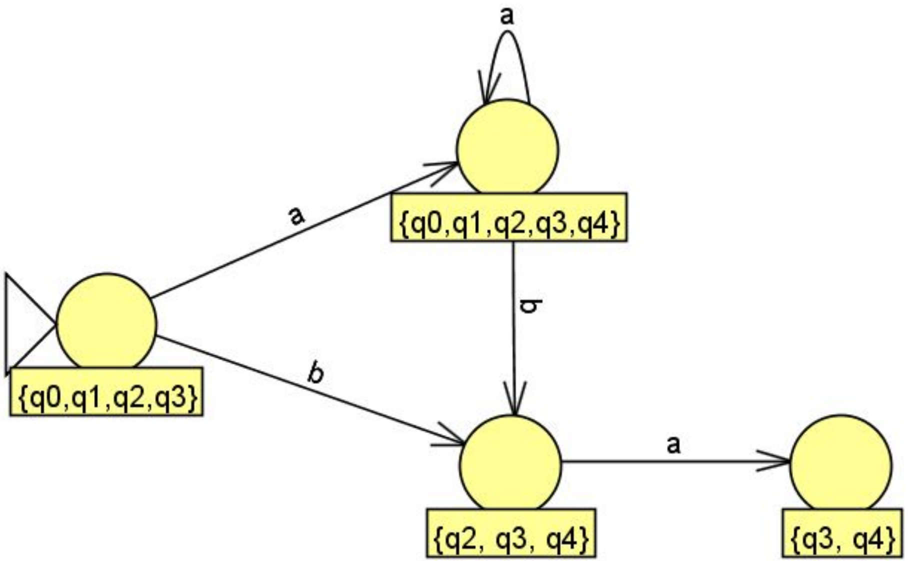
nfa and dfa equivalence



$\Delta'' (\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$

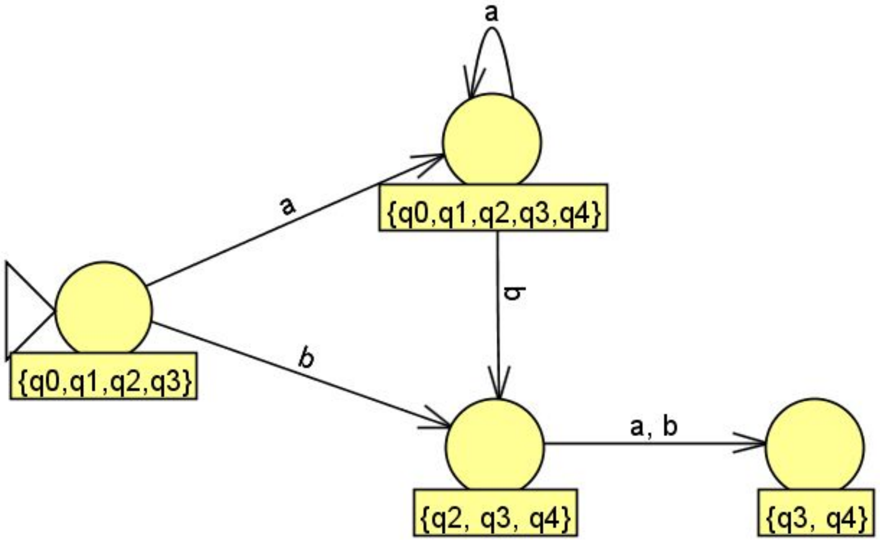


nfa and dfa equivalence



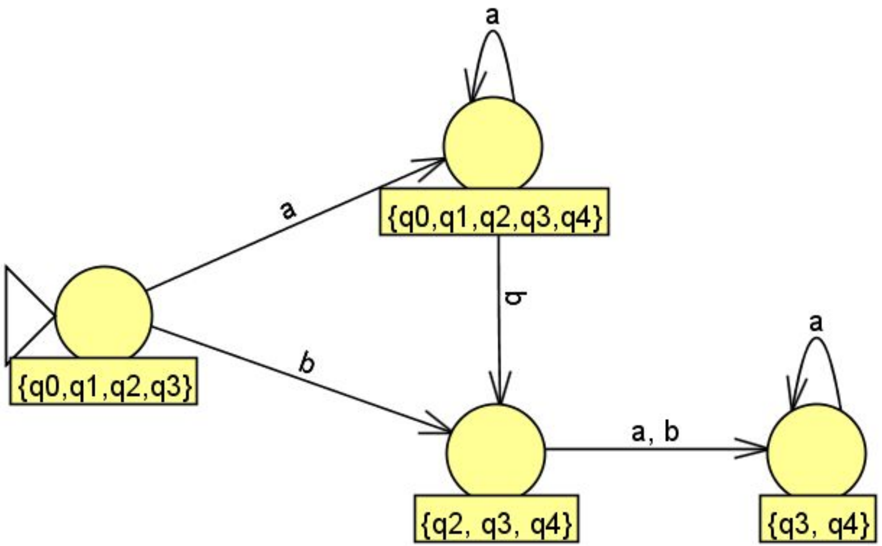
$\Delta''(\{q_2, q_3, q_4\}, a) = \{q_3, q_4\}$

nfa and dfa equivalence



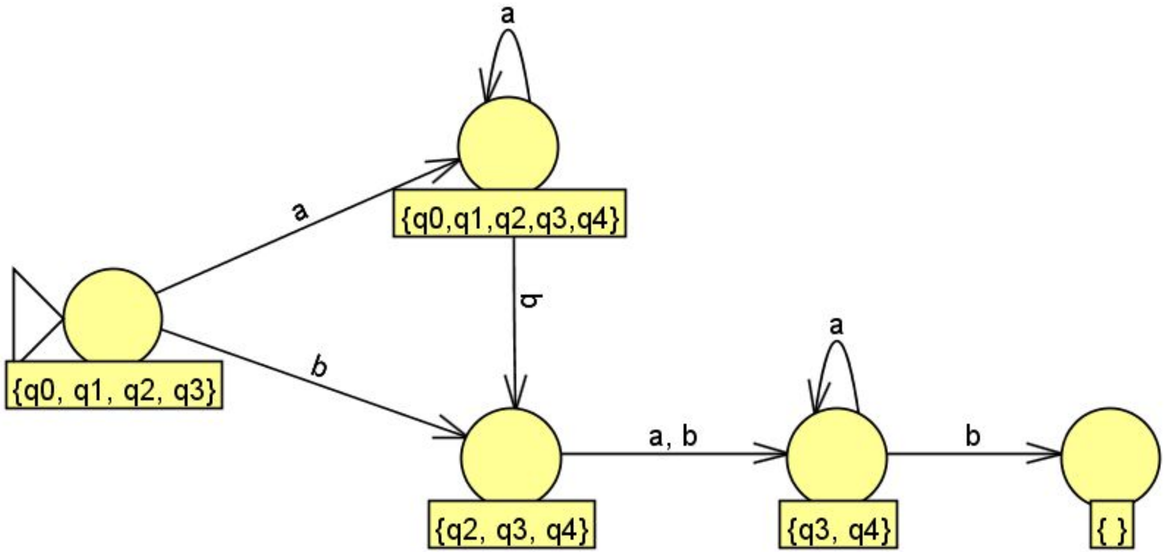
$\Delta'' (\{q_2, q_3, q_4\}, b) = \{q_3, q_4\}$

nfa and dfa equivalence



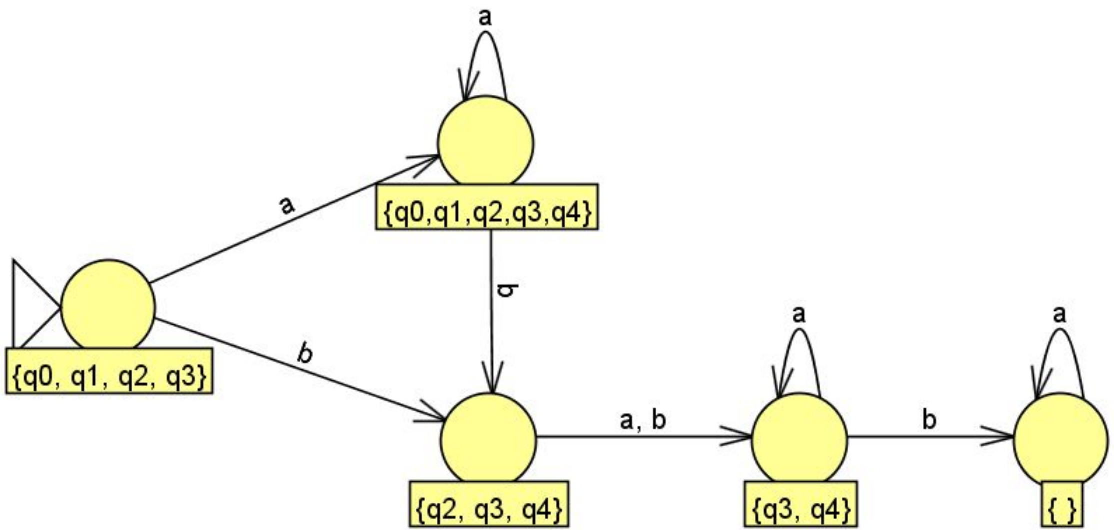
$\Delta'' (\{q_3, q_4\}, a) = \{q_3, q_4\}$

nfa and dfa equivalence



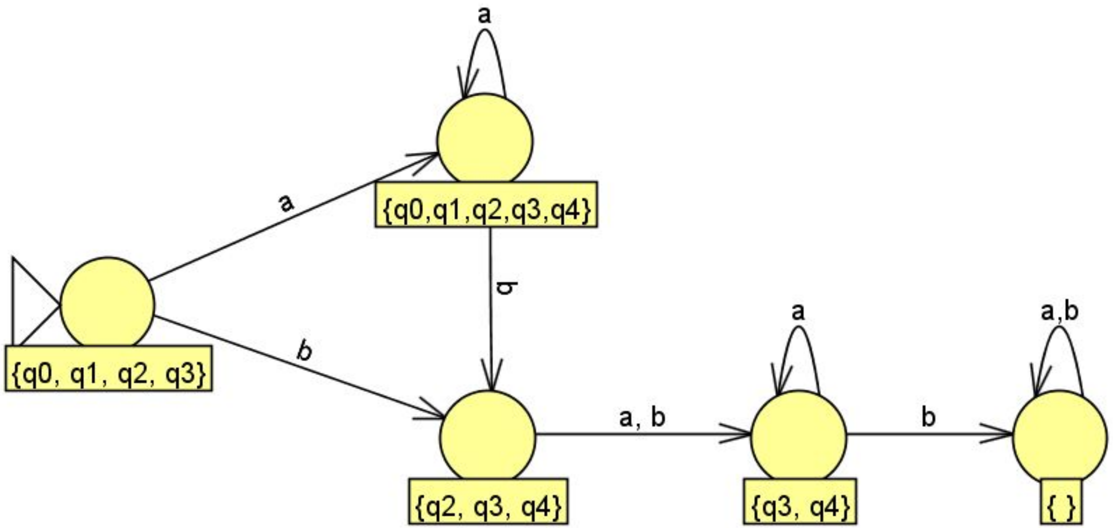
$\Delta''(\{q_3, q_4\}, b) = \emptyset$

nfa and dfa equivalence



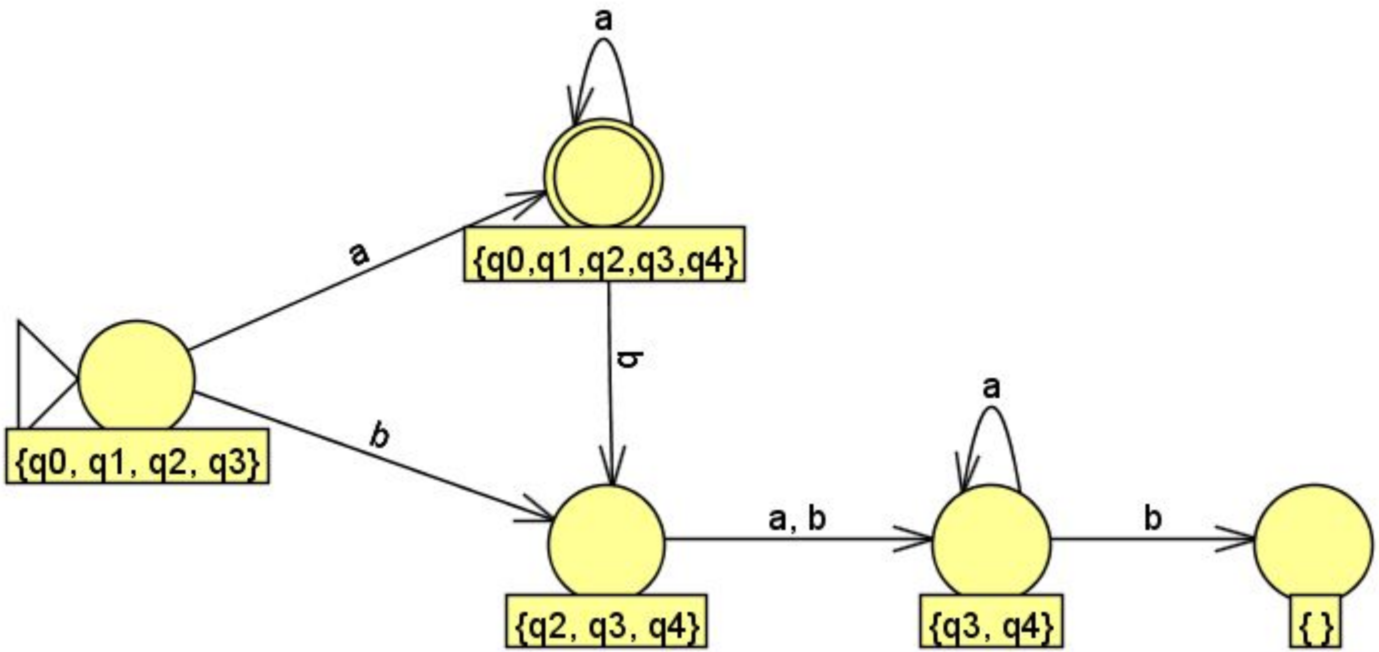
Δ'' (Ø, a) = Ø

nfa and dfa equivalence

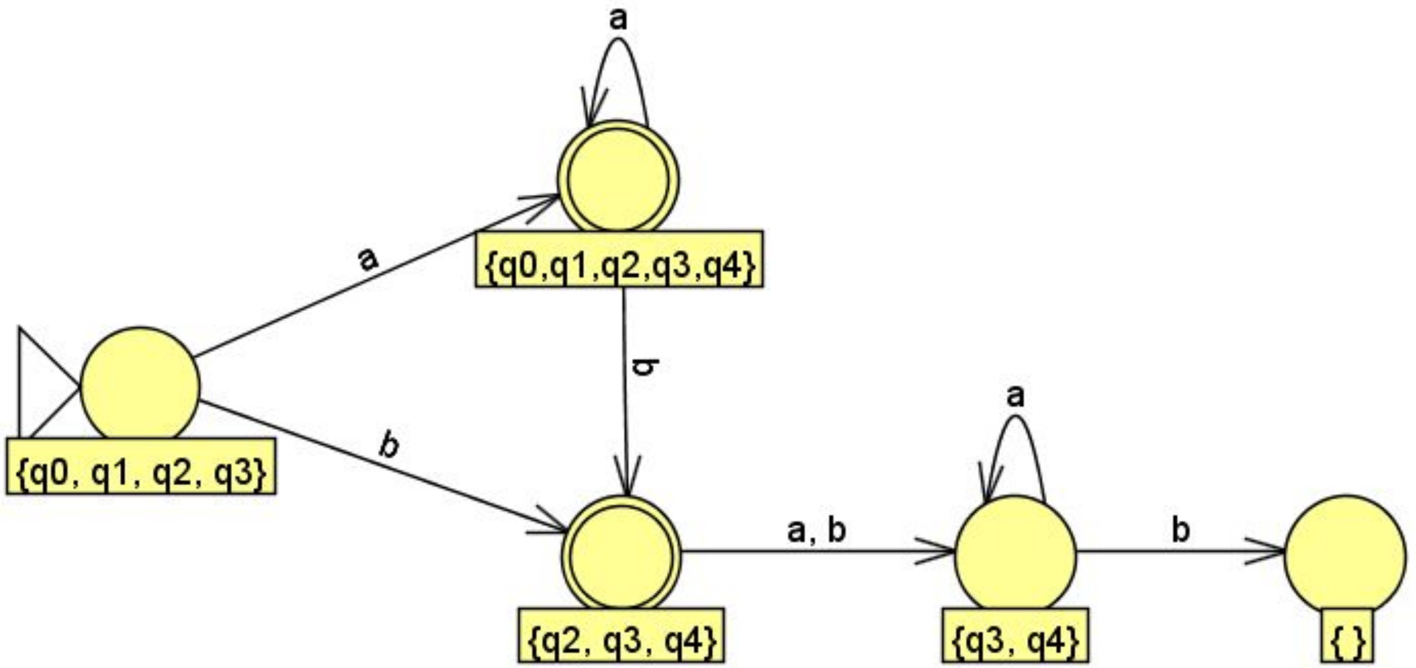


Δ'' (Ø, b) = Ø

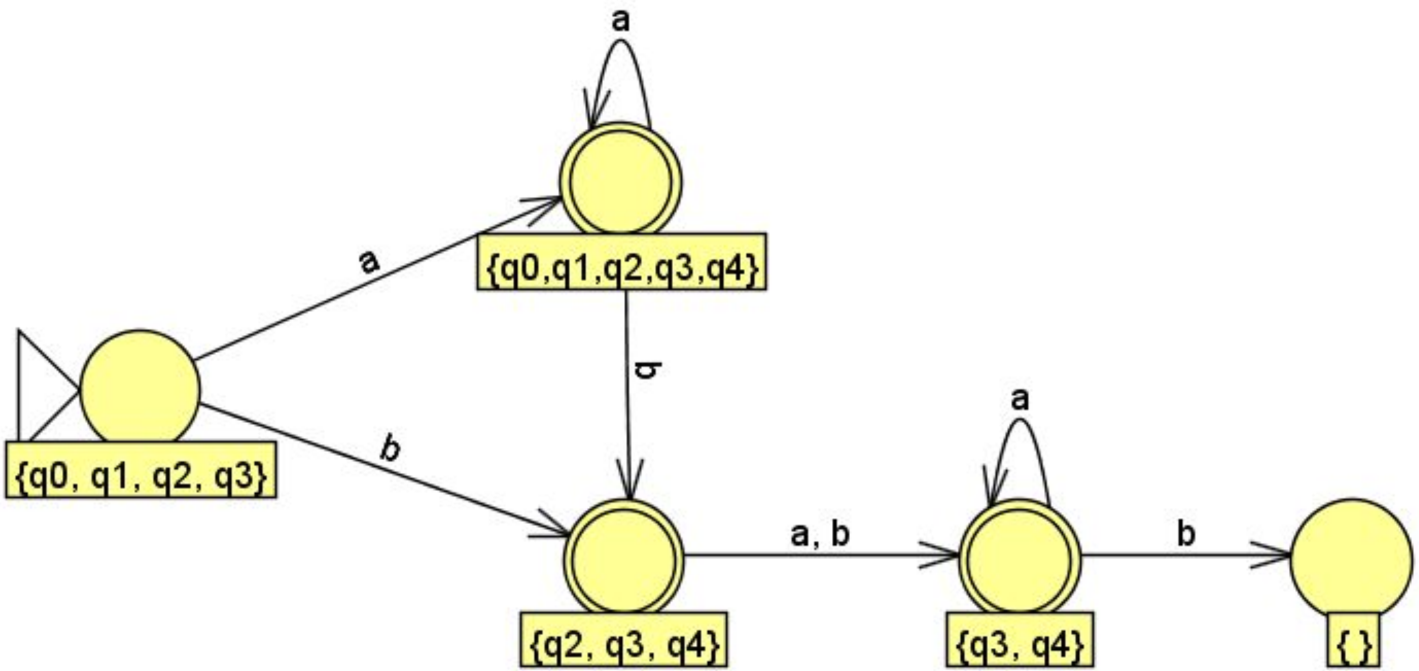
nfa and dfa equivalence



nfa and dfa equivalence



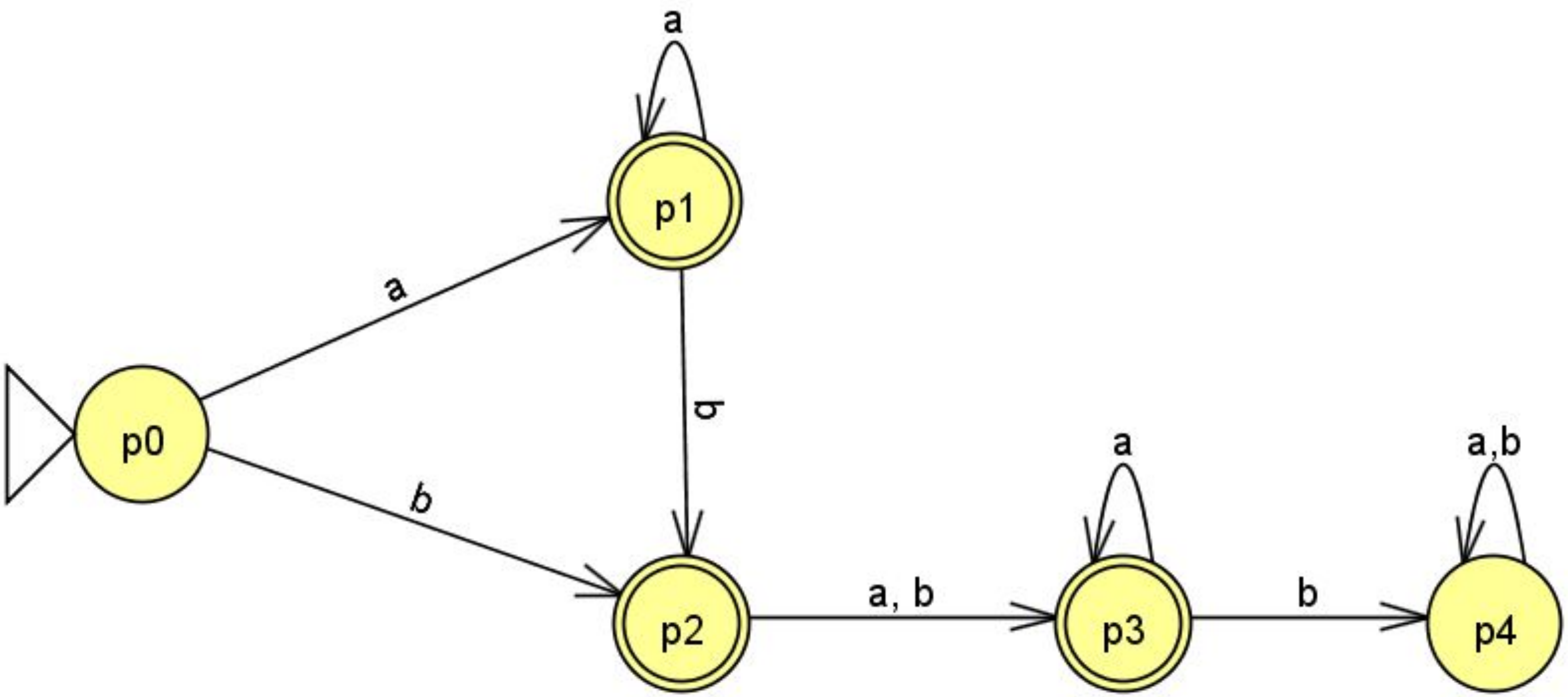
nfa and dfa equivalence



nfa and dfa equivalence

- $p_0 = \{q_0, q_1, q_2, q_3\}$
- $p_1 = \{q_0, q_1, q_2, q_3, q_4\}$
- $p_2 = \{q_2, q_3, q_4\}$
- $p_3 = \{q_3, q_4\}$
- $p_4 = \emptyset$
- Final states are those that have q_4

nfa and dfa equivalence



finite automata and regular expressions

- the class of languages accepted by finite automata remains the same even if a new and seemingly powerful feature, nondeterminism, is allowed
  - stability
  - The class of languages accepted by finite automata, deterministic or nondeterministic, is the same as the class of regular languages

finite automata and regular expressions

- Theorem: The class of languages accepted by finite automata is closed under
  - union
  - concatenation
  - Kleene star
  - complementation
  - intersection
- In each case we show how to construct an automaton M that accepts the appropriate language, given two automata M_1 and M_2
  - only M_1 in the cases of Kleene star and complementation)

union

- Let $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ be non-deterministic finite automata
  - construct a nondeterministic finite automaton M such that $L(M) = L(M_1) \cup L(M_2)$
  - $M = (K, \Sigma, \Delta, s, F)$
    - $K = K_1 \cup K_2 \cup \{s\}$
    - $F = F_1 \cup F_2$
    - $\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, e, s_1), (s, e, s_2)\}$

concatenation

- Let $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \Delta_2, s_2, F_2)$ be non-deterministic finite automata
  - construct a nondeterministic finite automaton M such that $L(M) = L(M_1) \circ L(M_2)$
  - $M = (K, \Sigma, \Delta, s, F)$
    - K = ?
    - F = ?
    - Δ = ?

Kleene Star

- Let $M_1 = (K_1, \Sigma, \Delta_1, s_1, F_1)$ a non-deterministic finite automaton
  - construct a nondeterministic finite automaton M such that $L(M) = L(M_1)^*$
  - $M = (K, \Sigma, \Delta, s, F)$
    - $K = ?$
    - $F = ?$
    - $\Delta = ?$

Complementation

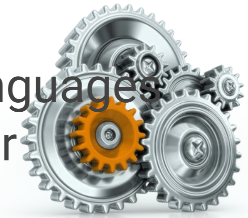
- Let $M = (K, \Sigma, \delta, s, F)$ a deterministic finite automaton
  - The complementary language $L' = \Sigma^* - L(M)$
  - $M = (K, \Sigma, \delta, s, F')$
    - $K = ?$
    - $\delta = ?$
    - $F' = ?$

intersection

- Recall definition of intersection
  - $L_1 \cap L_2$
    - $\Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$

finite automata and regular expression

- Theorem: A language is regular if and only if it is accepted by finite automaton
  - Only if:
    - the class of regular languages is the smallest class of languages containing the empty set ∅ and the singletons *a*, where *a* is a symbol, and closed under union, concatenation, and Kleene star.
    - It should be evident that the the empty set ∅ and the singletons are accepted by finite automata.
      - And by the previous theorem, the finite automaton languages are closed under union, concatenation, and Kleene star



finite automata and regular expression

- consider $(ab \cup aab)^*$

exercises

Construct non-deterministic FSM to accept each of the following languages:

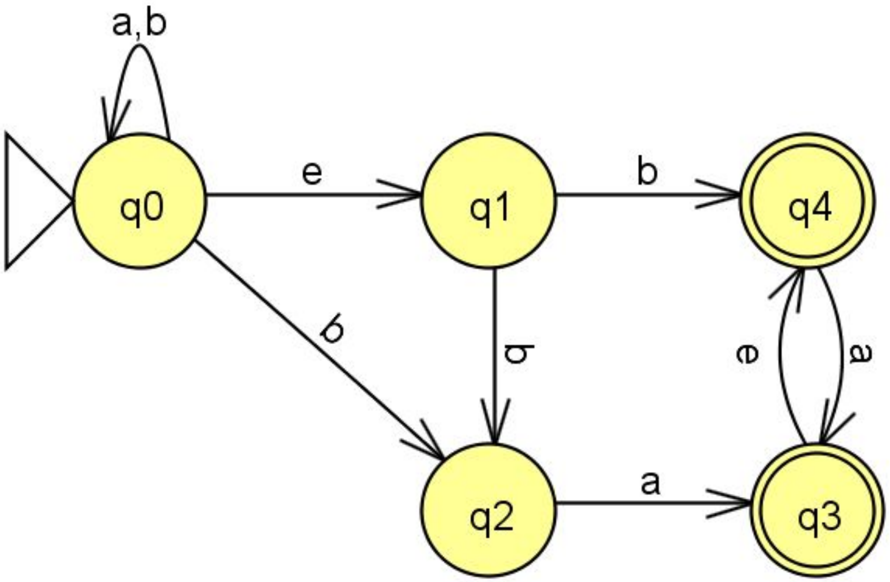
- {w ∈ {a, b}* : each ‘a’ in w is immediately preceded and followed by a ‘b’}
- {w ∈ {a, b}* : w has abab as a substring}
- {w ∈ {a, b}* : w has neither aa nor bb as a substring}
- {w ∈ {a, b}* : w has both ab and ba as substrings}
- {w ∈ {a, b}* : every odd position of w is a b}
- {w ∈ {a, b}* : w has an odd number of a's and an odd number of b's}
- {w ∈ {a, b}* : w has an odd number of a's and an even number of b's}

exercises

- construct a nondeterministic finite automaton for the following regular expressions using the construction proof on the closure properties of union, concatenation and Kleene star (assume Σ = {a, b, c}):
  - a*(ab U ba U e)b*
  - ((a U b)*(e U c)*)*
  - ((ab)* U (bc)*)ab

exercises

- construct the DFA equivalent of the following nondeterministic finite automaton



finite automata and regular expression

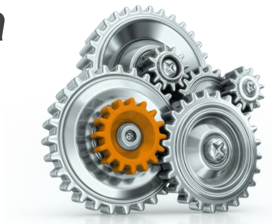
- Theorem: A language is regular if and only if it is accepted by finite automaton
  - if:
    - Given L(M), where M is a DFA, there is a regular expression R that generates L(R) = L(M)
    - proof by construction

finite automata and regular expression

- Let $M = (K, \Sigma, \Delta, s, F)$
  - construct a regular expression R such that $L(R) = L(M)$
  - $K = \{q_1, q_2, \ldots, q_n\}$ and $s = q_1$
  - For $i,j = 1, \ldots, n$ and $k = 0, \ldots, n$, define $R(i, j, k)$ as the set of all strings in Σ^* that may drive M from state q_i to state q_j without passing through any intermediate state numbered $k+1$ or greater
    - the endpoints q_i and q_j are allowed to be numbered higher than k.
    - when $k = n$, it follows that
      - $R(i,j,n) = \{w \in \Sigma^*: (q_i, w) \vdash^*_M (q_j, e)\}$.

finite automata and regular expression

- basis step: $k = 0$
  - no intermediate states at all
    - an edge from state i to j
    - a path of length 0 that consists of only some state i
  - if i=j
    - $\{e\} \cup \{a \in \Sigma \cup \{e\} : (q_i, a, q_j) \in \Delta\}$
  - if $i \neq j$
    - if no such symbol *a*, then $R(i,j,0) = \varnothing$
    - if there is exactly one such symbol *a*, then $R(i,j,0) = a$
    - if there are symbols $a_1, a_2, a_3, \ldots, a_k$, then $R(i,j,0) = a_1 \cup a_2 \cup a_3 \cup \ldots \cup a_k$
      - $\{a \in \Sigma \cup \{e\} : (q_i, a, q_j) \in \Delta\}$

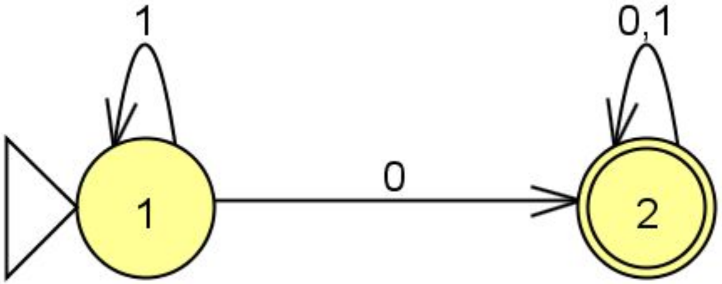


finite automata and regular expression

- induction
  - $R(i,j,k) = R(i,j,k-1) \cup R(i,k,k-1)\{R(k,k,k-1)\}^*R(k,j,k-1)$

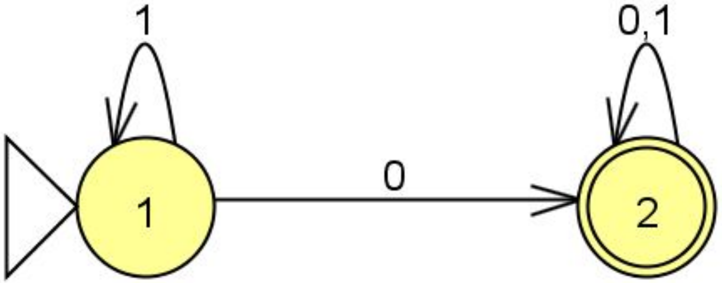
finite automata and regular expression

- $R(1,1,0) = e + 1$
- $R(1,2,0) = 0$
- $R(2,1,0) = \varnothing$
- $R(2,2,0) = e + 0 + 1$



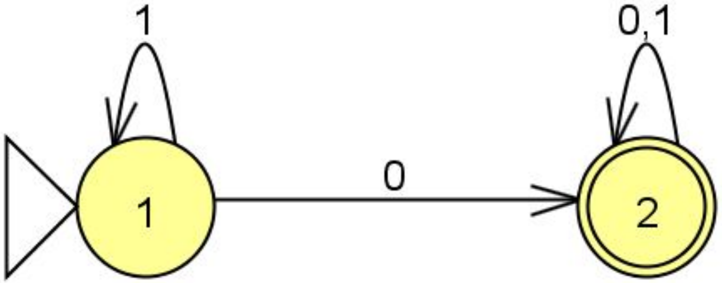
finite automata and regular expression

- R(i,j,1) = R(i,j,k-1) U R(i,k,k-1){R(k,k,k-1)}*R(k,j,k-1)
- R(i,j,1) = R(i,j,0) U R(i,k,0){R(k,k,0)}*R(k,j,0)



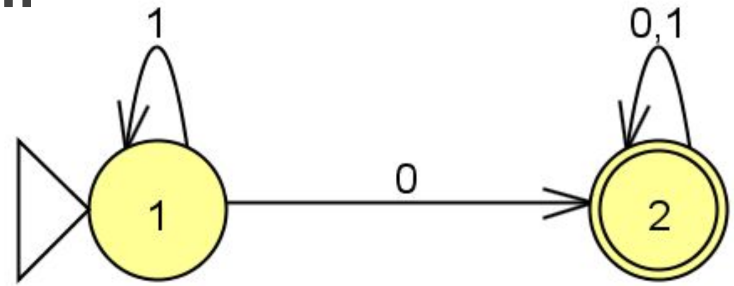
finite automata and regular expression

- $R(i,j,1) = R(i,j,0) \cup R(i,k,0)\{R(k,k,0)\}^*R(k,j,0)$
- $R(1,1,1) = e+1+(e+1)(e+1)^*(e+1)$
- $R(1,2,1) = 0+(e+1)(e+1)^*0$
- $R(2,1,1) = \varnothing+\varnothing(e+1)^*(e+1)$
- $R(2,2,1) = e + 0 + 1+\varnothing(e+1)^*0$



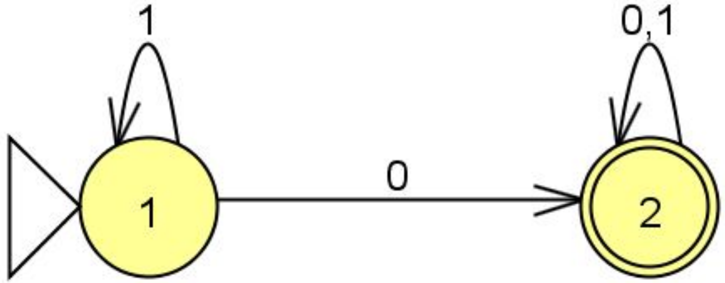
finite automata and regular expression

- R(i,j,2) = R(i,j,1) U R(i,k,1){R(k,k,1)}*R(k,j,1)
- R(1,1,2) = 1*+1*(e+0+1)*∅
- R(1,2,2) = 1*0+1*0(e+0+1)*(e+0+1)
- R(2,1,2) = ∅+(e+0+1)(e+0+1)*∅
- R(2,2,2) = e+0+1+(e+0+1)(e+0+1)*(e+0+1)

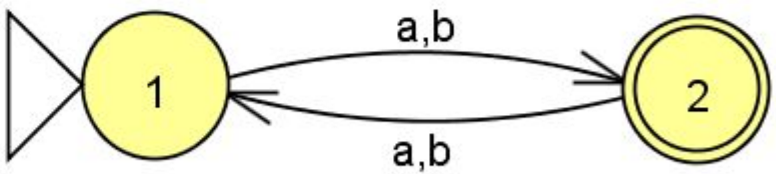


finite automata and regular expression

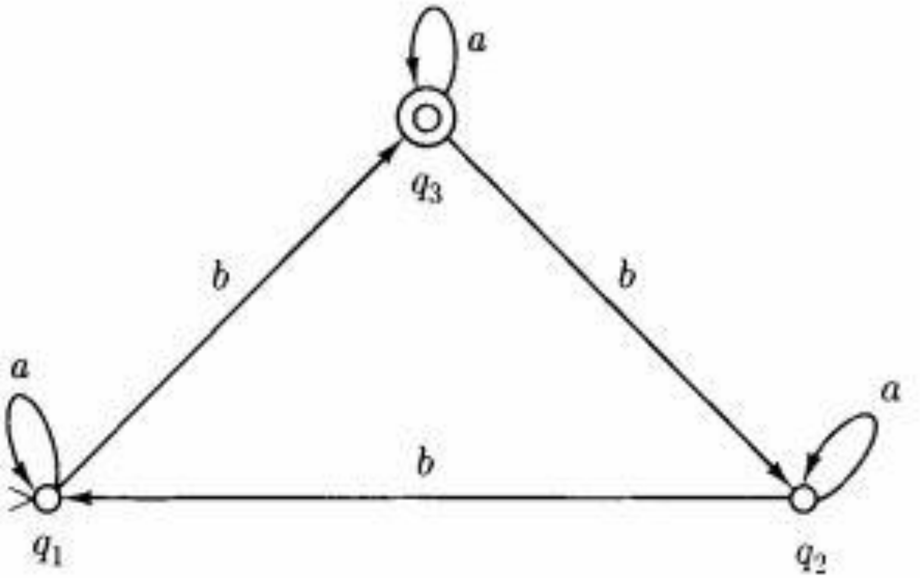
- R(i,j,2) = R(i,j,k-1) U R(i,k,k-1){R(k,k,k-1)}*R(k,j,k-1)
- R(i,j,2) = R(i,j,1) U R(i,k,1){R(k,k,1)}*R(k,j,1)



finite automata and regular expression

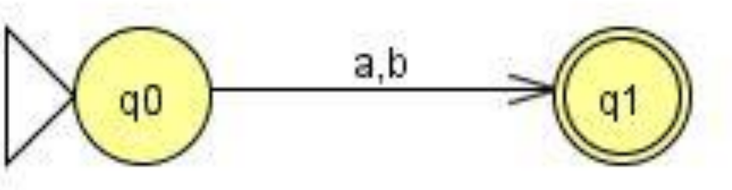


finite automata and regular expression



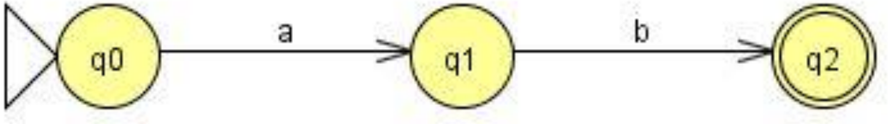
finite automata and regular expression

- state elimination
  - initial state should not have incoming edge
  - there should only be one final state
  - final state should not have outgoing edges
  - other than the initial state and final state, eliminate all remaining states



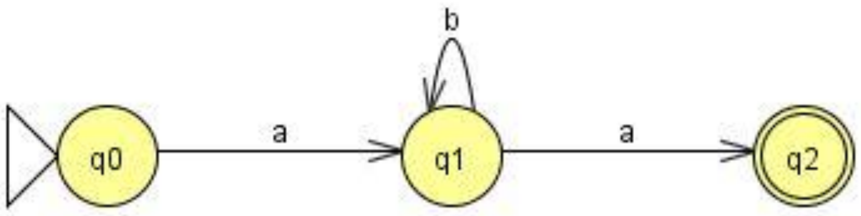
finite automata and regular expression

- state elimination
  - initial state should not have incoming edge
  - there should only be one final state
  - final state should not have outgoing edges
  - other than the initial state and final state, eliminate all remaining states



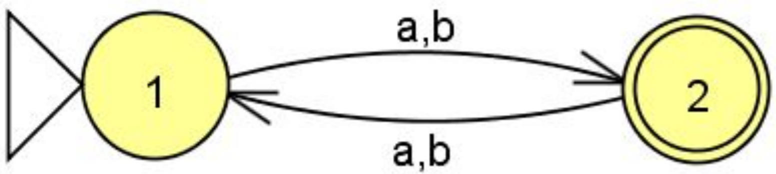
finite automata and regular expression

- state elimination
  - initial state should not have incoming edge
  - there should only be one final state
  - final state should not have outgoing edges
  - other than the initial state and final state, eliminate all remaining states



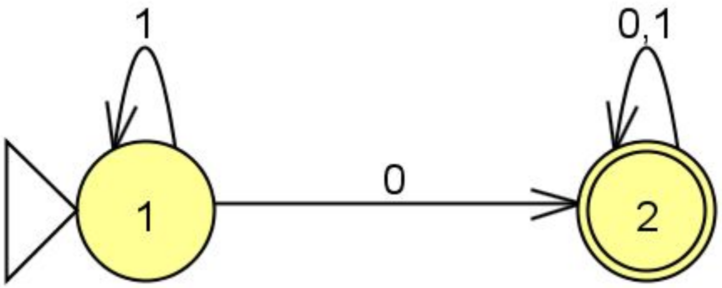
finite automata and regular expression

- state elimination



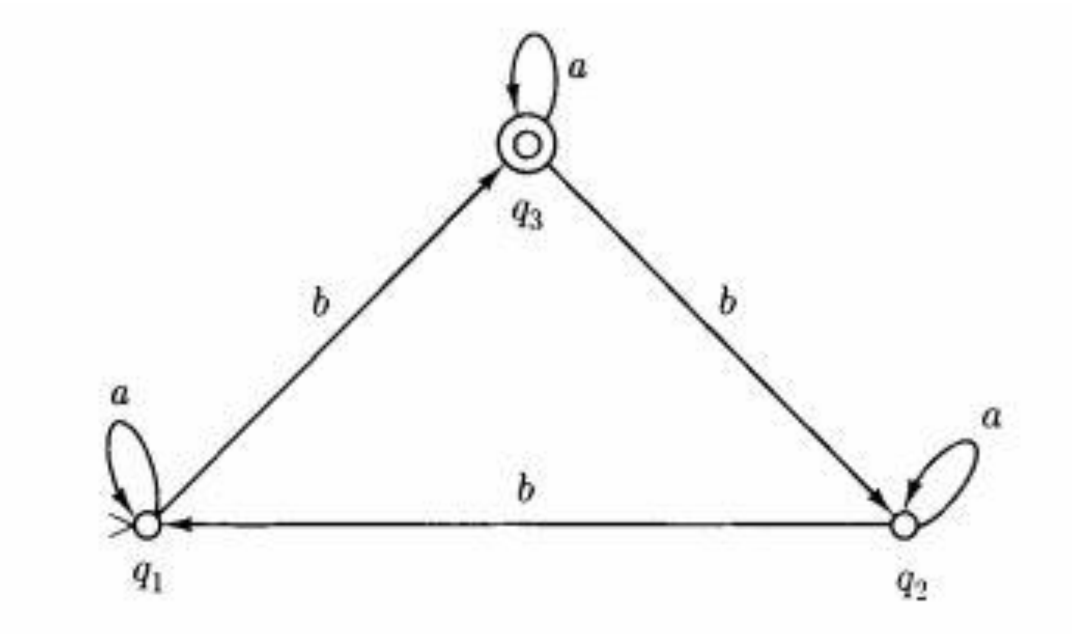
finite automata and regular expression

- state elimination



finite automata and regular expression

- state elimination



regular languages and non-regular languages

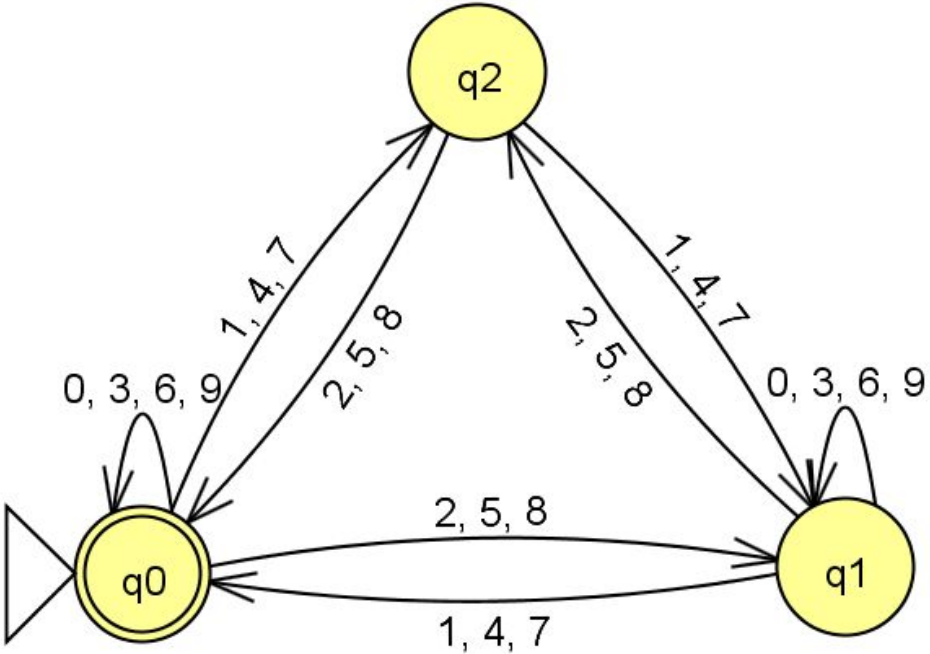
- regular languages are closed under a variety of operations
- regular languages may be specified either by regular expression or finite automata
- used singly or in combinations, they provide a variety of techniques for showing languages to be regular
  - let $\Sigma = \{0, 1, 2, \dots, 9\}$ and let $L \subset \Sigma^*$ be the set of decimal representations for nonnegative integers (without redundant leading 0's) divisible by 2 or 3
    - $0, 3, 6, 144 \in L$
    - $1, 0018, 47 \notin L$
  - show that L is regular

regular languages and non-regular languages

- let $\Sigma = \{0, 1, 2, \dots, 9\}$ and let $L \subset \Sigma^*$ be the set of decimal representations for nonnegative integers (without redundant leading 0's) divisible by 2 or 3
  - Let L_1 be the set of nonnegative integers
    - $0 \cup \{1, 2, \dots, 9\}\Sigma^*$
      - regular
  - Let L_2 be the set of nonnegative integers divisible by 2
    - $L_1 \cap \Sigma^*\{0, 2, 4, 6, 8\}$
      - regular

regular languages and non-regular languages

- Let L_3 be the intersection of L_1 and the L(M), with M below:



regular languages and non-regular languages

- let $\Sigma = \{0, 1, 2, \ldots, 9\}$ and let $L \subset \Sigma^*$ be the set of decimal representations for nonnegative integers (without redundant leading 0's) divisible by 2 or 3
  - $L = L_2 \cup L_3$
- With these techniques though, we have yet to find a way to formally show that a certain language is not regular
  - the number of regular expressions/automata, although infinite, it is countable
  - but the number of languages is uncountable

regular languages and non-regular languages

- properties shared by regular languages but not by certain non-regular languages
  - as a string is scanned left to right, the amount of memory that is required in order to determine at the end whether or not the string is in the language must be bounded, fixed in advance and dependent on the language, not the particular input string
    - $\{a^n b^n : n \geq 0\}$ is non-regular

regular languages and non-regular languages

- properties shared by regular languages but not by certain non-regular languages
  - regular languages with an infinite number of strings are represented by automata with cycles and regular expressions involving the Kleene star. Such languages must have infinite subsets with a certain simple repetitive structure that arises from the Kleene star in a corresponding regular expression or a cycle in the state diagram of a finite automaton.
    - $\{a^n : n \geq 1 \text{ is prime}\}$ is non-regular

regular languages and non-regular languages

- Theorem: Let L be a regular language. There is an integer $n \geq 1$ (pumping length) such that any string $w \in L$ with $|w| \geq n$ can be rewritten as $w = xyz$ such that $y \neq e$, $|xy| \leq n$, and $xy^iz \in L$ for each $i \geq 0$.
  - pumping lemma for regular languages

regular languages and non-regular languages

- Since L is regular, L is accepted by a deterministic finite automaton M. Suppose that n is the number of states of M, and let w be a string of length n or greater.
- Consider now the first n steps of the computation of M on w:
  - $(q_0, w_1w_2 \ldots w_n) \vdash_M (q_1, w_2w_3 \ldots w_n) \vdash_M \ldots \vdash_M (q_n, e)$
- Since M has only n states, and there are n + 1 configurations $(q_i, w_{i+1} \ldots , w_n)$ appearing in the computation above, by the **pigeonhole principle** there exist i and j, $0 \le i < j \le n$, such that $q_i = q_j$

regular languages and non-regular languages

- That is, the string $y = w_i w_{i+1} \ldots w_j$ drives M from state q_i back to state q_i, and this string is nonempty since $i < j$.
- But then this string could be removed from w, or repeated any number of times in w just after the jth symbol of w, and M would still accept this string
- That is, M accepts $xy^i z \in L$ for each $i \geq 0$, where $x = w_1 \ldots w_i$, and $z = w_{j+1} \ldots w_m$.
- Notice finally that the length of xy, the number we called j above, is by definition at most n, as required.
- h.n.

regular languages and non-regular languages

- Use the pumping lemma to show that the following are non-regular
  - $L = \{a^i b^i : i \geq 0\}$
  - $L = \{a^n : n \text{ is prime}\}$
  - $L = \{w \in \{a, b\}^* : w \text{ has an equal number of a's and b's}\}$
- proof by contradiction
  - assume that the language is regular
  - it has a pumping length, n
  - all strings longer than n can be pumped
  - find a string w in the language, s.t. $|w| \geq n$
  - divide w = xyz (all ways) and show that $xy^iz \notin L$ for some i

regular languages and non-regular languages

- $L = \{a^ib^i : i \geq 0\}$
  - assume L is regular
  - let n be the pumping length
  - let $w = a^nb^n$, a^5b^5 = aaaaabbbbb
  - divide w to xyz
    - case 1: y all a's
    - case 2: y all b's
    - case 3: combination of a's and b's
  - check if the different cases can be pumped
    - xy^iz for some $i \geq 0$

regular languages and non-regular languages

- case 1: y all a’s: aaaaabbbbb
  - i = 2
  - aaaaaaaabbbbb
- case 2: y all b’s: aaaaabbbbb
  - i = 2
  - aaaaabbbbbbbb
- case 3: combination of a’s and b’s: aaaaabbbbb
  - i = 2
  - aaaaabbbaabbbbb
- conditions
  - $xy^iz \in L$ and $i \geq 0$, $|y| > 0$, and $|xy| \leq n$

regular languages and non-regular languages

- $L = \{a^i b^i : i \geq 0\}$
  - assume L is regular
  - let n be the pumping length
  - let $w = a^n b^n$
  - rewrite w as xyz such that
    - $|xy| \leq n$
    - $y \neq e$
      - $y = a^i$, for some $i > 0$
  - $xz = a^{n-i} b^n \notin L$
    - contradicting the pumping theorem

regular languages and non-regular languages

- $L = \{a^n : n \text{ is prime}\}$
  - let $w = xyz$
  - $x = a^p$, $y = a^q$, and $z = a^r$ where $p, q \geq 0$ and $r > 0$
  - $xy^nz \in L$, for each $n \geq 0$
    - $p + nq + r$ should be prime
    - but this is impossible
      - let $n = p + 2q + r + 2$
      - $p + nq + r = (q+1)(p+2q+r)$
        - a product of two numbers > 1

regular languages and non-regular languages

- L = {w ∈ {a, b}* : w has an equal number of a's and b's}
  - sometimes it is useful to use closure properties in combination with the pumping lemma
  - recall that L(a*b*) is regular
  - if L is regular, then so should L ∩ L(a*b*)
    - closed under intersection
  - but this is a contradiction, why?